LilyPond

Das Notensatzprogramm

Notationsreferenz

Das LilyPond-Entwicklerteam

Dieses Handbuch stellt eine Referenz aller Notationsformen zur Verfügung, die mit LilyPond Version 2.14.1 erstellt werden können. Es wird vorausgesetzt, dass der Leser mit dem Abschnitt "Handbuch zum Lernen" in *Handbuch zum Lernen* vertraut ist.

Zu mehr Information, wie dieses Handbuch unter den anderen Handbüchern positioniert, oder um dieses Handbuch in einem anderen Format zu lesen, besuchen Sie bitte Abschnitt "Manuals" in *Allgemeine Information*.

Wenn Ihnen Handbücher fehlen, finden Sie die gesamte Dokumentation unter `http://www.lilypond.org/`.

Copyright © 1999–2011 bei den Autoren.

The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.

Die Übersetzung der folgenden Lizenzanmerkung ist zur Orientierung für Leser, die nicht Englisch sprechen. Im rechtlichen Sinne ist aber nur die englische Version gültig.

> Es ist erlaubt, dieses Dokument unter den Bedingungen der GNU Free Documentation Lizenz (Version 1.1 oder spätere, von der Free Software Foundation publizierte Versionen, ohne invariante Abschnitte), zu kopieren, zu verbreiten und/oder zu verändern. Eine Kopie der Lizenz ist im Abschnitt "GNU Free Documentation License" angefügt.
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections. A copy of the license is included in the section entitled "GNU Free Documentation License".

Für LilyPond Version 2.14.1

Inhaltsverzeichnis

1 Musikalische Notation ... 1

1.1 Tonhöhen ... 1
1.1.1 Tonhöhen setzen ... 1
Absolute Oktavenbezeichnung ... 1
Relative Oktavenbezeichnung ... 2
Versetzungszeichen ... 5
Notenbezeichnungen in anderen Sprachen ... 7
1.1.2 Viele Tonhöhen gleichzeitig verändern ... 9
Oktavenüberprüfung ... 9
Transposition ... 10
Umkehrung ... 13
Krebs ... 13
Modale Transformierungen ... 14
1.1.3 Tonhöhen anzeigen lassen ... 16
Notenschlüssel ... 16
Tonartbezeichnung ... 20
Oktavierungsklammern ... 21
Transposition von Instrumenten ... 22
Automatische Versetzungszeichen ... 24
Tonumfang ... 31
1.1.4 Notenköpfe ... 33
Besondere Notenköpfe ... 34
Easy-Notation-Notenköpfe ... 35
Notenköpfe mit besonderen Formen ... 37
Improvisation ... 40
1.2 Rhythmus ... 40
1.2.1 Rhythmen eingeben ... 41
Tondauern ... 41
Andere rhythmische Aufteilungen ... 43
Tondauern skalieren ... 47
Bindebögen ... 48
1.2.2 Pausen eingeben ... 51
Pausen ... 51
Unsichtbare Pausen ... 53
Ganztaktpausen ... 54
1.2.3 Rhythmen anzeigen lassen ... 58
Taktangabe ... 58
Metronomangabe ... 63
Auftakte ... 66
Musik ohne Metrum ... 67
Polymetrische Notation ... 68
Automatische Aufteilung von Noten ... 71
Melodierhythmus anzeigen ... 72
1.2.4 Balken ... 74
Automatische Balken ... 74
Einstellung von automatischen Balken ... 77
Manuelle Balken ... 84
Gespreizte Balken ... 86

1.2.5 Takte ... 87
Taktstriche ... 87
Taktzahlen ... 92
Takt- und Taktzahlüberprüfung ... 96
Übungszeichen ... 97
1.2.6 Besondere rhythmische Fragen ... 99
Verzierungen ... 99
An Kadenzen ausrichten ... 104
Verwaltung der Zeiteinheiten ... 105
1.3 Ausdrucksbezeichnungen ... 106
1.3.1 Ausdrucksbezeichnungen an Noten angehängt ... 106
Artikulationszeichen und Verzierungen ... 106
Dynamik ... 109
Neue Lautstärkezeichen ... 114
1.3.2 Ausdrucksbezeichnungen als Bögen ... 116
Legatobögen ... 116
Phrasierungsbögen ... 119
Atemzeichen ... 120
Glissando zu unbestimmter Tonhöhe ... 121
1.3.3 Ausdrucksbezeichnungen als Linien ... 122
Glissando ... 122
Arpeggio ... 123
Triller ... 126
1.4 Wiederholungszeichen ... 128
1.4.1 Lange Wiederholungen ... 129
Normale Wiederholungen ... 129
Manuelle Wiederholungszeichen ... 134
Ausgeschriebene Wiederholungen ... 136
1.4.2 Kurze Wiederholungen ... 138
Prozent-Wiederholungen ... 138
Tremolo-Wiederholung ... 140
1.5 Gleichzeitig erscheinende Noten ... 142
1.5.1 Eine einzelne Stimme ... 142
Noten mit Akkorden ... 142
Akkord-Wiederholungen ... 144
Gleichzeitige Ausdrücke ... 144
Cluster ... 145
1.5.2 Mehrere Stimmen ... 146
Mehrstimmigkeit in einem System ... 146
Stimmenstile ... 149
Auflösung von Zusammenstößen ... 149
Automatische Kombination von Stimmen ... 154
Musik parallel notieren ... 158
1.6 Notation auf Systemen ... 161
1.6.1 Systeme anzeigen lassen ... 161
Neue Notensysteme erstellen ... 161
Systeme gruppieren ... 162
Verschachtelte Notensysteme ... 166
Systeme trennen ... 168
1.6.2 Einzelne Systeme verändern ... 169
Das Notensystem ... 169
Ossia-Systeme ... 171
Systeme verstecken ... 175
1.6.3 Orchesterstimmen erstellen ... 178

Instrumentenbezeichnungen ... 178
Stichnoten ... 181
Stichnoten formatieren ... 185
1.7 Anmerkungen ... 189
1.7.1 Innerhalb des Systems ... 189
Auswahl der Notations-Schriftgröße ... 189
Fingersatzanweisungen ... 191
Unsichtbare Noten ... 193
Farbige Objekte ... 194
Klammern ... 195
Hälse ... 196
1.7.2 Außerhalb des Notensystems ... 196
Erklärungen in Ballonform ... 197
Gitternetzlinien ... 198
Analyseklammern ... 200
1.8 Text ... 201
1.8.1 Text eingeben ... 201
Textarten ... 201
Text mit Verbindungslinien ... 202
Textartige Zeichen ... 204
Separater Text ... 208
1.8.2 Text formatieren ... 210
Textbeschriftung (Einleitung) ... 210
Überblick über die wichtigsten Textbeschriftungsbefehle ... 211
Textausrichtung ... 213
Graphische Notation innerhalb einer Textbeschriftung ... 217
Musikalische Notation innerhalb einer Textbeschriftung ... 219
Textbeschriftung über mehrere Seiten ... 222
1.8.3 Schriftarten ... 223
Was sind Schriftarten ... 223
Schriftarten für einen Eintrag ... 224
Schriftart des gesamten Dokuments ... 225

2 Spezielle Notation ... 226

2.1 Notation von Gesang ... 226
2.1.1 Übliche Notation für Vokalmusik ... 226
Referenz für Vokalmusik ... 226
Eingabe von Text ... 227
Text an einer Melodie ausrichten ... 228
Automatische Silbendauern ... 230
Manuelle Silbendauer ... 233
Mehrere Silben zu einer Note ... 234
Mehrere Noten zu einer Silbe ... 235
Fülllinien und Trennstriche ... 238
2.1.2 Techniken für die Gesangstextnotation ... 238
Mit Gesangstexten und Bezeichnern arbeiten ... 239
Gesangstext vertikal verschieben ... 240
Silben horizontal verschieben ... 245
Gesangstext und Wiederholungen ... 247
Getrennte Texte ... 254
2.1.3 Strophen ... 255
Strophennummern hinzufügen ... 255
Lautstärkebezeichnung zu Strophen hinzufügen ... 255
Sängernamen zu Strophen hinzufügen ... 256

Strophen mit unterschiedlichem Rhythmus ... 256
Die Strophen am Ende ausdrucken ... 259
Die Strophen am Ende in mehreren Spalten drucken ... 260
2.1.4 Lieder ... 262
Verweise für Lieder ... 262
Liedblätter ... 262
2.1.5 Chormusik ... 263
Verweise für Chormusik ... 263
Partiturbeispiele für Chormusik ... 264
Geteilte Stimmen ... 265
2.1.6 Oper und Musical ... 266
Verweise für Oper und Musical ... 266
Namen von Figuren ... 267
Musikalische Stichnoten ... 269
Gesprochene Musik ... 273
Dialog zur Musik ... 273
2.1.7 Psalmengesänge und Hymnen ... 274
Verweise für Psalmen und Hymnen ... 274
Kirchengesang notieren ... 275
Einen Psalm notieren ... 282
Unvollständige Takte in Hymnen ... 285
2.1.8 Alte Vokalmusik ... 287
2.2 Tasteninstrumente und andere Instrumente mit mehreren Systemen ... 287
2.2.1 Übliche Notation für Tasteninstrumente ... 288
Referenz für Tasteninstrumente ... 288
Notensysteme manuell verändern ... 289
Automatischer Systemwechsel ... 290
Stimmführungslinien ... 292
Hälse über beide Systeme ... 292
2.2.2 Klavier ... 294
Klavierpedal ... 294
2.2.3 Akkordeon ... 295
Diskant-Symbole ... 295
2.2.4 Harfe ... 299
Referenzen für Harfe ... 299
Harfenpedal ... 299
2.3 Bundlose Saiteninstrumente ... 300
2.3.1 Übliche Notation für bundlose Saiteninstrumente ... 300
Hinweise für bundlose Saiteninstrumente ... 300
Bezeichnung des Bogens ... 301
Flageolett ... 301
Bartók-Pizzicato ... 302
2.4 Saiteninstrumente mit Bünden ... 302
2.4.1 Übliche Notation für Saiteninstrumente mit Bünden ... 303
Referenz für Saiteninstrumente mit Bünden ... 303
Seitennummerbezeichnung ... 303
Standardtabulaturen ... 305
Angepasste Tabulaturen ... 317
Bund-Diagramm-Beschriftung ... 321
Vordefinierte Bund-Diagramme ... 330
Automatische Bund-Diagramme ... 340
Fingersatz der rechten Hand ... 343
2.4.2 Gitarre ... 344
Position und Barré anzeigen ... 345

Flageolett und gedämpfte Noten ... 345
Powerakkorde anzeigen ... 346
2.4.3 Banjo ... 348
Banjo-Tabulaturen ... 348
2.5 Schlagzeug ... 349
2.5.1 Übliche Notation für Schlagzeug ... 349
Referenz für Schlagzeug ... 349
Grundlagen der Schlagzeugnotation ... 349
Trommelwirbel ... 350
Schlagzeug mit Tonhöhe ... 351
Schlagzeugsysteme ... 351
Eigene Schlagzeugsysteme ... 353
Geisternoten ... 357
2.6 Blasinstrumente ... 357
2.6.1 Übliche Notation für Bläser ... 358
Referenz für Blasinstrumente ... 358
Fingersatz ... 359
2.6.2 Dudelsack ... 361
Dudelsack-Definitionen ... 361
Dudelsack-Beispiele ... 362
2.6.3 Holzbläser ... 363
2.6.3.1 Holzbläserdiagramme ... 363
2.7 Notation von Akkorden ... 371
2.7.1 Akkord-Modus ... 371
Überblick über den Akkord-Modus ... 371
Übliche Akkorde ... 372
Erweiterte und modifizierte Akkorde ... 374
2.7.2 Akkorde anzeigen ... 376
Akkordbezeichnungen drucken ... 376
Akkordbezeichnungen anpassen ... 379
2.7.3 Generalbass ... 383
Grundlagen des Bezifferten Basses ... 384
Eingabe des Generalbass' ... 384
Generalbass anzeigen ... 387
2.8 Zeitgenössische Musik ... 390
2.8.1 Tonhöhe und Harmonie in zeitgenössischer Musik ... 390
Verweise zu Tonhöhe und Harmonie in zeitgenössischer Musik ... 390
Mikrotonale Notation ... 391
Zeitgenössische Tonartvorzeichnung und Harmonie ... 391
2.8.2 Zeitgenössische Notation von Rhythmen ... 391
Verweise für zeitgenössische Benutzung von Rhythmus ... 391
N-tolen in zeitgenössischer Musik ... 391
Zeitgenössische Taktarten ... 391
Erweiterte polymetrische Notation ... 391
Balken in zeitgenössischer Musik ... 391
Taktstriche in zeitgenössischer Musik ... 391
2.8.3 Graphische Notation ... 391
2.8.4 Zeitgenössische Partiturtechniken ... 391
2.8.5 Neue Instrumententechniken ... 391
2.8.6 Leseliste und interessante Referenzpartituren ... 391
Bücher und Artikel über zeitgenössische Notation ... 391
Partituren und Musikbeispiele ... 391
2.9 Notation von alter Musik ... 391
2.9.1 Überblick über die unterstützten Stile ... 392

2.9.2 Alte Notation – Allgemeines ... 393
Vordefinierte Umgebungen ... 393
Ligaturen ... 394
Custodes ... 394
Unterstützung für Generalbass ... 395
2.9.3 Mesurale Musik setzen ... 395
Mensural-Kontexte ... 395
Mensurale Schlüssel ... 396
Mensurale Taktartenbezeichnungen ... 397
Mensurale Notenköpfe ... 398
Mensurale Fähnchen ... 399
Mensurale Pausen ... 400
Mensurale Versetzungszeichen und Tonartbezeichnung ... 400
Vorgeschlagene Versetzungszeichen (*musica ficta*) ... 401
Weiße Mensuralligaturen ... 402
2.9.4 Gregorianischen Choral setzen ... 403
Gregorianische Gesangs-Kontexte ... 403
Gregorianische Schlüssel ... 404
Gregorianische Versetzungszeichen und Tonartbezeichnung ... 405
Divisiones ... 405
Artikulationszeichen des Gregorianischen Chorals ... 406
Augmentationspunkte (*morae*) ... 407
Ligaturen der gregorianischen Quadratnotation ... 407
2.9.5 Musiksatz Alter Musik in der Praxis – Szenarien und Lösungen ... 414
Incipite ... 414
Mensurstriche ... 414
Gregorianischen Choral transkribieren ... 415
Alte und moderne Edition aus einer Quelldatei ... 418
Herausgeberische Anmerkungen ... 418
2.10 Weltmusik ... 418
2.10.1 Übliche Notation für nichteuropäische Musik ... 418
Erweiterung von Notation und Stimmungssystemen ... 418
2.10.2 Arabische Musik ... 419
Referenz für arabische Musik ... 420
Arabische Notenbezeichnungen ... 420
Arabische Tonarten ... 421
Arabische Taktarten ... 422
Arabische Notenbeispiele ... 424
Weitere Literatur zur arabischen Musik ... 424
2.10.3 Türkische klassische Musik ... 425
Verweise für türkische klassische Musik ... 425
Türkische Notenbezeichnungen ... 425

3 Allgemeine Eingabe und Ausgabe ... 427

3.1 Eingabestruktur ... 427
3.1.1 Struktur einer Partitur ... 427
3.1.2 Mehrere Partituren in einem Buch ... 428
3.1.3 Mehrere Ausgabedateien aus einer Eingabedatei ... 429
3.1.4 Dateinamen der Ausgabedateien ... 430
3.1.5 Die Dateistruktur ... 431
3.2 Titel ... 433
3.2.1 Titel erstellen ... 433
3.2.2 Eigene Kopf- und Fußzeilen sowie Titel ... 436
3.2.3 Verweis auf die Seitenzahlen ... 438

3.2.4 Inhaltsverzeichnis ... 439
3.3 Arbeiten an Eingabe-Dateien ... 441
3.3.1 LilyPond-Dateien einfügen ... 441
3.3.2 Verschiedene Editionen aus einer Quelldatei ... 442
Variablen benutzen ... 442
Marken benutzen ... 444
Globale Einstellungen benutzen ... 447
3.3.3 Zeichenkodierung ... 447
3.3.4 LilyPond-Notation anzeigen ... 448
3.4 Ausgabe kontrollieren ... 449
3.4.1 Notationsfragmente extrahieren ... 449
3.4.2 Korrigierte Musik überspringen ... 449
3.4.3 Alternative Ausgabeformate ... 450
3.4.4 Die Notationsschriftart verändern ... 450
3.5 MIDI-Ausgabe ... 451
3.5.1 MIDI-Dateien erstellen ... 451
Instrumentenbezeichnungen ... 452
3.5.2 Der MIDI-Block ... 453
3.5.3 Was geht in die MIDI-Ausgabe ... 454
In MIDI unterstützt ... 454
In MIDI nicht unterstützt ... 454
3.5.4 Wiederholungen im MIDI ... 454
3.5.5 MIDI-Lautstärke kontrollieren ... 455
Dynamik-Zeichen ... 455
MIDI-Lautstärke ... 456
Verschiedene Instrumente angleichen (i) ... 457
Verschiedene Instrumente angleichen (ii) ... 458
3.5.6 Schlagzeug in MIDI ... 459

4 Abstände ... 461

4.1 Seitenlayout ... 461
4.1.1 Die `\paper`-Umgebung ... 461
4.1.2 Papierformat und automatische Skalierung ... 462
Das Papierformat einstellen ... 462
Automatische Skalierung auf ein Papierformat ... 463
Vertikale `\paper`-Variablen mit festen Abständen ... 463
Vertikale `\paper`-Variablen mit flexiblen Abständen ... 464
Struktur der Alisten für flexible vertikale Abstände ... 464
Liste der flexiblen vertikalen Abstandsvariblen in `\paper` ... 465
\paper-Variablen für horizontale Abstände ... 466
\paper-Variablen für Breite und Ränder ... 466
`\paper`-Variablen für zweiseitigen Satz ... 467
`\paper`-Variablen für Verschiebungen und Einrückungen ... 468
4.1.3 Andere `\paper`-Variablen ... 468
`\paper`-Variablen für den Zeilenumbruch ... 468
`\paper`-Variablen für den Seitenumbruch ... 469
`\paper`-Variablen für Seitenzahlen ... 470
Verschiedene `\paper`-Variablen ... 470
4.2 Partiturlayout ... 471
4.2.1 Die `ayout`-Umgebung ... 471
4.2.2 Die Notensystemgröße einstellen ... 472
4.3 Umbrüche ... 473
4.3.1 Zeilenumbrüche ... 473
4.3.2 Seitenumbrüche ... 475

4.3.3 Optimale Seitenumbrüche ... 476
4.3.4 Optimale Umbrüche zum Blättern ... 476
4.3.5 Minimale Seitenumbrüche ... 477
4.3.6 Ausdrückliche Umbrüche ... 478
4.3.7 Eine zusätzliche Stimme für Umbrüche benutzen ... 479
4.4 Vertikale Abstände ... 481
4.4.1 Flexible vertikale Abstände in Systemgruppen ... 482
Eigenschaften für Abstände innerhalb von Systemgruppen ... 482
Abstände von nicht gruppierten Notensystemen ... 485
Abstände von gruppierten Notensystemen ... 487
Abstände von nicht-Notensystemzeilen ... 488
4.4.2 Explizite Positionierung von Systemen ... 489
4.4.3 Vermeidung von vertikalen Zusammenstößen ... 496
4.5 Horizontale Abstände ... 497
4.5.1 Überblick über horizontale Abstände ... 497
4.5.2 Eine neuer Bereich mit anderen Abständen ... 499
4.5.3 Horizontale Abstände verändern ... 499
4.5.4 Zeilenlänge ... 501
4.5.5 Proportionale Notation ... 502
4.6 Die Musik auf weniger Seiten zwingen ... 509
4.6.1 Abstände anzeigen lassen ... 509
4.6.2 Abstände verändern ... 511

5 Standardeinstellungen verändern ... 513

5.1 Interpretationskontexte ... 513
5.1.1 Was sind Kontexte? ... 513
Score – der Vater aller Kontexte ... 513
Oberste Kontexte – Container für Systeme ... 513
Mittlere Kontexte – Systeme ... 514
Unterste Kontexte – Stimmen ... 514
5.1.2 Kontexte erstellen ... 515
5.1.3 Kontexte am Leben halten ... 516
5.1.4 Umgebungs-Plugins verändern ... 519
5.1.5 Die Standardeinstellungen von Kontexten ändern ... 521
5.1.6 Neue Kontexte definieren ... 522
5.1.7 Kontexte aneinander ausrichten ... 524
5.2 Die Referenz der Programminterna erklärt ... 525
5.2.1 Zurechtfinden in der Programmreferenz ... 525
5.2.2 Layout-Schnittstellen ... 526
5.2.3 Die Grob-Eigenschaften ... 527
5.2.4 Benennungskonventionen ... 528
5.3 Eigenschaften verändern ... 528
5.3.1 Grundlagen zum Verändern von Eigenschaften ... 528
5.3.2 Der `\set`-Befehl ... 529
5.3.3 Der `\\override`-Befehl ... 531
5.3.4 Der `\\tweak`-Befehl ... 533
5.3.5 `\\set` versus `\\override` ... 534
5.3.6 Alisten verändern ... 534
5.4 Nützliche Konzepte und Eigenschaften ... 536
5.4.1 Eingabe-Modi ... 536
5.4.2 Richtung und Platzierung ... 538
5.4.3 Reihenfolge des Kontextlayouts ... 539
5.4.4 Abstände und Maße ... 539
5.4.5 Eigenschaften des Staff-Symbols ... 540

5.4.6 Strecker ... 541
Das `spanner-interface` benutzen ... 541
Das `line-spanner-interface` benutzen ... 543
5.4.7 Sichtbarkeit von Objekten ... 546
Einen `stencil` entfernen ... 546
Objekten unsichtbar machen ... 546
Objekte weißmalen ... 547
`break-visibility` (unsichtbar machen) benutzen ... 547
Besonderheiten ... 549
5.4.8 Linienstile ... 551
5.4.9 Drehen von Objekten ... 552
Drehen von Objekten ... 552
Textbeschriftung drehen ... 552
5.5 Fortgeschrittene Optimierungen ... 552
5.5.1 Objekte ausrichten ... 553
`X-offset` und `Y-offset` direkt setzen ... 553
Das `side-position-interface` benutzen ... 554
Das `self-alignment-interface` benutzen ... 554
Benutzung des `break-alignable-interface` ... 556
5.5.2 Vertikale Gruppierung der grafischen Objekte („grob“s) ... 558
5.5.3 stencils verändern ... 558
5.5.4 Formen verändern ... 559
Bögen verändern ... 559
5.6 Musikfunktionen benutzen ... 560
5.6.1 Syntax der Ersetzungsfunktion ... 560
5.6.2 Beispiele der Ersetzungsfunktionen ... 561

Anhang A Notationsübersicht ... 563

A.1 Liste der Akkordbezeichnungen ... 563
A.2 Übliche Akkord-Variablen ... 564
A.3 Vordefinierte Saitenstimmungen ... 567
A.4 Die vordefinierten Bund-Diagramme ... 568
A.5 MIDI-Instrumente ... 576
A.6 Liste der Farben ... 577
A.7 Die Feta-Schriftart ... 578
Notenschlüssel-Glyphen ... 578
Taktart-Glyphen ... 579
Zahlen-Glyphen ... 579
Versetzungszeichen-Glyphen ... 579
Standard-Notenkopf-Glyphen ... 580
Spezielle Notenkopf-Glyphen ... 581
Geformte Notenkopf-Glyphen ... 581
Pausen-Glyphen ... 585
Fähnchen-Glyphen ... 586
Punkt-Glyphen ... 586
Dynamik-Glyphen ... 586
Schrift-Glyphen ... 587
Pfeilkopf-Glyphen ... 589
Klammerspitzen-Glyphen ... 589
Pedal-Glyphen ... 589
Akkordeon-Glyphen ... 590
Vaticana-Glyphen ... 590
Medicaea-Glyphen ... 591
Hufnagel-Glyphen ... 592

Mensural-Glyphen ... 592
Neomensural-Glyphen ... 595
Petrucci-Glyphen ... 596
Solesmes-Glyphen ... 596
A.8 Notenkopfstile ... 597
A.9 Textbeschriftungsbefehle ... 597
A.9.1 Font ... 597
A.9.2 Align ... 606
A.9.3 Graphic ... 620
A.9.4 Music ... 626
A.9.5 Instrument Specific Markup ... 630
A.9.6 Other ... 633
A.10 Textbeschriftungslistenbefehle ... 639
A.11 Liste der Artikulationszeichen ... 640
Artikulationsskripte ... 640
Ornamentale Skripte ... 640
Fermatenskripte ... 640
Instrumentenspezifische Skripte ... 641
Wiederholungszeichensrkipte ... 641
Ancient scripts ... 641
A.12 Schlagzeugnoten ... 641
A.13 Technisches Glossar ... 643
alist ... 643
callback ... 643
closure ... 643
glyph ... 643
grob ... 643
immutable ... 644
interface ... 644
lexer ... 644
mutable ... 644
output-def ... 644
parser ... 645
parser variable ... 645
prob ... 645
simple closure ... 645
smob ... 646
stencil ... 646
A.14 Alle Kontexteigenschaften ... 646
A.15 Eigenschaften des Layouts ... 656
A.16 Erhältliche Musikfunktionen ... 673
A.17 Vordefinierte Typenprädikate ... 678
R5RS primary predicates ... 678
R5RS secondary predicates ... 678
Guile predicates ... 679
LilyPond scheme predicates ... 679
LilyPond exported predicates ... 679
A.18 Scheme-Funktionen ... 680

Anhang B Befehlsübersicht ... 701

Anhang C LilyPond-Grammatik ... 705

Anhang D GNU Free Documentation License **724**

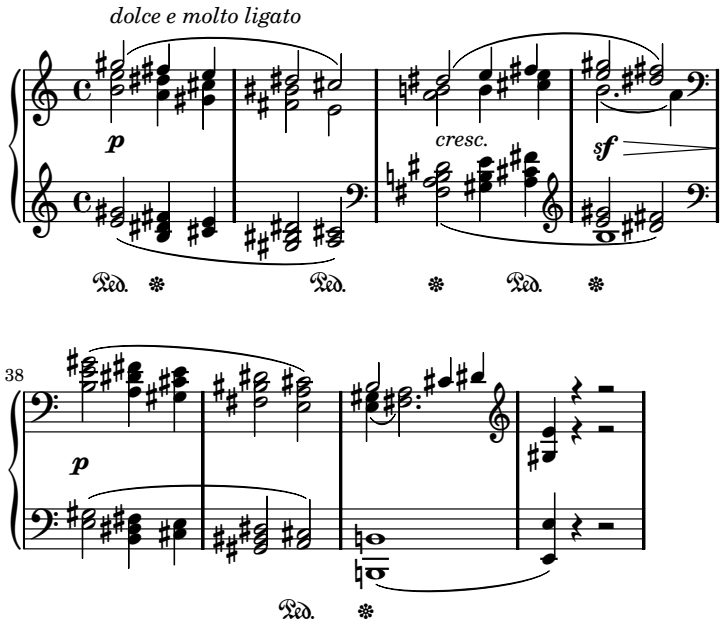
Anhang E Index der LilyPond-Befehle **731**

Anhang F LilyPond-Index **741**

1 Musikalische Notation

Dieses Kapitel erklärt, wie die Notation von Musik erstellt wird.

1.1 Tonhöhen



Dieser Abschnitt zeigt, wie man die Tonhöhe notieren kann. Es gibt drei Stufen in diesem Prozess: Eingabe, Veränderung und Ausgabe.
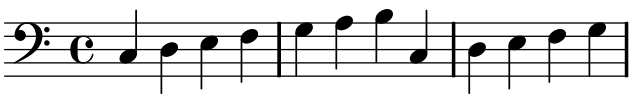
1.1.1 Tonhöhen setzen

Dieser Abschnitt zeigt, wie man Tonhöhen notiert. Es gibt zwei verschiedene Möglichkeiten, Noten in bestimmten Oktaven zu notieren: den absoluten und den relativen Modus. In den meisten Fällen eignet sich der relative Modus besser.

Absolute Oktavenbezeichnung
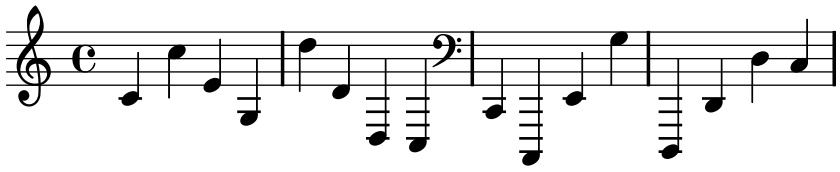
Tonhöhenbezeichnungen werden durch Kleinbuchstaben von `a` bis `g` angegeben. Dabei wird ein aus dem Englischen entlehntes Modell benutzt, das sich vom Deutschen dadurch unterscheidet, dass `b` für die Note „H“ steht. Die Benutzung deutscher Notenbezeichnungen mit der Unterscheidung von b und h ist auch möglich, siehe [Notenbezeichnungen in anderen Sprachen], Seite 7. Die Notenbezeichnungen `c` bis `b` werden in der Oktave unter dem zweigestrichenen C gesetzt.

```
{
  \clef bass
  c4 d e f
  g4 a b c
  d4 e f g
}
```



Andere Oktaven können erreicht werden, indem man ein Apostroph (`'`) oder ein Komma (`,`) benutzt. Jedes `'` erhöht die Tonhöhe um eine Oktave, jedes `,` erniedrigt sie um eine Oktave.

```
{
  \clef treble
  c'4 c'' e' g
  d''4 d' d c
  \clef bass
  c,4 c,, e, g
  d,,4 d, d c
}
```



Siehe auch

Glossar: Abschnitt “Pitch names” in *Glossar*.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Relative Oktavenbezeichnung

Wenn Oktaven im absoluten Modus notiert, passiert es schnell, eine Note auf der falschen Oktave zu notieren. Mit dem relativen Modus kommen solche Fehler seltener vor, weil man die Oktave nur noch sehr selten spezifizieren muss. Hinzu kommt, dass im absoluten Modus ein einzelner Fehler schwer zu finden ist, während er im relativen Modus den ganzen Rest des Stückes um eine Oktave verschiebt.
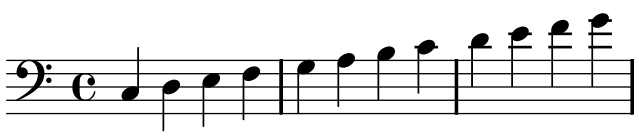
`\relative` ***`Anfangstonhöhe musikalischer Ausdruck`***

Im relativen Modus wird angenommen, dass sich jede folgende Note so dicht wie möglich bei der nächsten befindet. Das bedeutet, dass die Oktave jeder Tonhöhe innerhalb eines *musikalischen Ausdrucks* wie folgt errechnet wird:

- Wenn kein Oktaväänderungszeichen an einer Tonhöhe benutzt wird, wird ihre Oktave so errechnet, dass das Intervall zur vorigen Noten weniger als eine Quinte ist. Das Intervall wird errechnet, ohne Versetzungszeichen zu berücksichtigen.
- Ein Oktaväänderungszeichen `'` oder `,` kann hinzugefügt werden, um eine Tonhöhe explizit um eine Oktave zu erhöhen bzw. zu erniedrigen, relativ zu der Tonhöhe, die ohne das Oktaväänderungszeichen errechnet wurde.
- Mehrfache Oktaväänderungszeichen können benutzt werden. Die Zeichen`''` und `,,` ändern zum Beispiel die Tonhöhe um zwei Oktaven.
- Die Tonhöhe der ersten Note ist relativ zu ***`Anfangstonhöhe`***. Die *Anfangstonhöhe* wird im absoluten Modus gesetzt, und als Empfehlung gilt, eine Oktave von C zu nehmen.
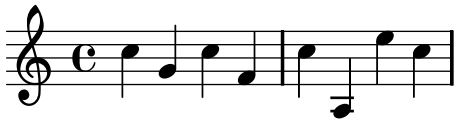
So funktioniert der relative Modus:

```
\relative c {
  \clef bass
  c d e f
  g a b c
  d e f g
}
```
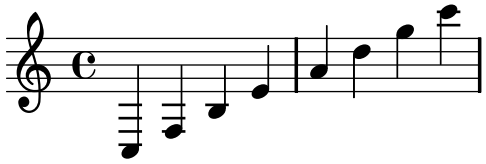


Oktavversetzungen müssen für alle Intervalle angezeigt werden, die größer als eine Quarte sind.

```
\relative c'' {
  c g c f,
  c' a, e'' c
}
```
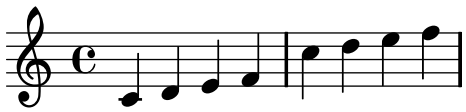


Eine Sequenz ohne ein einziges Oktavänderungszeichen kann aber trotzdem weite Intervalle umfassen:

```
\relative c {
  c f b e
  a d g c
}
```



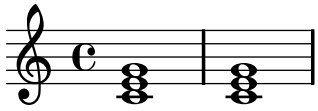
Wenn `\relative`-Umgebungen geschachtelt werden, gilt der innerste `\relative`-Abschnitt.

```
\relative c' {
  c d e f
  \relative c'' {
    c d e f
  }
}
```



`\relative` hat keine Auswirkung auf `\chordmode`-Abschnitte.
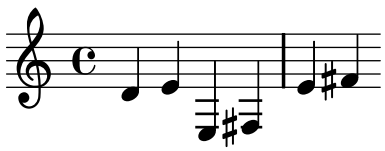
```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```



`\relative` darf nicht innerhalb von `\chordmode` notiert werden.

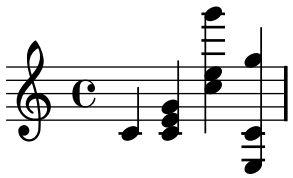
Tonhöhen innerhalb eines `\transpose`-Abschnitts sind absolut, es sei denn ein `\relativ` wird eingefügt.

```
\relative c' {
  d e
  \transpose f g {
    d e
    \relative c' {
      d e
    }
  }
}
```
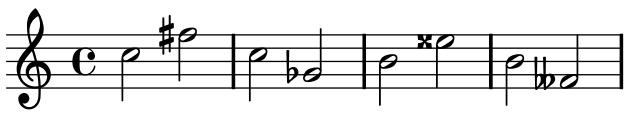


Wenn der vorherige Ausdruck ein Akkord ist, wird die erste Note des Akkordes benutzt, um die erste Note des nächsten Akkordes zu bestimmen. Innerhalb von Akkorden ist die nächste Note immer relativ zur vorherigen. Betrachten Sie das folgende Beispiel aufmerksam, insbesondere die `c`-Noten.

```
\relative c' {
  c
  <c e g>
  <c' e g'>
  <c, e, g''>
}
```



Wie oben erklärt wurde, wird die Oktave einer Tonhöhe nur nach ihrer Notenbezeichnung errechnet, unabhängig von allen Versetzungszeichen. Darum wird ein Eisis auf ein H (notiert als `b`) folgend höher gesetzt, während ein Feses tiefer gesetzt wird. Anders gesagt wird eine doppelterhöhte Quarte wird als kleineres Intervall angesehen als eine doppelterniedrigte Quinte, unabhängig von der Anzahl an Halbtönen, die jedes Intervall enthält.

```
\relative c'' {
  c2 fis
  c2 ges
  b2 eisis
  b2 feses
}
```



Siehe auch

Musikgossar: Abschnitt “fifth” in *Glossar*, Abschnitt “interval” in *Glossar*, Abschnitt “Pitch names” in *Glossar*.

Notationsreferenz: [Oktavenüberprüfung], Seite 9.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “RelativeOctaveMusic” in *Referenz der Interna*.
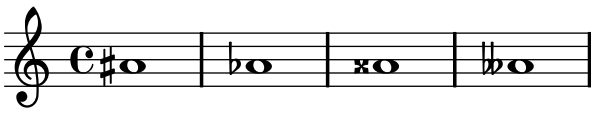
Wenn keine *Anfangstonhöhe* für `\relative` angegeben wird, wird `c'` angenommen. Das ist aber eine veraltete Option, die in späteren Programmversionen verschwinden kann. Darum wird von der Benutzung abgeraten.

Versetzungszeichen

> **Achtung:** Neue Benutzer sind manchmal verwirrt, wie Versetzungszeichen und Vorzeichen/Tonarten funktionieren. In LilyPond sind Notenbezeichnungen die wirkliche Tonhöhe, erst durch Vorzeichen wird bestimmt, wie diese Tonhöhe dann im Notenbild dargestellt wird. Eine einfache Tonhöhe wie etwa `c` bedeutet also immer das eingestrichene C ohne Versetzungszeichen, egal was für Vorzeichen/Tonart oder Schlüssel gesetzt sind. Mehr Information dazu in Abschnitt “Versetzungszeichen und Tonartbezeichnung (Vorzeichen)” in *Handbuch zum Lernen*.

Ein *Kreuz* wird eingegeben, indem man `-is` an die Notenbezeichnung hängt, ein b durch `-es`. Doppelkreuze und Doppel-Bs werden durch Hinzufügen von `-isis` und `-eses` hinter die Notenbezeichnung erzeugt. Diese Syntax leitet sich von den holländischen Notenbezeichnungen ab. Um andere Bezeichnungen für Versetzungszeichen zu benutzen, siehe [Notenbezeichnungen in anderen Sprachen], Seite 7.
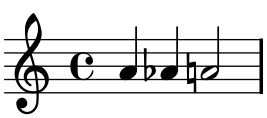
```
ais1 aes aisis aeses
```



Auch die deutschen Varianten `as` für `aes` und `es` für `ees` sind erlaubt. Im Unterschied zum Deutschen ist aber `bes` die einzige Version für den Ton B, während his als `bis` geschrieben werden muss. Das kann aber auch verändert werden, siehe [Notenbezeichnungen in anderen Sprachen], Seite 7.

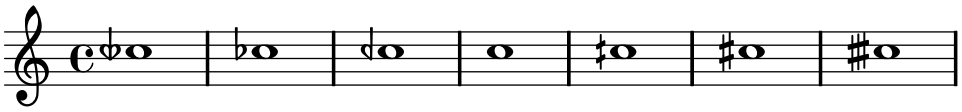
Ein Auflösungszeichen macht die Wirkung eines Kreuzes oder Bs rückgängig. Diese Auflösungszeichen werden jedoch nicht als Suffix einer Tonhöhenbezeichnung eingegeben, sondern sie ergeben sich (automatisch) aus dem Kontext, wenn die nicht alterierte Notenbezeichnung eingegeben wird.

```
a4 aes a2
```
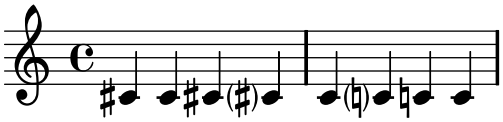


Versetzungszeichen für Vierteltöne werden durch Anhängen der Endungen `-eh` (Erniedrigung) und `-ih` (Erhöhung) an den Tonhöhenbuchstaben erstellt. Das Beispiel zeigt eine in Vierteltönen aufsteigende Serie vom eingestrichenen C.

```
ceseh1 ces ceh c cih cis cisih
```



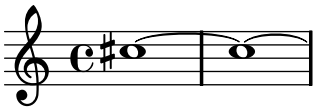
Normalerweise werden Versetzungszeichen automatisch gesetzt, aber sie können auch manuell hinzugefügt werden. Ein erinnerndes Versetzungszeichen kann erzwungen werden, indem man ein Ausrufungszeichen (`!`) hinter die Notenbezeichnung schreibt. Ein warnendes Versetzungszeichen (also ein Vorzeichen in Klammern) wird durch Anfügen eines Fragezeichens (`?`) erstellt. Mit diesen zusätzlichen Zeichen kann man sich auch Auflösungszeichen ausgeben lassen.
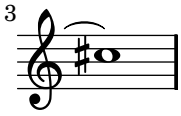
```
cis cis cis! cis? c c? c! c
```



Versetzungzeichen von übergebundenen Noten werden nur dann gesetzt, wenn ein neues System begonnen wird:

```
cis1~ cis~
\break
cis
```



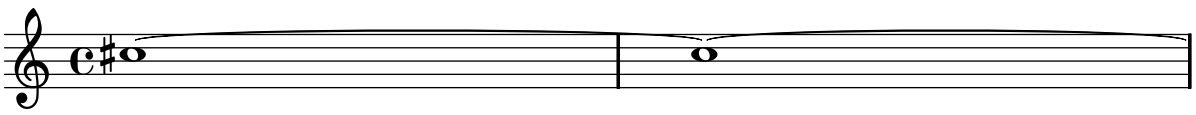


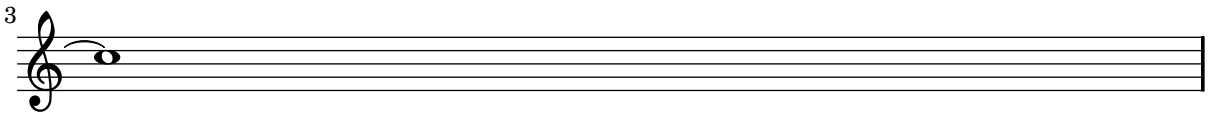
Ausgewählte Schnipsel

Hiding accidentals on tied notes at the beginning of a new system

This shows how to hide accidentals on tied notes at the beginning of a new system.

```
\relative c'' {
  \override Accidental #'hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```
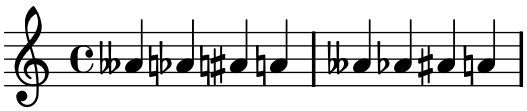




*Verhindern, dass zusätzliche Auflösungszeichen automatisch*hinzugefügt werden

Den traditionellen Notensatzregeln zufolge wird ein Auflösungszeichen immer dann vor einem Kreuz oder B gesetzt, wenn ein vorheriges Versetzungszeichen der gleichen Note aufgehoben werden soll. Um dieses Verhalten zu ändern, muss die Eigenschaft `extraNatural` im `Staff`-Kontext auf "false" gesetzt werden.

```
\relative c'' {
  aeses4 aes ais a
  \set Staff.extraNatural = ##f
  aeses4 aes ais a
}
```



Siehe auch

Glossar: Abschnitt “sharp” in *Glossar*, Abschnitt “flat” in *Glossar*, Abschnitt “double sharp” in *Glossar*, Abschnitt “double flat” in *Glossar*, Abschnitt “Pitch names” in *Glossar*, Abschnitt “quarter tone” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Versetzungszeichen und Tonartbezeichnung (Vorzeichen)” in *Handbuch zum Lernen*.

Notationsreferenz: [Automatische Versetzungszeichen], Seite 24, [Vorgeschlagene Versetzungszeichen (musica ficta)], Seite 401, [Notenbezeichnungen in anderen Sprachen], Seite 7.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “Accidental_engraver” in *Referenz der Interna*, Abschnitt “Accidental” in *Referenz der Interna*, Abschnitt “AccidentalCautionary” in *Referenz der Interna*, Abschnitt “accidental-interface” in *Referenz der Interna*.

Bekannte Probleme und Warnungen
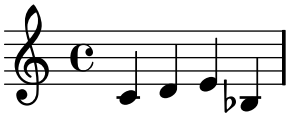
Es gibt keine allgemeinen Regeln für die Notation von Vierteltönen, die Symbole von LilyPond folgen also keinem Standard.

Notenbezeichnungen in anderen Sprachen

Es gibt vordefinierte Bezeichnungen für die Notenbezeichnungen in anderen Sprachen als Englisch. Die Sprache für die Notenbezeichnungen wird normalerweise zu Beginn einer Datei ausgewählt: das folgende Beispiel zeigt die Verwendung von italienischen Notenbezeichnungen:

```
\language "italiano"

\relative do' {
  do re mi sib
}
```



In der Tabelle sind die existierenden Sprachdefinitionen mit den dazugehörigen Notenbezeichnungen dargestellt.
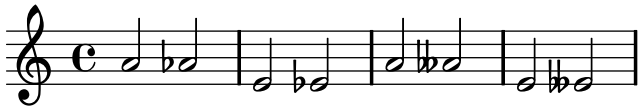
Sprache	Notenbezeichnungen
nederlands	c d e f g a bes b
catalan	do re mi fa sol la sib si
deutsch	c d e f g a b h
english	c d e f g a bf b
espanol	do re mi fa sol la sib si
italiano	do re mi fa sol la sib si
norsk	c d e f g a b h
portugues	do re mi fa sol la sib si
suomi	c d e f g a b h
svenska	c d e f g a b h
vlaams	do re mi fa sol la sib si

und die dazugehörigen Versetzungszeichen-Endungen:

Sprache	Kreuz	B	Doppelkreuz	Doppel-B
nederlands	-is	-es	-isis	-eses
catalan	-d/-s	-b	-dd/-ss	-bb
deutsch	-is	-es	-isis	-eses
english	-s/-sharp	-f/-flat	-ss/-x/-sharpsharp	-ff/-flatflat
espanol	-s	-b	-ss/-x	-bb
italiano	-d	-b	-dd	-bb
norsk	-iss/-is	-ess/-es	-ississ/-isis	-essess/-eses
portugues	-s	-b	-ss	-bb
suomi	-is	-es	-isis	-eses
svenska	-iss	-ess	-ississ	-essess
vlaams	-k	-b	-kk	-bb

Auf Holländisch, Deutsch, Norwegisch und Schwedisch (u. a.) wird die Erniedrigungen von ‚a' – `aes` – zu `as` zusammengezogen. Beide Formen werden jedoch akzeptiert. Genauso gelten auch `es` und `ees`. Das gilt auch für `aeses` / `ases` und `eeses` / `eses`. In manchen Sprachen sind nur diese Kurzformen definiert.

```
a2 as e es a ases e eses
```



Bestimmte Musik verwendet Alterationen, die Bruchteile von den „normalen" Kreuzen oder Bs sind. Die Notenbezeichnungen für Vierteltöne für die verschiedenen Sprachen sind in der folgenden Tabelle aufgeführt. Die Präfixe *semi-* und *sesqui-* bedeuten „halb" bzw. „eineinhalb". Für Sprachen, die nicht in der Tabelle auftauchen, sind noch keine eigenen Namen definiert.

Sprache	Vierteltonkreuz	Viertelton-B	3/4-Tonkreuz	3/4-Ton-B
nederlands	-ih	-eh	-isih	-eseh
deutsch	-ih	-eh	-isih	-eseh
english	-qs	-qf	-tqs	-tqf
espanol	-cs	-cb	-tcs	-tcb
italiano	-sd	-sb	-dsd	-bsb
portugues	-sqt	-bqt	-stqt	-btqt

Die meisten Sprachen, die hier vorkommen, werden normalerweise mit der klassischen westlichen Musik assoziiert. Alternative Tonhöhen und Stimmungen sind aber auch unterstützt: siehe Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418

Siehe auch

Glossar: Abschnitt “Pitch names” in *Glossar*, Abschnitt “Common Practice Period” in *Glossar*.

Notationsreferenz: Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.

Installierte Dateien: ‘`scm/define-note-names.scm`’.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

1.1.2 Viele Tonhöhen gleichzeitig verändern
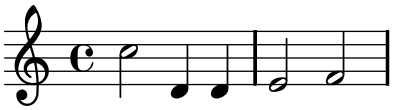
Dieser Abschnitt zeigt, wie man Tonhöhen beeinflusst.

Oktavenüberprüfung

Im relativen Modus geschieht es recht häufig, dass ein Oktaväderungszeichen vergessen wird. Oktavenüberprüfungen machen es einfacher, solche Fehler zu entdecken und zu korrigieren. Sie geben eine Warnung aus und korrigieren die Oktave, wenn eine Note in einer unerwarteten Oktave gefunden wird.
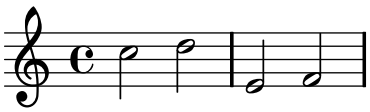
Um die Oktave einer Note zu überprüfen, muss die absolute Oktave nach dem `=`-Symbol angegeben werden. Im folgenden Beispiel wird eine Warnung (und eine Tonhöhenänderung) generiert, weil die zweite Note als absolute Oktave ein `d''` anstelle von `d'` notiert ist, wie es die Oktavierungskorrektur markiert.

```
\relative c'' {
  c2 d='4 d
  e2 f
}
```



Die Oktave von einer Note kann auch mit dem `\octaveCheck` *Kontrolltonhöhe*-Befehl überprüft werden. *Kontrollhöhe* wird im absoluten Modus eingegeben. Dabei wird überprüft, ob das Intervall zwischen der vorherigen Note und der *Kontrolltonhöhe* nicht größer als eine Quarte ist (die normale Berechnung im relativen Modus). Wenn diese Überprüfung einen Fehler ausgibt, wird eine Warnung gemeldet, aber die vorigen Note wird nicht verändert. Folgende Noten sind dann relativ zur *Kontrolltonhöhe*.
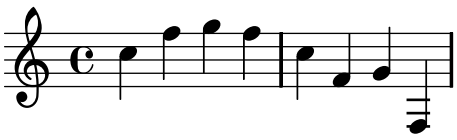
```
\relative c'' {
  c2 d
  \octaveCheck c'
  e2 f
}
```



Vergleichen Sie die zwei Takte im nächsten Beispiel. Die erste und dritte `\octaveCheck`-Überprüfung gibt einen Fehler aus, die zweite dagegen ist erfolgreich:

```
\relative c'' {
  c4 f g f

  c4
  \octaveCheck c'
  f
  \octaveCheck c'
  g
  \octaveCheck c'
  f
}
```



Siehe auch

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “RelativeOctaveCheck” in *Referenz der Interna*.

Transposition

Ein musikalischer Ausdruck kann mit dem Befehl `\transpose` transponiert werden. Die Syntax lautet:
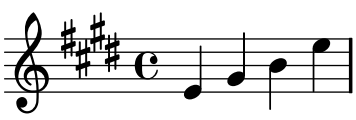
```
\transpose vonTonhöhe nachTonhöhe mus. Ausdruck
```

Das bedeutet, dass der *mus. Ausdruck* um das Intervall zwischen den Tonhöhen *vonTonhöhe* und *nachTonhöhe* transponiert wird: Jede Note, die die Tonhöhe *vonTonhöhe* hat, wird in die Tonhöhe *nachTonhöhe* umgewandelt, und alle anderen Noten um das gleiche Intervall. Beide Tonhöhen werden im absoluten Modus eingegeben.

> **Achtung:** Tonhöhen innerhalb einer `\transpose`-Umgebung sind absolut, es sei denn, ein `\relative` wird eingefügt.
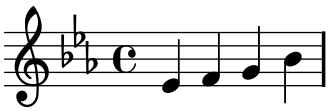
So kann z. B. ein Stück in D-Dur, wenn es für den Sänger etwas zu tief ist, nach E-Dur transponiert werden. Dabei werden auch die Vorzeichen entsprechend angepasst:

```
\transpose d e {
  \relative c' {
    \key d \major
    d4 fis a d
  }
}
```



Wenn eine Stimme, die in C notiert ist, von einer A-Klarinette gespielt werden soll (für die A als C notiert wird, aber eine kleine Terz tiefer erklingt als es notiert ist), kann die entsprechende Stimme wie folgt erstellt werden:
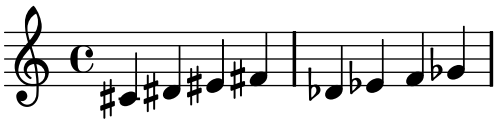
```
\transpose a c' {
  \relative c' {
    \key c \major
    c4 d e g
  }
}
```



Beachten Sie, dass `\key c \major` explizit angegeben werden muss. Wenn hier keine Tonart angemerkt würde, würde die Noten zwar transponiert, aber keine Vorzeichen angezeigt werden.

`\transpose` unterscheidet enharmonische Verwechslungen: sowohl `\transpose c cis` als auch `\transpose c des` transponieren die Musik einen Halbton nach oben. Aber die erste Version gibt als Versetzungszeichen Kreuze aus, die zweite dagegen B-Versetzungszeichen.

```
music = \relative c' { c d e f }
\new Staff {
  \transpose c cis { \music }
  \transpose c des { \music }
}
```



`\transpose` kann auch benutzt werden, um die geschriebenen Noten eines transponierenden Instruments zu notieren. Im vorigen Beispiel wurde die Tonhöhen so eingegeben, wie sie erklingen (also in C), aber man kann genauso gut auch andersherum aus einer Stimme, die für ein transponierendes Instrument in einem anderen Ton als C geschrieben wurde, eine Partitur in C erstellen. Die Noten einer B-Trompete, die mit einem notierten E (also einem klingenden D) anfangen, könnte man also auch so eingeben:

```
musicInBflat = { e4 ... }
\transpose c bes, \musicInBflat
```

Um die Noten dann in F zu setzen (um sie etwa für ein Horn zu arrangieren), könnte man die schon geschriebenen Noten wieder mit einem weiteren `\transpose` umgeben:

```
musicInBflat = { e4 ... }
\transpose f c' { \transpose c bes, \musicInBflat }
```

Für mehr Information zu transponierenden Instrumenten siehe auch [Transposition von Instrumenten], Seite 22.

Ausgewählte Schnipsel

Noten mit minimaler Anzahl an Versetzungszeichen transponieren.

Dieses Beispiel benutzt Scheme-Code, um enharmonische Verwechslungen für Noten zu erzwingen, damit nur eine minimale Anzahl an Versetzungszeichen ausgegeben wird. In diesem Fall gelten die folgenden Regeln:

- Doppelte Versetzungszeichen sollen entfernt werden
- His -> C
- Eis -> F

- Ces -> B
- Fes -> E

Auf diese Art werden am meisten natürliche Tonhöhen als enharmonische Variante gewählt.

```
#(define (naturalize-pitch p)
   (let ((o (ly:pitch-octave p))
         (a (* 4 (ly:pitch-alteration p)))
         ;; alteration, a, in quarter tone steps,
         ;; for historical reasons
         (n (ly:pitch-notename p)))
     (cond
      ((and (> a 1) (or (eq? n 6) (eq? n 2)))
       (set! a (- a 2))
       (set! n (+ n 1)))
      ((and (< a -1) (or (eq? n 0) (eq? n 3)))
       (set! a (+ a 2))
       (set! n (- n 1))))
     (cond
      ((> a 2) (set! a (- a 4)) (set! n (+ n 1)))
      ((< a -2) (set! a (+ a 4)) (set! n (- n 1))))
     (if (< n 0) (begin (set! o (- o 1)) (set! n (+ n 7))))
     (if (> n 6) (begin (set! o (+ o 1)) (set! n (- n 7))))
     (ly:make-pitch o n (/ a 4))))
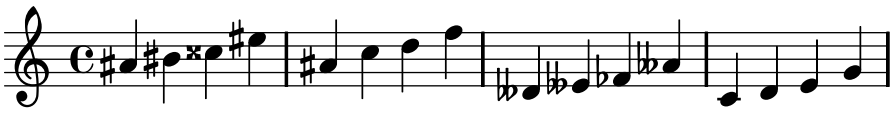
#(define (naturalize music)
   (let ((es (ly:music-property music 'elements))
         (e (ly:music-property music 'element))
         (p (ly:music-property music 'pitch)))
     (if (pair? es)
         (ly:music-set-property!
          music 'elements
          (map (lambda (x) (naturalize x)) es)))
     (if (ly:music? e)
         (ly:music-set-property!
          music 'element
          (naturalize e)))
     (if (ly:pitch? p)
         (begin
           (set! p (naturalize-pitch p))
           (ly:music-set-property! music 'pitch p)))
     music))

naturalizeMusic =
#(define-music-function (parser location m)
   (ly:music?)
   (naturalize m))

music = \relative c' { c4 d e g }

\score {
  \new Staff {
    \transpose c ais { \music }
```

```
    \naturalizeMusic \transpose c ais { \music }
    \transpose c deses { \music }
    \naturalizeMusic \transpose c deses { \music }
  }
  \layout { }
}
```



Siehe auch

Notationsreferenz: [Relative Oktavenbezeichnung], Seite 2, [Transposition von Instrumenten], Seite 22. [Umkehrung], Seite 13 [Krebs], Seite 13 [Modale Transformierungen], Seite 14.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “TransposedMusic” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Der relative Modus wirkt nicht in `\transpose`, `\chordmode` oder `\relative`. Um auch im relativen Modus transponieren zu können, muss ein `\relative` innerhalb des `\tranpose` zusätzlich gesetzt werden.
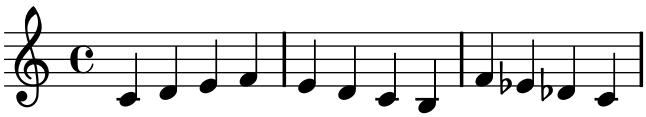
Umkehrung

Ein musikalischer Ausdruck kann in seine Umkehrung transformiert werden:

`\inversion` *`vonTonhöhe nachTonhöhe mus. Ausdruck`*

Der *`mus. Ausdruck`* wird Intervall für Intervall umgekehrt und dann von v*`vonTonhöhe`* nach *`nachTonhöhe`* transponiert.

```
music = \relative c' { c d e f }
\new Staff {
  \music
  \inversion d' d' \music
  \inversion d' ees' \music
}
```



Siehe auch

Notationsreferenz: ⟨undefined⟩ [Modale Transformationen], Seite ⟨undefined⟩, [Krebs], Seite 13, [Transposition], Seite 10.
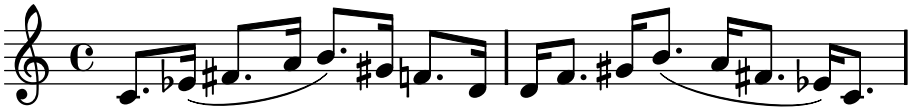
Krebs

Ein musikalischer Ausdruck kann umgekehrt werden, um im Krebsgang dargestellt zu werden:

```
music = \relative c' { c8. ees16( fis8. a16 b8.) gis16 f8. d16 }

\new Staff {
```

```
  \music
  \retrograde \music
}
```



Bekannte Probleme und Warnungen

Manuell gesetzte Bindebögen innerhalb des Krebsgangs `\retrograde` werden aufgebrochen und erzeugen Warnungen. Einige Bindebögen können automatisch ersetllt werden, indem man ⟨undefined⟩ [Automatic note splitting], Seite ⟨undefined⟩ aktiviert.

Siehe auch

Notationsreferenz: ⟨undefined⟩ [Modale Transformationen], Seite ⟨undefined⟩, ⟨undefined⟩ [Umgekehrung], Seite ⟨undefined⟩, [Transposition], Seite 10.

Modale Transformierungen

In einer Komposition, die auf einer Skala basiert, wird ein Motiv oft in unterschiedlicher Weise transformiert. Es kann transponiert werden, um von einer anderen Tonhöhe in der Skala zu beginnen, oder beginnend von einer bestimmten Tonhöhe der Skala umgekehrt werden. Es kann auch im Krebsgang, also rückwärts erscheinen, siehe [Krebs], Seite 13.

> **Achtung:** Alle Noten, die nicht zu der vorgegebenen Skala gehören, werden nicht transformiert.

Modale Transposition
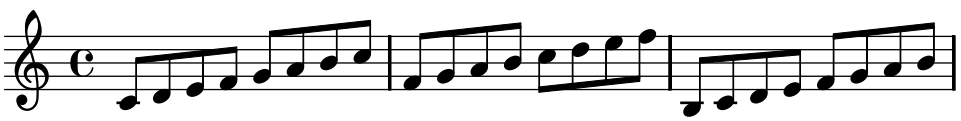
Ein Motiv kann innerhalb einer Skala transponiert werden:

`\modalTranspose` *`vonTonhöhe nachTonhöhe Skala mus. Ausdruck`*

Die Noten von *`mus. Ausdruck`* werden innerhalb von *`Skala`* um die Anzahl von Skalenstufen transponiert, die dem Intervall zwischen *`vonTonhöhe`* und *`nachTonhöhe`* entspricht.

```
diatonicScale = \relative c' { c d e f g a b }
motif = \relative c' { c8 d e f g a b c }

\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
```
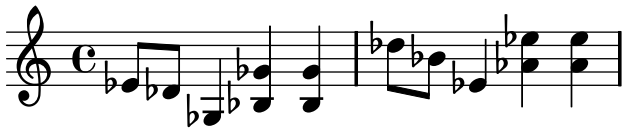


Eine absteigende Skala beliebiger Länge und mit beliebigen Intervallen kann definiert werden:

```
pentatonicScale = \relative c' { ges aes bes des ees }
motif = \relative c' { ees8 des ges,4 <ges' bes,> <ges bes,> }

\new Staff {
```
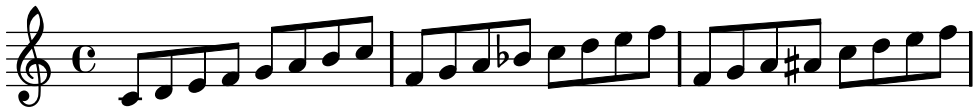
```
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
```



Wenn `\modalTranspose` mit einer chromatischen Skala benutzt wird, entspricht es der Funktion von `\transpose` mit dem Unterschied, dass die Bezeichnungen der eingesetzten Noten definiert werden können:

```
chromaticScale = \relative c' { c cis d dis e f fis g gis a ais b }
motif = \relative c' { c8 d e f g a b c }

\new Staff {
  \motif
  \transpose c f \motif
  \modalTranspose c f \chromaticScale \motif
}
```



Modale Umkehrung

Ein musikalischer Ausdruck kann innerhalb einer definierten Skala um eine Referenznote umgekehrt werden und transponiert werden:
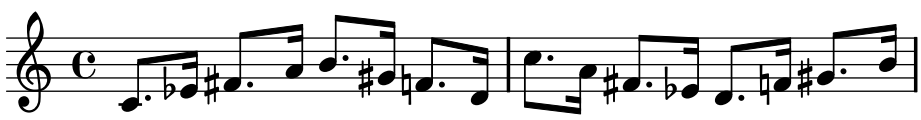
`\modalInversion` *`Umkehrungstonhöhe nachTonhöhe Skala mus. Ausdruck`*

Die Noten von *`mus. Ausdruck`* werden mit den gleichen Intervallabständen voneinander gesetzt, ausgehend von *`Umkehrungstonhöhe`*, allerdings in die entgegengesetzte Richtung. Das Result wird dann innerhalb der *`Skala`* um das Intervall transponiert, das sich aus dem Unterschied von *`Umkehrungstonhöhe`* und *`nachTonhöhe`* ergibt.

Um also einfach eeine Umkehrung ohne zusätzliche Transposition zu machen, sollten *`Umkehrungstonhöhe`* und *`nachTonhöhe`* gleich sein.

```
octatonicScale = \relative c' { ees f fis gis a b c d }
motif = \relative c' { c8. ees16 fis8. a16 b8. gis16 f8. d16 }
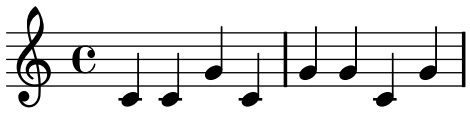
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
```



Um den Drehpunkt der Umkehrung zwischen zwei Noten der Skala zu definieren, wird um eine der Noten die Umkehrung vorgenommen und dann eine Skalenstufe transponiert. Die beiden Noten klammern sozusagen den Drehpunkt ein.

```
scale = \relative c' { c g' }
motive = \relative c' { c c g' c, }
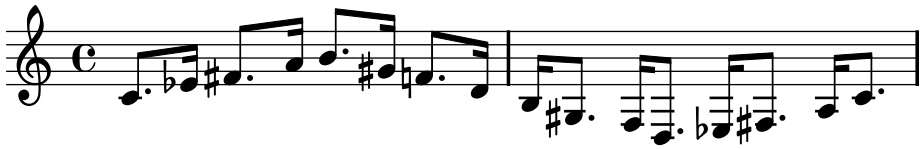
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
```



Die kombinierte Operation von Umkehrung und Krebs erzeugt die Krebsumkehrung:

```
octatonicScale = \relative c' { ees f fis gis a b c d }
motif = \relative c' { c8. ees16 fis8. a16 b8. gis16 f8. d16 }

\new Staff {
  \motif
  \retrograde \modalInversion c' c' \octatonicScale \motif
}
```



Siehe auch

Notationsreferenz: [Umkehrung], Seite 13, [Krebs], Seite 13, [Transposition], Seite 10.
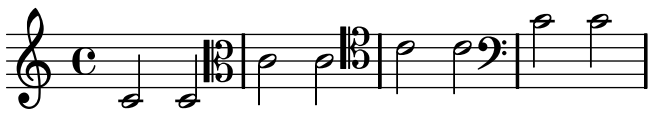
1.1.3 Tonhöhen anzeigen lassen

Dieser Abschnitt zeigt, wie die Ausgabe von Tonhöhen verändern werden kann.

Notenschlüssel

Der Schlüssel kann verändert werden. Das eingestrichene C wird in jedem Beispiel gezeigt:

```
\clef treble
c2 c
\clef alto
c2 c
\clef tenor
c2 c
\clef bass
c2 c
```



Andere Schlüssel sind u. A.:

```
\clef french
c2 c
```
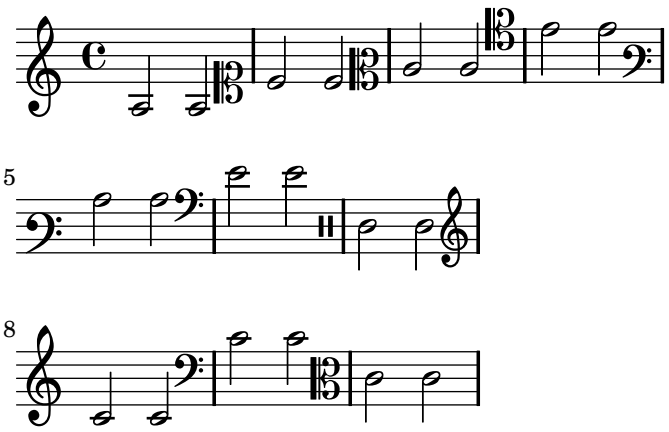
```
\clef soprano
c2 c
\clef mezzosoprano
c2 c
\clef baritone
c2 c

\break

\clef varbaritone
c2 c
\clef subbass
c2 c
\clef percussion
c2 c

\break
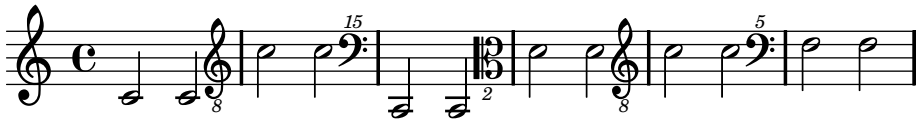
\clef G   % synonym for treble
c2 c
\clef F   % synonym for bass
c2 c
\clef C   % synonym for alto
c2 c
```



Indem `_8` oder `^8` an die jeweilige Schlüsselbezeichnung angehängt wird, wird der Schlüssel um eine Oktave nach oben oder unten transponiert, mit `_15` oder `^15` um zwei Oktaven. Auch andere Ganzzahlen können verwendet werden, wenn es gewünscht wird. Die Schlüsselbezeichnung muss in Anführungszeichen gesetzt werden, wenn nicht-alphabetische Zeichen enthält, siehe Beispiel:

```
\clef treble
c2 c
\clef "treble_8"
c2 c
\clef "bass^15"
c2 c
\clef "alto_2"
c2 c
\clef "G_8"
c2 c
```

```
\clef "F^5"
c2 c
```



Weitere unterstützte Schlüssel sind beschrieben in [Mensurale Schlüssel], Seite 396, [Gregorianische Schlüssel], Seite 404, [Standardtabulaturen], Seite 305 und [Angepasste Tabulaturen], Seite 317.

Ausgewählte Schnipsel

Eigenschaften des Schlüssels optimieren

Der Befehl `\clef "treble_8"` ist gleichbedeutend mit einem expliziten Setzen der Eigenschaften von `clefGlyph`, `clefPosition` (welche die vertikale Position des Schlüssels bestimmt), `middleCPosition` und `clefOctavation`. Ein Schlüssel wird ausgegeben, wenn eine der Eigenschaften außer `middleCPosition` sich ändert.

Eine Änderung des Schriftzeichens (Glyph), der Schlüsselposition oder der Oktavierung selber ändert noch nicht die Position der darauf folgenden Noten auf dem System: das geschieht nur, wenn auch die Position des eingestrichenen C (middleCPosition) angegeben wird. Die Positionsparameter sind relativ zur Mittellinie des Systems, dabei versetzen positive Zahlen die Position nach oben, jeweils eine Zahl für jede Linie plus Zwischenraum. Der `clefOctavation`-Wert ist normalerweise auf 7, -7, 15 oder -15 gesetzt, aber auch andere Werte sind gültig.

Wenn ein Schlüsselwechsel an einem Zeilenwechsel geschieht, wird das neue Symbol sowohl am Ende der alten Zeilen als auch am Anfang der neuen Zeile ausgegeben. Wenn der Warnungs-Schlüssel am Ende der alten Zeile nicht erforderlich ist, kann er unterdrückt werden, indem die `explicitClefVisibility`-Eigenschaft des `Staff`-Kontextes auf den Wert `end-of-line-invisible` gesetzt wird. Das Standardverhalten kann mit `\unset Staff.explicitClefVisibility` wieder hergestellt werden.

Die folgenden Beispiele zeigen die Möglichkeiten, wenn man diese Eigenschaften manuell setzt. Auf der ersten Zeile erhalten die manuellen Änderungen die ursprüngliche relative Positionierung von Schlüssel und Noten, auf der zweiten Zeile nicht.

```
\layout { ragged-right = ##t }

{
  % The default treble clef
  c'1
  % The standard bass clef
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  \set Staff.middleCPosition = #6
  c'1
  % The baritone clef
  \set Staff.clefGlyph = #"clefs.C"
  \set Staff.clefPosition = #4
  \set Staff.middleCPosition = #4
  c'1
  % The standard choral tenor clef
  \set Staff.clefGlyph = #"clefs.G"
  \set Staff.clefPosition = #-2
```

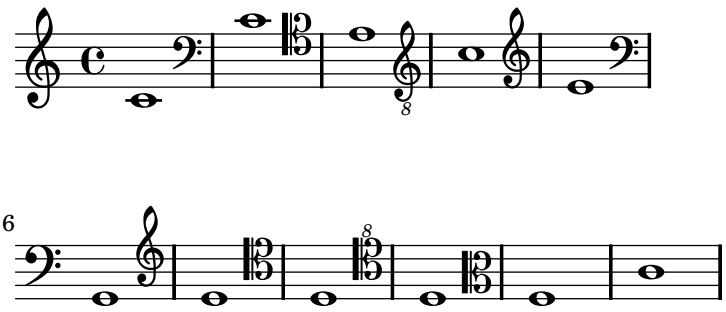
```
  \set Staff.clefOctavation = #-7
  \set Staff.middleCPosition = #1
  c'1
  % A non-standard clef
  \set Staff.clefPosition = #0
  \set Staff.clefOctavation = #0
  \set Staff.middleCPosition = #-4
  c'1 \break

  % The following clef changes do not preserve
  % the normal relationship between notes and clefs:

  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  c'1
  \set Staff.clefGlyph = #"clefs.G"
  c'1
  \set Staff.clefGlyph = #"clefs.C"
  c'1
  \set Staff.clefOctavation = #7
  c'1
  \set Staff.clefOctavation = #0
  \set Staff.clefPosition = #0
  c'1

  % Return to the normal clef:

  \set Staff.middleCPosition = #0
  c'1
}
```



Siehe auch

Notationsreferenz: [Mensurale Schlüssel], Seite 396, [Gregorianische Schlüssel], Seite 404 [Standardtabulaturen], Seite 305, [Angepasste Tabulaturen], Seite 317.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “Clef_engraver” in *Referenz der Interna*, Abschnitt “Clef” in *Referenz der Interna*, Abschnitt “OctavateEight” in *Referenz der Interna*, Abschnitt “clef-interface” in *Referenz der Interna*.

Tonartbezeichnung

> **Achtung:** Neue Benutzer sind manchmal verwirrt, wie Versetzungszeichen und Vorzeichen/Tonarten funktionieren. In LilyPond sind Notenbezeichnungen die wirkliche Tonhöhe, erst durch Vorzeichen wird bestimmt, wie diese Tonhöhe dann im Notenbild dargestellt wird. Eine einfache Tonhöhe wie etwa `c` bedeutet also immer das eingestrichene C ohne Versetzungszeichen, egal was für Vorzeichen/Tonart oder Schlüssel gesetzt sind. Mehr Information dazu in Abschnitt "Versetzungszeichen und Tonartbezeichnung (Vorzeichen)" in *Handbuch zum Lernen*.
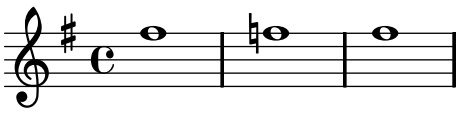
Die Vorzeichen zeigen die Tonart an, in welcher ein Stück notiert ist. Es handelt sich um eine Anzahl von Alterationszeichen (Kreuzen oder Bs) am Beginn jedes Notensystems.

Die Tonart kann geändert werden:

```
\key Tonhöhe Modus
```

Der Wert *Modus* sollte entweder `\major` oder `\minor` sein, um Moll oder Dur der *Tonhöhe* zu erhalten. Es können auch Modusbezeichnungen für Kirchentonarten verwendet werden: `\ionian` (Ionisch), `\locrian` (Lokrisch), `\aeolian` (Aeolisch), `\mixolydian` (Mixolydisch), `\lydian` (Lydisch), `\phrygian` (Phrygisch) und `\dorian` (Dorisch).

```
\key g \major
fis1
f
fis
```
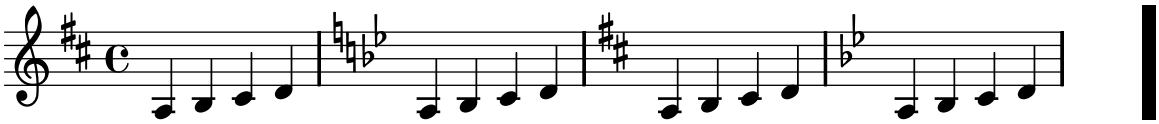


Ausgewählte Schnipsel

Auflösungzeichen nicht setzen wenn die Tonart wechselt

Wenn die Tonart wechselt, werden automatisch Auflösungszeichen ausgegeben, um Versetzungszeichen der vorherigen Tonart aufzulösen. Das kann verhindert werden, indem die `printKeyCancellation`-Eigenschaft im `Staff`-Kontext auf "false" gesetzt wird.

```
\relative c' {
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
  \set Staff.printKeyCancellation = ##f
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
}
```



Untypische Tonarten
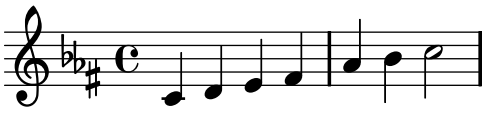
Der üblicherweise benutzte `\key`-Befehl setzt die `keySignature`-Eigenschaft im `Staff`-Kontext.

Um untypische Tonartenvorzeichen zu erstellen, muss man diese Eigenschaft direkt setzen. Das Format für den Befehl ist eine Liste: `\set Staff.keySignature = #`(((Oktave . Schritt) . Alteration) ((Oktave . Schritt) . Alteration) ...)` wobei für jedes Element in der Liste `Oktave` die Oktave angibt (0 ist die Oktave vom eingestrichenen C bis zum eingestrichenen H), `Schritt` gibt die Note innerhalb der Oktave an (0 heißt C und 6 heißt H), und `Alteration` ist `,SHARP` `,FLAT` `,DOUBLE-SHARP` usw. (Beachte das beginnende Komma.)

Alternativ kann auch jedes Element der Liste mit dem allgemeineren Format `(Schritt . Alteration)` gesetzt werden, wobei dann die Einstellungen für alle Oktaven gelten.

Hier ein Beispiel einer möglichen Tonart für eine Ganztonleiter:

```
\relative c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
                              ((0 . 5) . ,FLAT)
                              ((0 . 3) . ,SHARP))
  c4 d e fis
  aes4 bes c2
}
```



Siehe auch

Glossar: Abschnitt “church mode” in *Glossar*, Abschnitt “scordatura” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Versetzungszeichen und Tonartbezeichnung (Vorzeichen)” in *Handbuch zum Lernen*.

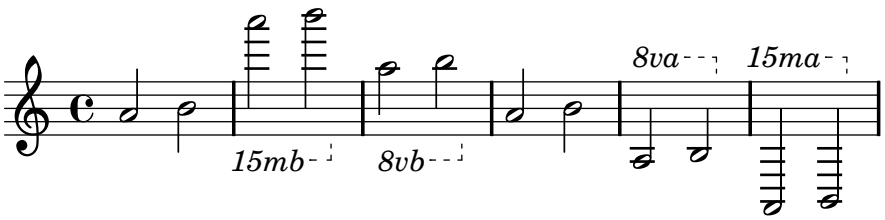
Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “KeyChangeEvent” in *Referenz der Interna*, Abschnitt “Key_engraver” in *Referenz der Interna*, Abschnitt “Key_performer” in *Referenz der Interna*, Abschnitt “KeyCancellation” in *Referenz der Interna*, Abschnitt “KeySignature” in *Referenz der Interna*, Abschnitt “key-cancellation-interface” in *Referenz der Interna*, Abschnitt “key-signature-interface” in *Referenz der Interna*.

Oktavierungsklammern

Oktavierungsklammern zeigen eine zusätzliche Transposition von einer Oktave an:

```
a2 b
\ottava #-2
a2 b
\ottava #-1
a2 b
\ottava #0
a2 b
\ottava #1
a2 b
\ottava #2
a2 b
```

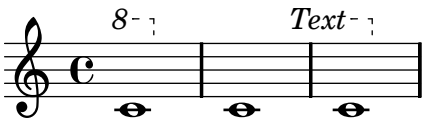


Ausgewählte Schnipsel

Ottava-Text

Intern setzt die `set-octavation`-Funktion die Eigenschaften `ottavation` (etwa auf den Wert `"8va"` oder `"8vb"`) und `middleCPosition`. Um den Text der Oktavierungsklammer zu ändern, kann `ottavation` manuell gesetzt werden, nachdem `set-octavation` benützt wurde.

```
{
  \ottava #1
  \set Staff.ottavation = #"8"
  c''1
  \ottava #0
  c'1
  \ottava #1
  \set Staff.ottavation = #"Text"
  c''1
}
```



Siehe auch

Glossar: Abschnitt “octavation” in *Glossar*.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “Ottava_spanner_engraver” in *Referenz der Interna*, Abschnitt “OttavaBracket” in *Referenz der Interna*, Abschnitt “ottava-bracket-interface” in *Referenz der Interna*.

Transposition von Instrumenten

Wenn man Noten setzt, die von transponierenden Instrumenten gespielt werden, sind oft einige Stimmen auf einer anderen Tonhöhe notiert als dem Kammerton. In diesem Fall muss die Tonart des transponierenden Instruments gekennzeichnet werden, weil sonst die MIDI-Ausgabe und Stichnoten in anderen Stimmen falsche Tonhöhen produzieren. Mehr Information zu Stichnoten in [Stichnoten], Seite 181.

`\transposition` *Tonhöhe*

Die Tonhöhe, die für `\transposition` benutzt wird, muss mit dem wirklichen Ton übereinstimmen, der erklingt, wenn das Instrument ein `c'` in seiner Stimme spielt. Die Tonhöhe wird im absoluten Modus angegeben, ein Instrument also, dass einen Ton höher erklingt als es notiert wird, muss folgenden Befehl benutzen: `\transposition d'`. `\transposition` sollte *nur* dann benutzt werden, wenn sie nicht *nicht* in C notiert werden.
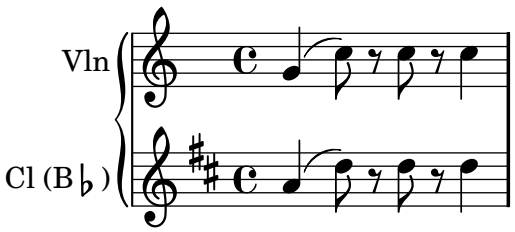
Hier einige Noten für Geige und B-Klarinette: die Stimmen (Noten und Vorzeichen) sind so notiert, wie sie in der Partitur erscheinen. Die zwei Instrumente spielen unisono.

```
\new GrandStaff <<
  \new Staff = "violin" {
```

```
    \relative c'' {
      \set Staff.instrumentName = #"Vln"
      \set Staff.midiInstrument = #"violin"
      % not strictly necessary, but a good reminder
      \transposition c'

      \key c \major
      g4( c8) r c r c4
    }
  }
  \new Staff = "clarinet" {
    \relative c'' {
      \set Staff.instrumentName = \markup { Cl (B\flat) }
      \set Staff.midiInstrument = #"clarinet"
      \transposition bes

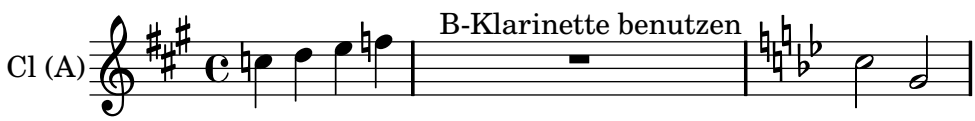
      \key d \major
      a4( d8) r d r d4
    }
  }
>>
```



Die `\transposition` kann während eines Stückes geändert werden. Ein Klarinettist zum Beispiel kann zwischen B- und A-Klarinette wechseln.

```
\set Staff.instrumentName = #"Cl (A)"
\key a \major
\transposition a
c d e f
\textLengthOn
s1*0^\markup { B-Klarinette benutzen }
R1

\key bes \major
\transposition bes
c2 g
```



Siehe auch

Glossar: Abschnitt “concert pitch” in *Glossar*, Abschnitt “transposing instrument” in *Glossar*.

Notationsreferenz: [Stichnoten], Seite 181, [Transposition], Seite 10.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Automatische Versetzungszeichen

Es gibt viele unterschiedliche Regeln, wie Versetzungszeichen notiert werden. LilyPond hat eine Funktion, mit der spezifiziert werden kann, welcher Stil benutzt werden soll. Diese Funktion kann man wie folgt benutzen:

```
\new Staff <<
  #(set-accidental-style 'voice)
  { ... }
>>
```

Der Versetzungszeichenstil bezieht sich auf das aktuelle Notensystem in der Standardeinstellung (eine Ausnahme bilden die Stile `piano` und `piano-cautionary`, die weiter unten erklärt werden). Die Funktion kann aber auch ein zweites Argument erhalten, mit der spezifiziert wird, auf welchen Bereich sich der neue Stil erstreckt. Um etwa den neuen Stil in allen Systemen einer Stimmgruppe (`StaffGroup`) zu benutzen, müsste der Befehl so aussehen:

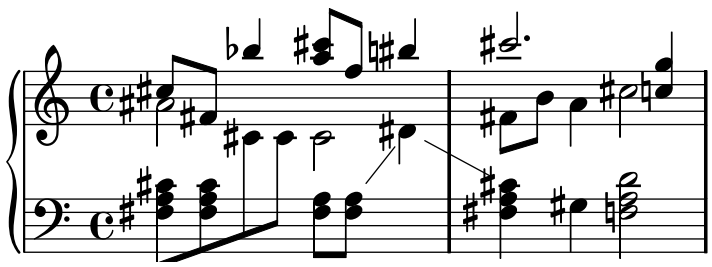
```
#(set-accidental-style 'voice 'StaffGroup)
```

Folgende Versetzungszeichenstile sind unterstützt. Um jeden Stil zu erklären, wird folgendes Beispiel benützt:

```
musicA = {
  <<
    \relative c' {
      cis'8 fis, bes'4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \\
    \relative c' {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative c' {
      <fis, a cis>8 <fis a cis>
      \change Staff = up
      cis' cis
      \change Staff = down
      <fis, a> <fis a>
      \showStaffSwitch
      \change Staff = up
      dis'4 |
      \change Staff = down
      <fis, a cis>4 gis <f a d>2 |
    }
  }
}
```

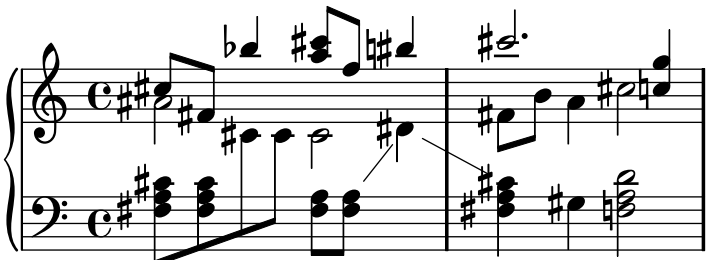
```
\new PianoStaff {
  <<
    \context Staff = "up" {
      #(set-accidental-style 'default)
      \musicA
    }
    \context Staff = "down" {
      #(set-accidental-style 'default)
      \musicB
    }
  >>
}
```



Die letzten Zeilen des Beispiels könnten auch mit folgendem Code ersetzt werden, solange der gleiche Versetzungszeichenstil in beiden Systemen benutzt werden soll:

```
\new PianoStaff {
  <<
    \context Staff = "up" {
      %%% change the next line as desired:
      #(set-accidental-style 'default 'Score)
      \musicA
    }
    \context Staff = "down" {
      \musicB
    }
  >>
}
```

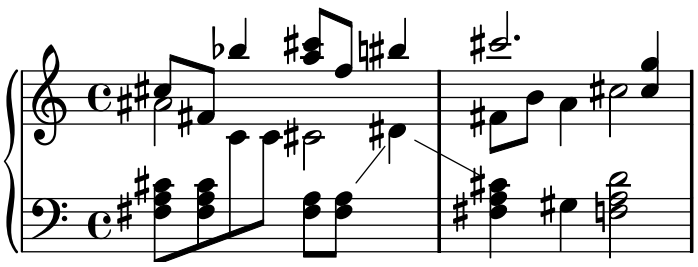
`default (Standard)`
: Das ist das Standardverhalten. Es entspricht der Konvention für Notation von Musik des 18. Jahrhunderts: Versetzungszeichen werden bis zum Taktende erinnert, in dem sie gesetzt wurden, und nur in ihrer eigenen Oktave. Im nächsten Beispiel wird also kein Auslösungszeichen vor dem `b` (H) im zweiten Takt oder dem letzen `c` gesetzt:



`voice (Stimme)`

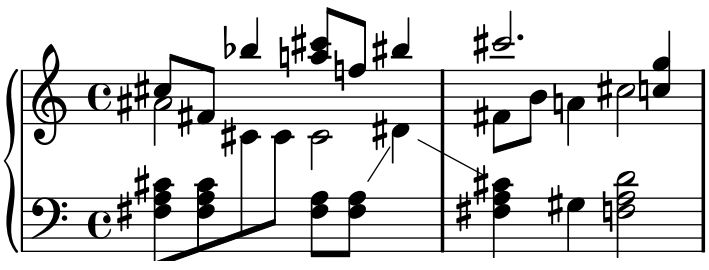
Das normale Verhalten ist es, die Versetzungszeichen auf der Notensystemebene zu erinnern. In diesem Stil aber werden Versetzungszeichen individuell für jede Stimme errechnet. Abgesehen davon gelten die Regeln des Standardstiles (`default`).

Das hat zur Folge, dass Versetzungszeichen von einer Stimme in der anderen nicht aufgelöst werden, was oft ein unerwünschtes Ergebnis ist: im folgenden Beispiel kann man schwer sagen, ob das zweite `a` unalteriert oder erhöht gespielt werden soll. Die `voice`-Option sollte also nur benutzt werden, wenn die Stimmen separat von unterschiedlichen Musikern gelesen werden. Wenn das System nur von einem Musiker benutzt wird (etwa der Dirigent oder ein Klavierspieler), dann sind die Stile `modern` oder `modern-cautionary` besser.



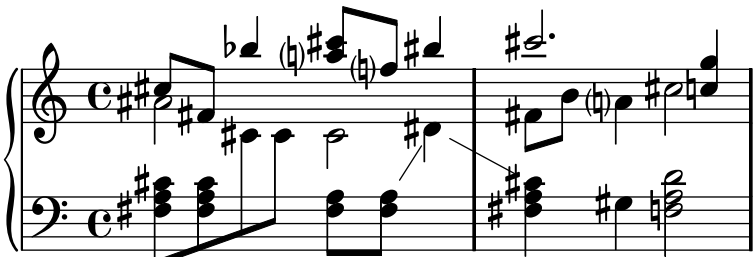
`modern (Modern)`

Dieser Stil orientiert sich an den üblichen Regeln für das 20. Jahrhundert. Hierbei werden einige zusätzliche Auflösungszeichen ausgelassen, die traditionell für Versetzungszeichen benutzt wurden, welche die Richtung des vorherigen Versetzungszeichens verringern oder umkehren. Die gleichen Versetzungszeichen wie im Standardstil werden gesetzt, allerdings mit zwei Ausnahmen, die Uneindeutigkeiten verhindern sollen: nach vorübergehenden Versetzungszeichen werden Auflösungszeichen auch im folgenden Takt gesetzt (für Noten innerhalb der selben Oktave) und im gleichen Takt für Noten in unterschiedlichen Oktaven. Daher kommen also die Auflösungszeichen vor dem H und dem C im zweiten Takt des oberen Systems:



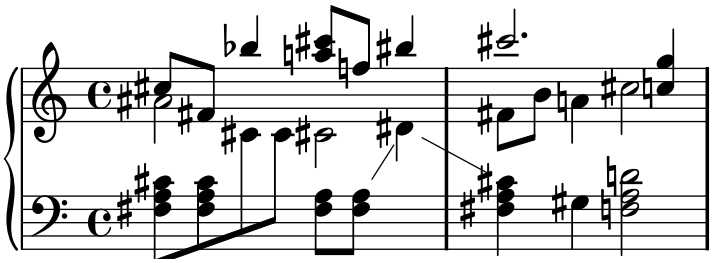
`modern-cautionary (Modern mit Warnungen)`

Dieser Stil ähnelt `modern`, aber die „zusätzlichen“ Versetzungszeichen (die normalerweise nicht gesetzt werden) werden als Warnungen gesetzt. In der Standardeinstellung werden sie in Klammern gesetzt, aber sie können auch in kleinerer Größe gesetzt werden, wenn man die `cautionary-style`-Eigenschaft von `AccidentalSuggestion` definiert.



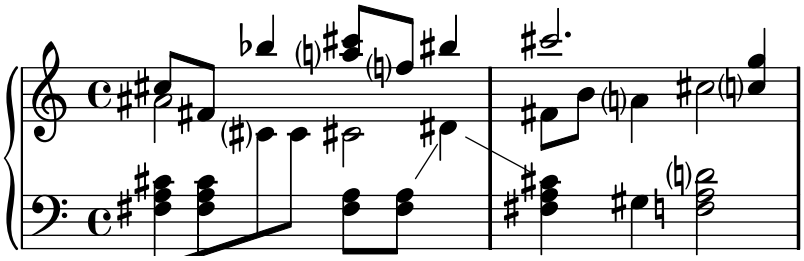
`modern-voice (Modern für Stimmen)`

Diese Regel wird für vielstimmige Noten benutzt, die sowohl von unterschiedlichen Spielern für jede Stimme als auch von einem Spieler für alle Stimmen benutzt. Versetzungszeichen werden für jede Stimme gesetzt, aber sie *werden* über die Stimme hinweg aufgelöst innerhalb des selben Notensystems. Das `a` im letzten Takt ist also aufgelöst, weil die vorigen Auflösung in einer anderen Stimme stattgefunden hatte, und das `d` im unteren System ist aufgelöst wegen eines Versetzungszeichens in einer anderen Stimme im vorigen Takt:



`modern-voice-cautionary (modern mit Warnungen für einzelne Stimmen)`

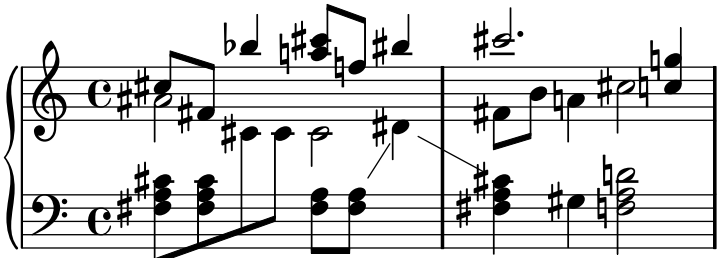
Dieser Stil ist der gleiche wie `modern-voice`, nur dass hier die zusätzlichen Versetzungszeichen (die nicht vom `voice`-Stil gesetzt werden) als Warnungsversetzungszeichen gesetzt werden. Obwohl alle Versetzungszeichen, die mit `default` gesetzt werden, auch mit diesem Stil gesetzt werden, sind manche Warnungsversetzungszeichen.



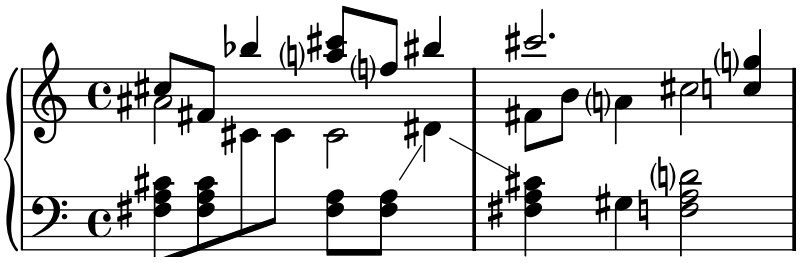
`piano (Klavier)`

Dieser Stil orientiert sich an den Regeln im 20. Jahrhundert für die Notation von Klaviermusik. Er ist sehr ähnlich mit dem modernen Stil, aber Versetzungszeichen werden auch über Notensysteme hinweg für die selbe Akkolade (`GrandStaff` oder `PianoStaff`) aufgelöst.

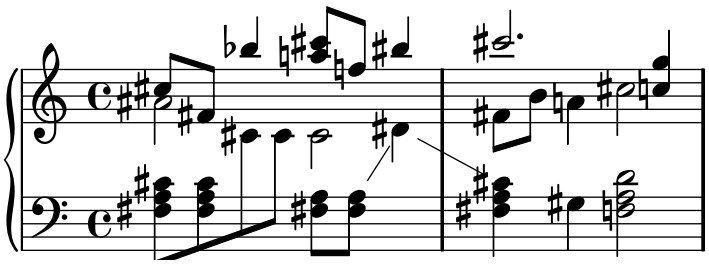
Dieser Versetzungszeichenstil wirkt sich standardmäßig auf die gesamte Akkolade (`GrandStaff` oder `PianoStaff`) aus.



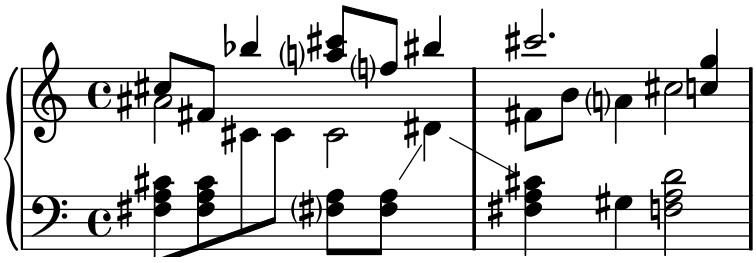
`piano-cautionary (Klavier mit Warnungen)`
: Dieser Stil verhält sich wie `piano`, aber die zusätzlichen Versetzungszeichen werden als Warnungen ausgegeben:



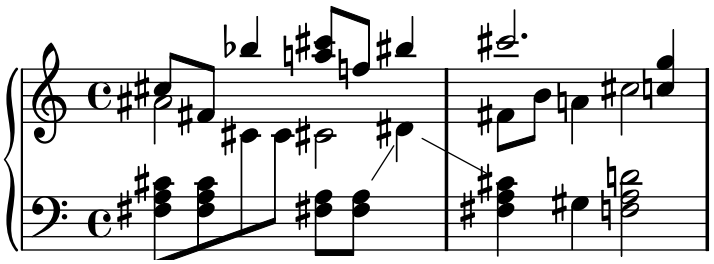
`neo-modern`
: Dieser Stil richtet sich nach den Regeln für moderne Musik: Versetzungszeichen werden mit im `modern`-Stil gesetzt, aber sie werden nochmal gesetzt, wenn die gleiche Note später im selben Takt auftritt – außer die Note wird unmittelbar wiederholt.



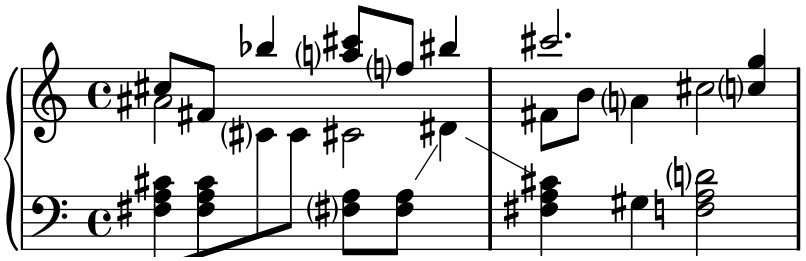
`neo-modern-cautionary (neo-modern mit Warnungen)`
: Dieser Stil ähnelt `neo-modern`, aber die zusätzlichen Versetzungszeichen werden als Warnungen gesetzt.



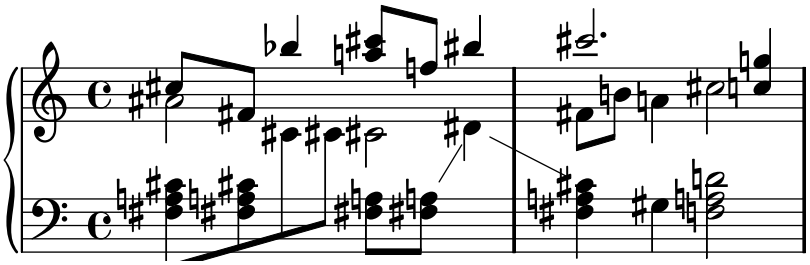
`neo-modern-voice (neo-modern für Stimmen)`
: Diese Regel wird für für Versetzungszeichen in mehreren Stimmen eingesetzt, wenn die Noten sowohl von Musikern gelesen werden, die nur eine Stimme lesen, als auch von Musikern, die alle Stimmen lesen. Versetzungszeichen werden für jede Stimme so wie mit der `neo-modern`-Regel gesetzt, aber innerhalb des gesamten Notensystems mit Auflösungszeichen versehen.



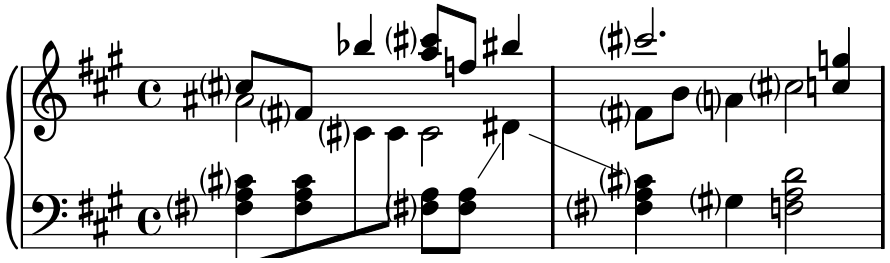
`neo-modern-voice-cautionary`
: Diese Regel ähnelt `neo-modern-voice`, aber die zusätzlichen Versetzungszeichen werden hier als warnende Versetzungszeichen gesetzt.



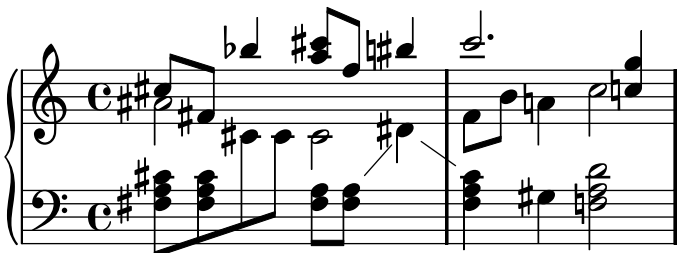
`dodecaphonic (Zwölftonmusik)`
: Dieser Stil orientiert sich an der Notation von sog. Zwölftonmusik, der Stil wurde Anfang des 20. Jahrhunderts in Gebrauch genommen. In diesem Stil erhält *jede* Note ein Versetzungszeichen, wozu auch Auflösungszeichen zählen.
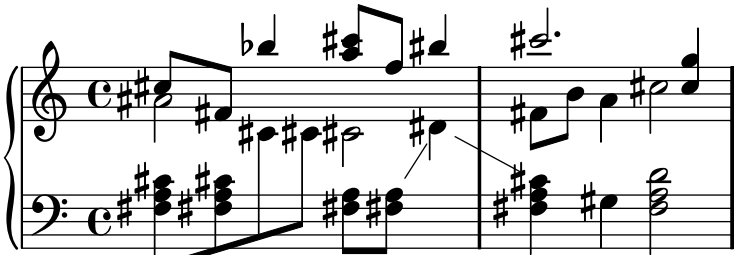


`teaching (didaktisch)`
: Dieser Stil ist für Lernende bestimmt: der Stil orientiert sich am `modern`-Stil, aber die Alterationen, die sich durch die Tonart ergeben, werden zusätzlich als Warnungsversetzungszeichen gesetzt. Eine Ausnahme sind direkt wiederholte Noten.



`no-reset (nicht zurücksetzen)`
: Das ist der gleiche Stil wie `default`, aber die Versetzungszeichen dauern für „immer“ an, nicht nur im aktuellen Takt:



`forget (vergessen)`
: Das ist das Gegenteil von `no-reset`: Versetzungszeichen werden überhaupt nicht erinnert und folgerichtig werden alle Versetzungszeichen entsprechend der Tonart gesetzt, unabhängig vom Kontext der Noten.



Siehe auch

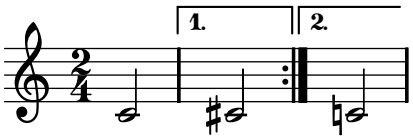
Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “Accidental” in *Referenz der Interna*, Abschnitt “Accidental_engraver” in *Referenz der Interna*, Abschnitt “GrandStaff” in *Referenz der Interna*, Abschnitt “PianoStaff” in *Referenz der Interna*, Abschnitt “Staff” in *Referenz der Interna*, Abschnitt “AccidentalSuggestion” in *Referenz der Interna*, Abschnitt “AccidentalPlacement” in *Referenz der Interna*, Abschnitt “accidental-suggestion-interface” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Gleichzeitig erklingende Noten werden bei der automatischen Bestimmung der Versetzungszeichen nicht berücksichtigt: nur die vorige Note und die Vorzeichen werden einbezogen. Man muss die Versetzungszeichen mit `!` oder `?` schreiben, wenn gleichzeitig unterschiedliche Alterationen vorkommen, wie etwa für ‘`<f! fis!>`’.
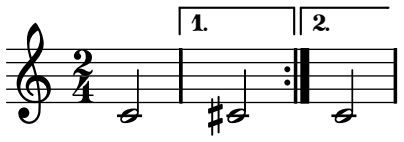
Die warndenden Auflösungzeichen werden gesetzt, indem die vorangegangenen Takte betrachtet werden. In der zweiten oder einer weiteren Wiederholungsklammer erwartet man jedoch, dass die Auflösungszeichen sich aus dem letzten *gespielten* und nicht dem letzten *gesetzten* Takt ergeben. Im folgenden Beispiel bräuchte das `c` in der zweiten Klammer kein Auflösungszeichen:



Die folgende Notlösung kann benutzt werden: Man definiert eine Funktion, die den Versetzungszeichenstil kurzzeitig auf `forget` umschaltet:

```
forget = #(define-music-function (parser location music) (ly:music?) #{
  #(set-accidental-style 'forget)
  $music
  #(set-accidental-style 'modern)
#})
{
  #(set-accidental-style 'modern)
  \time 2/4
  \repeat volta 2 {
    c'2
  }
  \alternative {
```

```
      cis'
      \forget c'
  }
}
```
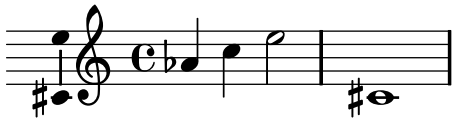


Tonumfang

Der Begriff *ambitus* (Pl. ambitus) beschreibt den Stimmumfang einer Stimme. Er kann auch die Töne bedeuten, die ein Musikinstrument zu spielen in der Lage ist. Ambitus werden in Chorpartituren gesetzt, damit die Sänger schnell wissen, ob sie die Stimme meistern können.

Ambitus werden zu Beginn des Stückes nahe des ersten Schlüssels notiert. Der Stimmumfang wird durch zwei Notenköpfe dargestellt, die die tiefste und höchste Note der Stimme repräsentieren. Versetzungszeichen werden nur gesetzt, wenn sie nicht durch die Tonart definiert werden.

```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

\relative c'' {
  aes c e2
  cis,1
}
```
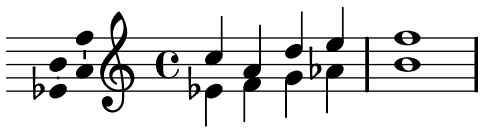


Ausgewählte Schnipsel

Ambitus pro Stimme hinzufügen

Ambitus können pro Stimme gesetzt werden. In diesem Fall müssen sie manuell verschoben werden, um Zusammenstöße zu verhindern.

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus #'X-offset = #2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
```
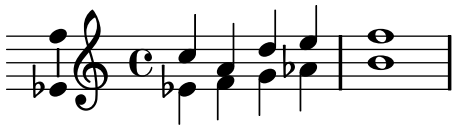
```
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Ambitus mit vielen Stimmen

Indem man den `Ambitus_engraver` im `Staff`-Kontext hinzufügt, erhält man einen einzigen Ambitus pro System, auch in dem Fall, dass mehrere Stimmen sich im gleichen System befinden.

```
\new Staff \with {
  \consists "Ambitus_engraver"
  }
<<
  \new Voice \relative c'' {
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Changing the ambitus gap

It is possible to change the default gap setting for ambitus.

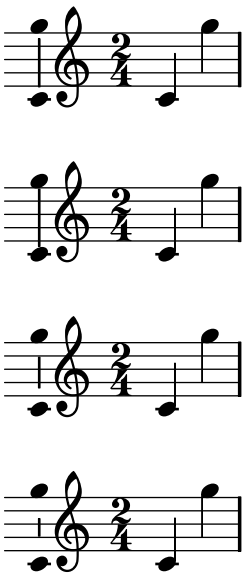
```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

\new Staff {
  \time 2/4
  % Default setting
  c'4 g''
}
```

```
\new Staff {
  \time 2/4
  \override AmbitusLine #'gap = #0
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine #'gap = #1
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine #'gap = #1.5
  c'4 g''
}
```



Siehe auch

Glossar: Abschnitt “ambitus” in *Glossar*.

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “Ambitus_engraver” in *Referenz der Interna*, Abschnitt “Voice” in *Referenz der Interna*, Abschnitt “Staff” in *Referenz der Interna*, Abschnitt “Ambitus” in *Referenz der Interna*, Abschnitt “AmbitusAccidental” in *Referenz der Interna*, Abschnitt “AmbitusLine” in *Referenz der Interna*, Abschnitt “AmbitusNoteHead” in *Referenz der Interna*, Abschnitt “ambitus-interface” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Es gibt keine Kollisionskontrolle bei mehreren Ambitus in einem System.
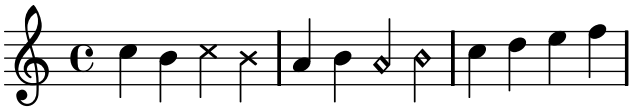
1.1.4 Notenköpfe

Dieser Abschnitt zeigt, wie man Notenköpfe ändern kann.

Besondere Notenköpfe

Notenköpfe können verändert werden:

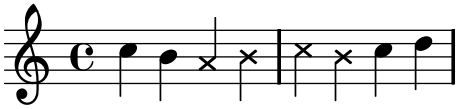
```
c4 b
\override NoteHead #'style = #'cross
c4 b
\revert NoteHead #'style
a b
\override NoteHead #'style = #'harmonic
a b
\revert NoteHead #'style
c4 d e f
```



Für alle Notenkopfstile siehe Abschnitt A.8 [Notenkopfstile], Seite 597

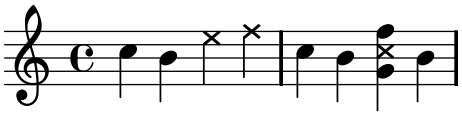
Der Kreuz-(`cross`) Stil wird mit unterschiedlichen musikalischen Absichten eingesetzt. Die folgenden vordefinierten Befehle verändern die Notenköpfe sowohl in Notensystemen als auch in Tabulaturen und können benutzt werden, um alle musikalischen Bedeutungen zu notieren:

```
c4 b
\xNotesOn
 a b c4 b
\xNotesOff
c4 d
```



Die Form als musikalische Funktion dieses Befehls kann innerhalb und außerhalb von Akkorden benutzt werden, um Notenköpfe mit Kreuzen in normalen und Tabulatursystemen zu erstellen:
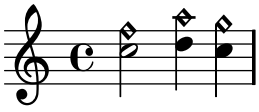
```
c4 b
\xNote { e f }
c b < g \xNote c f > b
```



Als Synonym für `\xNote`, `\xNotesOn` und `\xNotesOff` kann `\deadNote`, `\deadNotesOn` und `\deadNotesOff` benutzt werden. Der Begriff *dean note* (engl. tote Note) wird regelmäßig von Gitaristen benutzt.

Es gibt auch einen Kurzbefehl für die Rautenform, der nur innerhalb von Akkorden benutzt werden kann:

```
<c f\harmonic>2 <d a'\harmonic>4 <c g'\harmonic>
```



Vordefinierte Befehle

`\harmonic`, `\xNotesOn`, `\xNotesOff`, `\xNote`.

Siehe auch

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

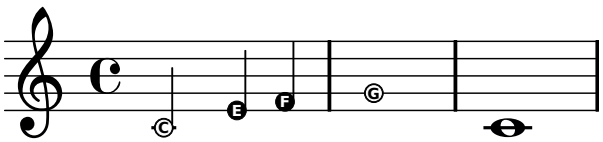
Notationsreferenz: Abschnitt A.8 [Notenkopfstile], Seite 597, [Noten mit Akkorden], Seite 142 [Flageolett und gedämpfte Noten], Seite 345.

Referenz der Interna: Abschnitt “note-event” in *Referenz der Interna*, Abschnitt “Note_heads_engraver” in *Referenz der Interna*, Abschnitt “Ledger_line_engraver” in *Referenz der Interna*, Abschnitt “NoteHead” in *Referenz der Interna*, Abschnitt “LedgerLineSpanner” in *Referenz der Interna*, Abschnitt “note-head-interface” in *Referenz der Interna*, Abschnitt “ledger-line-spanner-interface” in *Referenz der Interna*.

Easy-Notation-Notenköpfe

Die „einfachen Notenköpfe“ haben die Bezeichnung der Note im Kopf gedruckt. Das wird eingesetzt, um die Notation beizubringen. Damit die Buchstaben noch lesbar sind, müssen sie sehr groß gesetzt werden. Wie man eine größere Schriftart einstellt, findet sich in Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472.

```
#(set-global-staff-size 26)
\relative c' {
  \easyHeadsOn
  c2 e4 f
  g1
  \easyHeadsOff
  c,1
}
```



Vordefinierte Befehle

`\easyHeadsOn`, `\easyHeadsOff`.

Ausgewählte Schnipsel

Numbers as easy note heads

Easy notation note heads use the `note-names` property of the `NoteHead` object to determine what appears inside the note head. By overriding this property, it is possible to print numbers representing the scale-degree.

A simple engraver can be created to do this for every note head object it sees.

```
#(define Ez_numbers_engraver
   (list
    (cons 'acknowledgers
          (list
           (cons 'note-head-interface
                 (lambda (engraver grob source-engraver)
                   (let* ((context (ly:translator-context engraver))
                          (tonic-pitch (ly:context-property context 'tonic))
```

```
                                (tonic-name (ly:pitch-notename tonic-pitch))
                                (grob-pitch
                                 (ly:event-property (event-cause grob) 'pitch))
                                (grob-name (ly:pitch-notename grob-pitch))
                                (delta (modulo (- grob-name tonic-name) 7))
                                (note-names
                                 (make-vector 7 (number->string (1+ delta)))))
                           (ly:grob-set-property! grob 'note-names note-names)))))))

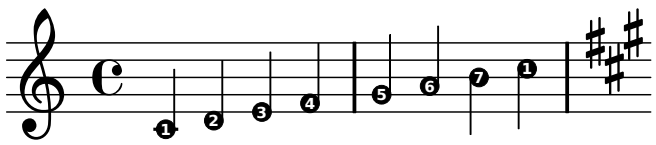
#(set-global-staff-size 26)

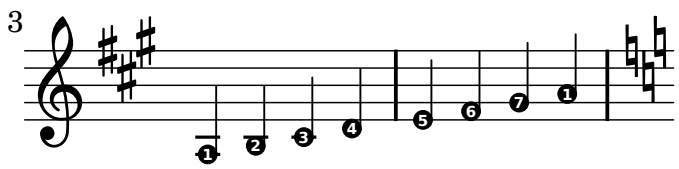
\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

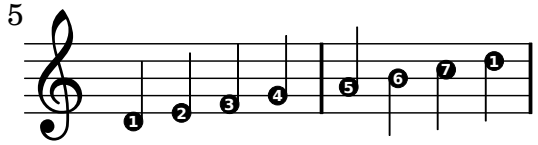
\relative c' {
  \easyHeadsOn
  c4 d e f
  g4 a b c \break

  \key a \major
  a,4 b cis d
  e4 fis gis a \break

  \key d \dorian
  d,4 e f g
  a4 b c d
}
```







Siehe auch

Notationsreferenz: Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472.

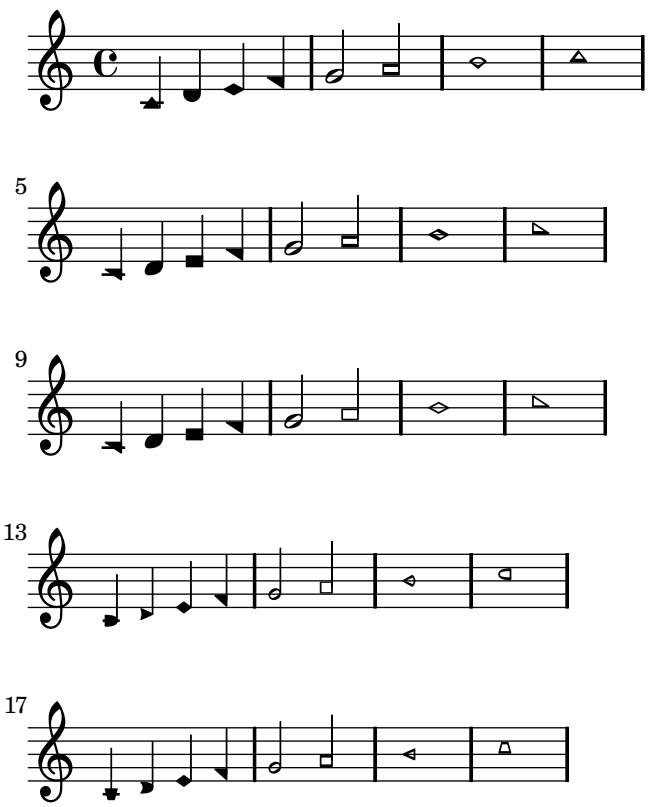
Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “note-event” in *Referenz der Interna*, Abschnitt “Note_heads_engraver” in *Referenz der Interna*, Abschnitt “NoteHead” in *Referenz der Interna*, Abschnitt “note-head-interface” in *Referenz der Interna*.

Notenköpfe mit besonderen Formen
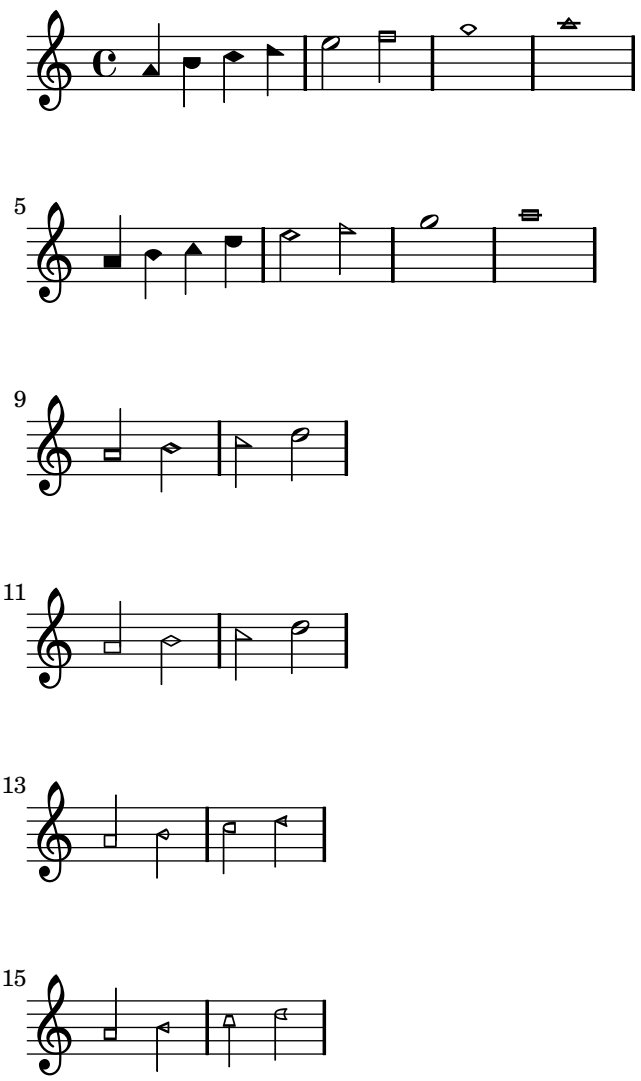
In dieser Notation haben die Notenköpfe eine Form, die ihrer harmonischen Funktion innerhalb der Tonleiter entspricht. Die Notation war sehr beliebt in amerikanischen Liederbüchern des 19. Jahrhunderts. Auf diese Weise können die Formen Sacred Harp, Southern Harmony, Funk (Harmonica Sacra), Walker und Aiken (Christian Harmony) benutzt werden:

```
\aikenHeads
c, d e f g2 a b1 c \break
\sacredHarpHeads
c,4 d e f g2 a b1 c \break
\southernHarmonyHeads
c,4 d e f g2 a b1 c \break
\funkHeads
c,4 d e f g2 a b1 c \break
\walkerHeads
c,4 d e f g2 a b1 c \break
```



Die unterschiedlichen Formen richten sich nach der Stufe in der Skala, wobei der Grundton der Skala aus dem `\key`-Befehl entnommen wird. Wenn man eine Moll-Skala benutzt, ergibt sich die Form der Notenköpfe aus der parallelen Dur-Tonleiter:

```
\key a \minor
\aikenHeads
a b c d e2 f g1 a \break
\aikenHeadsMinor
a,4 b c d e2 f g1 a \break
\sacredHarpHeadsMinor
a,2 b c d \break
\southernHarmonyHeadsMinor
a2 b c d \break
\funkHeadsMinor
a2 b c d \break
\walkerHeadsMinor
a2 b c d \break
```



Vordefinierte Befehle

`\aikenHeads`, `\aikenHeadsMinor`, `\funkHeads`, `\funkHeadsMinor`, `\sacredHarpHeads`, `\sacredHarpHeadsMinor`, `\southernHarmonyHeads`, `\southernHarmonyHeadsMinor`, `\walkerHeads`, `\walkerHeadsMinor`.

Ausgewählte Schnipsel

Notenkopfstile basierend auf der Tonleiterstufe erstellen

Die `shapeNoteStyles`-(NotenFormenStile)-Eigenschaft kann benutzt werden, um verschiedene Notenstile für jeden Schritt der Tonleiter zu definieren (vorgegeben von der Tonart oder der „tonic“ (Tonika)-Eigenschaft. Diese Eigenschaft braucht eine Anzahl von Symbolen, welche beliebig sein können (geometrische Ausdrücke wie `triangle` (Dreieck), `cross` (Kreuz) und `xcircle` (X-Kreis) sind erlaubt) oder basierend auf einer alten amerikanischen Notensatztradition (einige lateinische Notenbezeichnungen sind auch erlaubt).

Um alte amerikanische Liederbücher zu imitieren, gibt es einige vordefinierte Notenstile wie etwa `\aikenHeads` (im Stil von Aiken) oder `\sacredHarpHeads` (im Stil der Sacred Harp-Tradition).

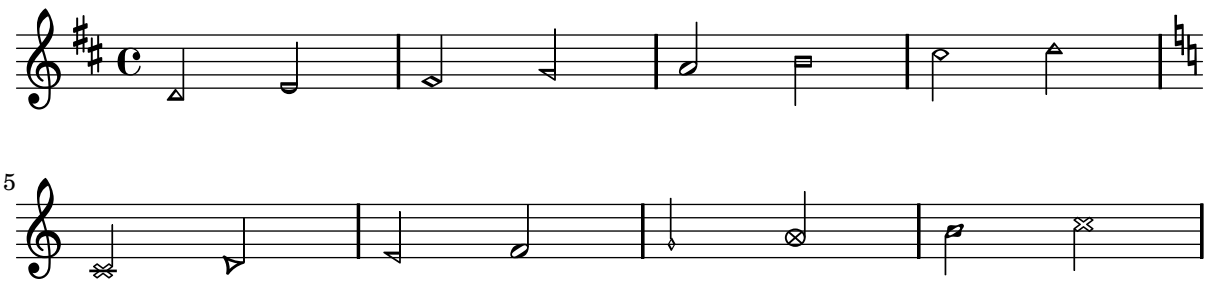
Dieses Beispiel zeigt, wie man unterschiedlich geformte Noten erhält und eine Melodie transponieren kann, ohne dass das Verhältnis zwischen den harmonischen Funktionen und dem Notenstil verloren geht.

```
fragment = {
  \key c \major
  c2 d
  e2 f
  g2 a
  b2 c
}

\new Staff {
  \transpose c d
  \relative c' {
    \set shapeNoteStyles = #'#(do re mi fa
                               #f la ti)
    \fragment
  }

  \break

  \relative c' {
    \set shapeNoteStyles = #'#(cross triangle fa #f
                               mensural xcircle diamond)
    \fragment
  }
}
```



Alle Notenkopfstile finden sich in Abschnitt A.8 [Notenkopfstile], Seite 597.

Siehe auch

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

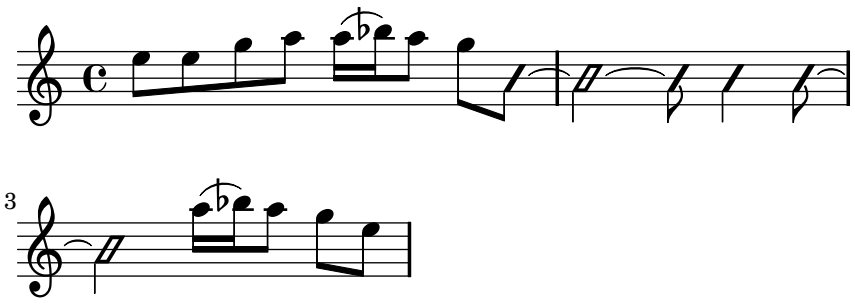
Notationsreferenz: Abschnitt A.8 [Notenkopfstile], Seite 597.

Referenz der Interna: Abschnitt “note-event” in *Referenz der Interna*, Abschnitt “Note_heads_engraver” in *Referenz der Interna*, Abschnitt “NoteHead” in *Referenz der Interna*, Abschnitt “note-head-interface” in *Referenz der Interna*.

Improvisation

Improvisation wird manchmal angezeigt, indem schräge Notenköpfe gesetzt werden, wenn der Spieler eine beliebige Tonhöhe wählen kann aber den vorgegebenen Rhythmus spielen soll. Sie können wie folgt benutzt werden:

```
\new Voice \with {
  \consists "Pitch_squash_engraver"
} {
  e8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  e2 ~ e8 f4 f8 ~
  f2
  \improvisationOff
  a16( bes) a8 g e
}
```



Vordefinierte Befehle

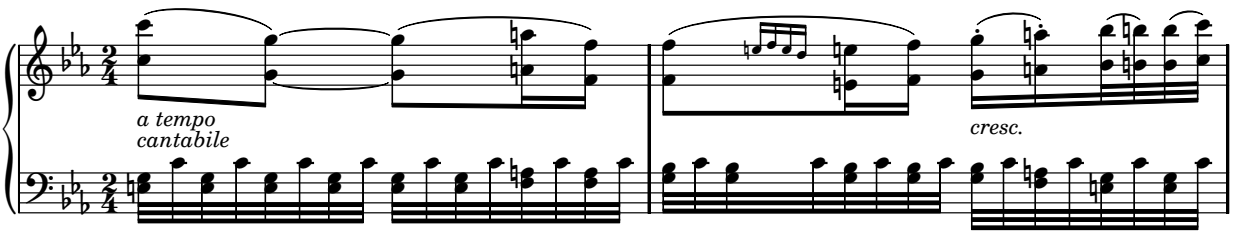
`\improvisationOn`, `\improvisationOff`.

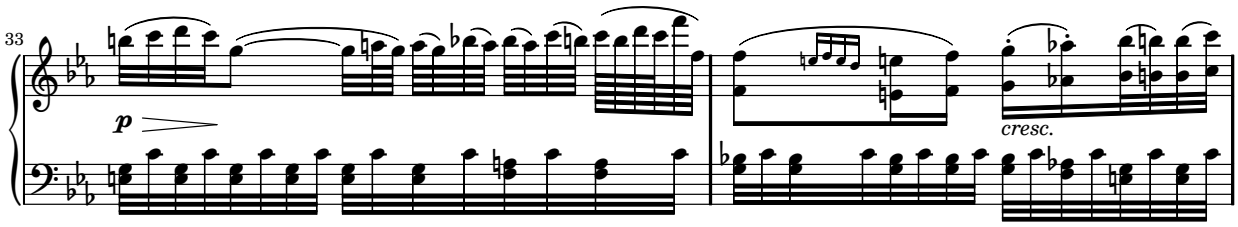
Siehe auch

Schnipsel: Abschnitt “Pitches” in *Schnipsel*.

Referenz der Interna: Abschnitt “Pitch_squash_engraver” in *Referenz der Interna*, Abschnitt “Voice” in *Referenz der Interna*, Abschnitt “RhythmicStaff” in *Referenz der Interna*.

1.2 Rhythmus
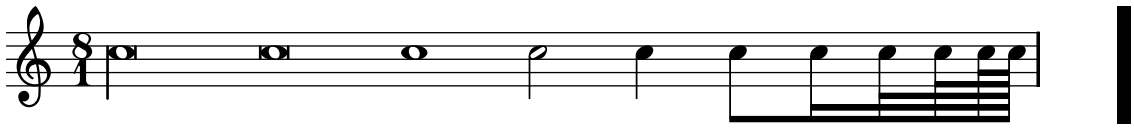




Dieser Abschnitt erklärt die Eingabe von Rhythmen, Pausen, Dauern, Bebalkung und Takten.

1.2.1 Rhythmen eingeben

Tondauern
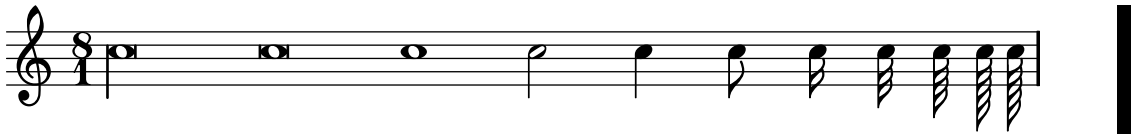
Notenlängen (Dauern) werden durch Zahlen und Punkte notiert: Dauern werden als reziproke Werte geschrieben. Zum Beispiel wird eine Viertelnote mit `4` notiert (weil sie eine 1/4-Note ist), eine halbe Note mit `2` (weil sie eine 1/2-Note ist). Noten, die länger als eine Ganze sind, müssen mit `\longa` (für die Longa, also vier Ganze) und `\breve` (für die Brevis, auch Doppelganze genannt) notiert werden. Notendauern bis hin zu 128steln sind unterstützt. Kürzere Notenwerte können auch notiert werden, können allerdings nur als Noten mit Balken auftreten.

```
\time 8/1
c\longa c\breve c1 c2
c4 c8 c16 c32 c64 c128 c128
```



Hier die selben Notendauern ohne die Balken.
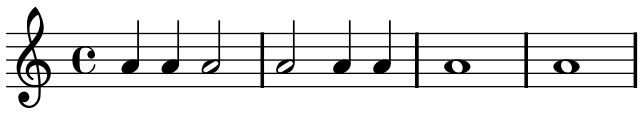
```
\time 8/1
\autoBeamOff
c\longa c\breve c1 c2
c4 c8 c16 c32 c64 c128 c128
```



Eine Note mit der vierfachen Dauer einer Brevis kann mit dem Befehl `\maxima` eingegeben werden, aber ihre Darstellung ist nur für die Alte Musiknotation unterstützt. Zu Einzelheiten siehe Abschnitt 2.9 [Notation von alter Musik], Seite 391.
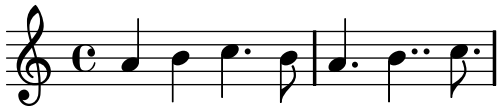
Wenn die Dauer hinter einer Notenbezeichnung nicht angegeben ist, wird die Dauer der vorhergehenden Note eingesetzt. Der Standardwert für die erste Note ist eine Viertel.

```
a a a2 a a4 a a1 a
```



Um punktierte Notendauern zu erhalten, muss einfach nur ein Punkt (`.`) hinter die Zahl der Dauer gesetzt werden. Zwei Punkte ergeben eine doppelte Punktierung, usw.

```
a4 b c4. b8 a4. b4.. c8.
```



Manche Notenlängen können nicht mit binären Dauern und Punkten dargestellt werden, sie können nur erreicht werden, indem man Noten überbindet. Für Einzelheiten siehe [Bindebögen], Seite 48.

Wie den Silben von Gesangstext eigene Dauern zugewiesen werden können und wie man sie an den Noten ausrichtet ist erklärt in Abschnitt 2.1 [Notation von Gesang], Seite 226.

Optional können Noten streng proportional nach ihrer exakten Dauer gesetzt werden. Zu Einzelheiten hierzu und weiteren Einstellungen für proportionale Notation siehe Abschnitt 4.5.5 [Proportionale Notation], Seite 502.

Punkte werden normalerweise nach oben verschoben, damit sie die Notenlinien nicht berühren. Punkte können manuelle über oder unter dem Notensystem gesetzt werden, zu Einzelheiten siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

Vordefinierte Befehle
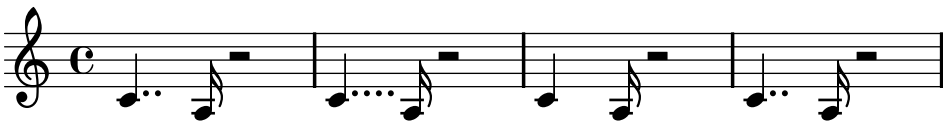
`\autoBeamOn`, `\autoBeamOff`, `\dotsUp`, `\dotsDown`, `\dotsNeutral`.

Ausgewählte Schnipsel

Changing the number of augmentation dots per note

This code demonstrates how to change the number of augmentation dots on a single note.

```
\relative c' {
  c4.. a16 r2 |
  \override Dots #'dot-count = #4
  c4.. a16 r2 |
  \override Dots #'dot-count = #0
  c4.. a16 r2 |
  \revert Dots #'dot-count
  c4.. a16 r2 |
}
```



Siehe auch

Glossar: Abschnitt “breve” in *Glossar*, Abschnitt “longa” in *Glossar*, Abschnitt “maxima” in *Glossar*, Abschnitt “note value” in *Glossar*, Abschnitt “Duration names notes and rests” in *Glossar*.

Notationsreferenz: [Automatische Balken], Seite 74, [Bindebögen], Seite 48, [Hälse], Seite 196, Abschnitt 1.2.1 [Rhythmen eingeben], Seite 41, Abschnitt 1.2.2 [Pausen eingeben], Seite 51, Abschnitt 2.1 [Notation von Gesang], Seite 226, Abschnitt 2.9 [Notation von alter Musik], Seite 391, Abschnitt 4.5.5 [Proportionale Notation], Seite 502.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Dots” in *Referenz der Interna*, Abschnitt “DotColumn” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Es gibt keine grundlegende Grenze für die Dauer von Pausen (sowohl kürzer als auch länger), aber die Anzahl an Symbolen ist begrenzt: Einzelne Pausen können von 128stel bis zur Maxima (8 Ganze) gesetzt werden.
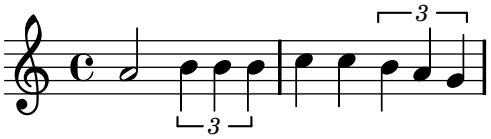
Andere rhythmische Aufteilungen

Triolen und andere rhythmische Aufteilungen werden aus einem musikalischen Ausdruck erstellt, indem dessen Tondauern mit einem Bruch multipliziert werden.

```
\times Bruch musikalischer Ausdruck
```

Die Dauer eines *musikalischen Ausdrucks* wird mit dem Bruch multipliziert. Der Nenner des Bruchs wird über (oder unter) den Noten ausgegeben, optional mit einer eckigen Klammer, die die Noten einfasst. Die üblichste Aufteilung ist die Triole, in welcher drei Noten die Länge von zwei haben, der Wert jeder einzelnen Note ist also 2/3 der notierten Länge.
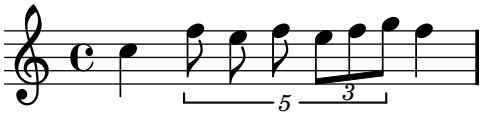
```
a2 \times 2/3 { b4 b b }
c4 c \times 2/3 { b4 a g }
```



Triolenklammern können manuell über oder unter dem Notensystem ausgegeben werden, siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

N-tolen können ineinander geschachtelt werden:

```
\autoBeamOff
c4 \times 4/5 { f8 e f \times 2/3 { e[ f g] } } f4
```



Wenn man die Eigenschaften von N-tolen verändern will, die zum selben musikalischen Zeitpunkt beginnen, muss `\tweak` eingesetzt werden.

Um die Dauern von Noten zu ändern, ohne die N-tolen-Klammern zu setzen, siehe [Tondauern skalieren], Seite 47.

Vordefinierte Befehle

`\tupletUp`, `\tupletDown`, `\tupletNeutral`.

Ausgewählte Schnipsel
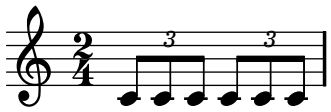
Mehrere Triolen notieren aber nur einmal \times benutzen

Die Eigenschaft `tupletSpannerDuration` bestimmt, wie lange jede der N-tolen innerhalb der Klammern nach dem `\times`-Befehl dauert. Auf diese Art können etwa viele Triolen nacheinander mit nur einem `\times`-Befehl geschrieben werden.

Im Beispiel sind zwei Triolen zu sehen, obwohl `\times` nur einmal geschrieben wurde.

Mehr Information über `make-moment` gibt es in "Verwaltung der Zeiteinheiten".

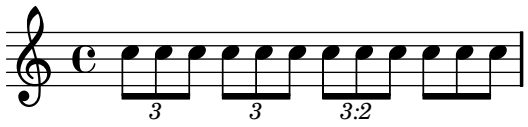
```
\relative c' {
  \time 2/4
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```



Die Zahl der N-tole verändern

Standardmäßig wird nur der Zähler des N-tolen-Bruchs über der Klammer dargestellt, wie er dem `\times`-Befehl übergeben wird. Man kann aber auch Zähler/Nenner ausgeben lassen, oder die Zahl vollständig unterdrücken.

```
\relative c'' {
  \times 2/3 { c8 c c }
  \times 2/3 { c8 c c }
  \override TupletNumber #'text = #tuplet-number::calc-fraction-text
  \times 2/3 { c8 c c }
  \override TupletNumber #'stencil = ##f
  \times 2/3 { c8 c c }
}
```



Nicht-standard-N-tolennummern

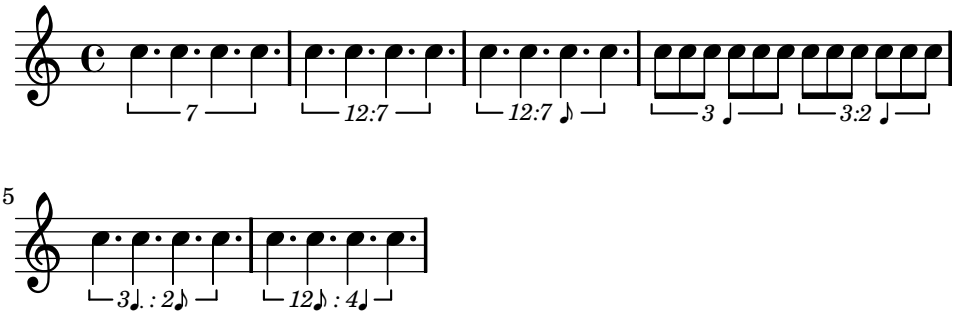
LilyPond stellt auch Formatierungsfunktionen zur Verfügung, mit denen N-tolennummern gesetzt werden können, die sich von dem eigentlichen Bruch unterscheiden. Auch ein Notenwert kann zu Nenner oder Zähler des Bruchs hinzugefügt werden.

```
\relative c'' {
  \once \override TupletNumber #'text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \times 2/3  { c4. c4. c4. c4. }
  \once \override TupletNumber #'text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \times 2/3  { c4. c4. c4. c4. }
  \once \override TupletNumber #'text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7) "8")
  \times 2/3  { c4. c4. c4. c4. }

  \once \override TupletNumber #'text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text "4")
  \times 2/3  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber #'text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text "4")
  \times 2/3  { c8 c8 c8 c8 c8 c8 }

  \once \override TupletNumber #'text =
    #(tuplet-number::fraction-with-notes "4." "8")
  \times 2/3  { c4. c4. c4. c4. }
  \once \override TupletNumber #'text =
    #(tuplet-number::non-default-fraction-with-notes 12 "8" 4 "4")
  \times 2/3  { c4. c4. c4. c4. }
```
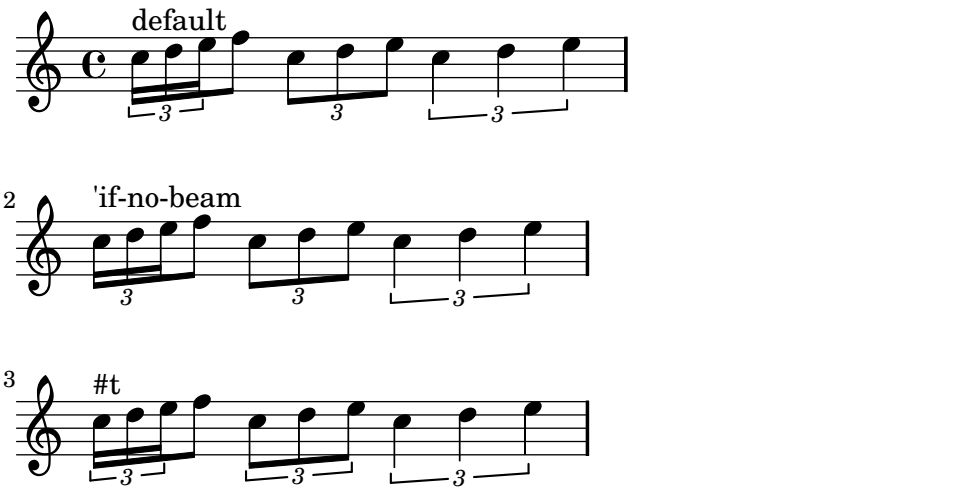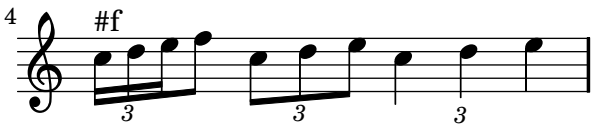
```
}
```



Controlling tuplet bracket visibility

The default behavior of tuplet-bracket visibility is to print a bracket unless there is a beam of the same length as the tuplet. To control the visibility of tuplet brackets, set the property `'bracket-visibility` to either `#t` (always print a bracket), `#f` (never print a bracket) or `#'if-no-beam` (only print a bracket if there is no beam).

```
music = \relative c'' {
  \times 2/3 { c16[ d e } f8]
  \times 2/3 { c8 d e }
  \times 2/3 { c4 d e }
}

\new Voice {
  \relative c' {
    << \music s4^"default" >>
    \override TupletBracket #'bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket #'bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket #'bracket-visibility = ##f
    << \music s4^"#f" >>
  }
}
```

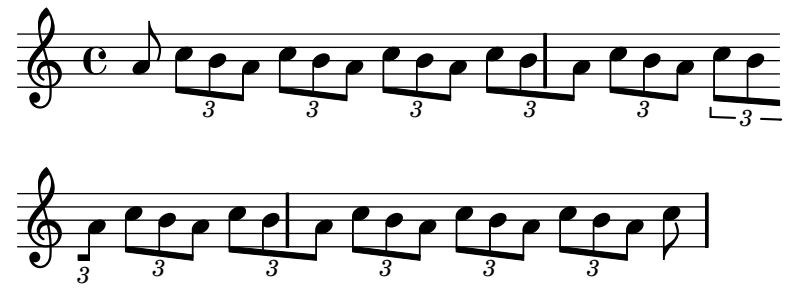




Zeilenumbrüche bei N-tolen mit Balken erlauben

Dieses künstliche Beispiel zeigt, wie sowohl automatische als auch manuelle Zeilenumbrüche innerhalb einer N-tole mit Balken erlaubt werden können. Diese unregelmäßige Bebalkung muss allerdings manuell gesetzt werden.

```
\layout {
  \context {
    \Voice
    % Permit line breaks within tuplets
    \remove "Forbid_line_break_engraver"
    % Allow beams to be broken at line breaks
    \override Beam #'breakable = ##t
  }
}
\relative c'' {
  a8
  \repeat unfold 5 { \times 2/3 { c[ b a] } }
  % Insert a manual line break within a tuplet
  \times 2/3 { c[ b \bar "" \break a] }
  \repeat unfold 5 { \times 2/3 { c[ b a] } }
  c8
}
```



Siehe auch

Glossar: Abschnitt “triplet” in *Glossar*, Abschnitt “tuplet” in *Glossar*, Abschnitt “polymetric” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Optimierungsmethoden” in *Handbuch zum Lernen*.

Notationrreferenz: [Verwaltung der Zeiteinheiten], Seite 105, [Tondauern skalieren], Seite 47, Abschnitt 5.3.4 [Der tweak-Befehl], Seite 533, [Polymetrische Notation], Seite 68.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “TupletBracket” in *Referenz der Interna*, Abschnitt “TupletNumber” in *Referenz der Interna*, Abschnitt “TimeScaledMusic” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Verzierungen können innerhalb von Triolenklammern gesetzt werden, *außer* wenn ein System mit einer Verzierung beginnt, die von einer N-tole gefolgt wird. In diesem besonderen Fall müssen die Verzierungen vor dem `\times`-Befehl gesetzt werden, damit sich keine Fehler ergeben.
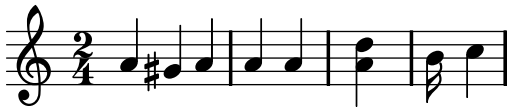
Wenn man eine N-tole zu Beginn eines Stückes notiert, das eine Tempobezeichnung mit `\tempo` enthält, müssen die Noten in einer explizit begonnenen Stimme notiert werden. Siehe auch Abschnitt “Voice enthält Noten” in *Handbuch zum Lernen*.

Tondauern skalieren

Die Dauer von einzelnen Noten, Pausen oder Akkorden kann mit einem Bruch multipliziert werden, indem hinter die Notendauer „*N/M“ (oder „*N“ wenn M 1 ist) geschrieben wird. Die Erscheinung der Noten oder Pausen wird dadurch nicht beeinflusst, die neue Dauer wird aber dazu benutzt, ihre Position im Takt zu errechnen und die neue Dauer in der MIDI-Ausgabe einzusetzen. Die Faktoren, mit denen multipliziert wird, können auch kombiniert werden, etwa „*L*M*/N“.

Im nächsten Beispiel nehmen die drei ersten Noten genau zwei Schläge ein, aber es wird keine Triolenklammer über ihnen ausgegeben.
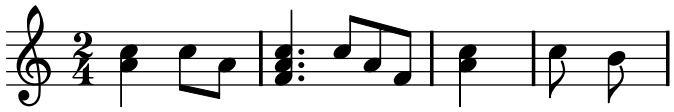
```
\time 2/4
% Alter durations to triplets
a4*2/3 gis4*2/3 a4*2/3
% Normal durations
a4 a4
% Double the duration of chord
<a d>4*2
% Duration of quarter, appears like sixteenth
b16*4 c4
```



Die Dauer von unsichtbaren Noten kann auch mit einem Faktor beeinflusst werden. Das ist sinnvoll, wenn man viele Takte überspringen muss, etwa `s1*23`.

Längere Notenabschnitte können auf die gleiche Art durch Multiplikation mit einem Bruch komprimiert werden, als ob jede Note, jeder Akkord oder jede Pause mit dem Bruch multipliziert würde. Damit bleibt das Aussehen der Musik unverändert, aber die interne Dauer der Noten wird mit dem Bruch multipliziert. Die Leerzeichen um den Punkt im Beispiel sind notwendig. Hier ein Beispiel, das zeigt, wie Noten komprimiert und ausgedehnt werden kann:

```
\time 2/4
% Normal durations
<c a>4 c8 a
% Scale music by *2/3
\scaleDurations #'(2 . 3) {
  <c a f>4. c8 a f
}
% Scale music by *2
\scaleDurations #'(2 . 1) {
  <c' a>4 c8 b
}
```



Eine Anwendung für diesen Befehl ist polymetrische Notation, siehe [Polymetrische Notation], Seite 68.

Siehe auch

Notationsreferenz: [Andere rhythmische Aufteilungen], Seite 43, [Unsichtbare Pausen], Seite 53, [Polymetrische Notation], Seite 68.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Bindebögen

Ein Bindebogen verbindet zwei benachbarte Noten der selben Tonhöhe. Als Resultat wird die Dauer der Notenlänge verlängert.
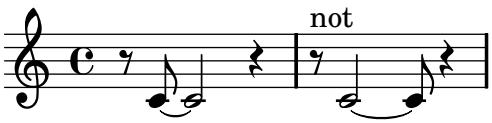
> **Achtung:** Bindebögen (engl. tie) dürfen nicht mit Legatobögen (engl. slur) verwechselt werden, durch die die Vortragsart bezeichnet wird, noch mit Phrasierungsbögen (engl. phrasing slur), die musikalische Phrasen anzeigen. Ein Bindebogen ist nur eine Art, die Tondauer zu verlängern, ähnlich etwa wie die Punktierung.

Ein Bindebogen wird mit der Tilde ~ (AltGr++) notiert.

```
a2 ~ a
```



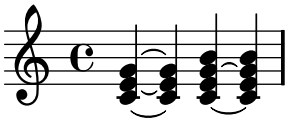
Bindebögen werden eingesetzt, wenn die Note entweder über eine Taktlinie hinüberreicht, oder wenn die entsprechende Dauer der Note nicht mit Punktierung erreicht werden kann. Bindebögen sollten auch benutzt werden, wenn Notenwerte über die inneren Unterteilungen von Takten hinüberreichen:



Wenn viele Noten über Taktlinien gebunden werden müssen, kann es einfacher sein, automatische Notenaufteilung einzustellen, wie beschrieben in [Automatische Aufteilung von Noten], Seite 71. Mit diesem Mechanismus werden lange Noten automatisch aufgeteilt, wenn sie über Taktgrenzen reichen.
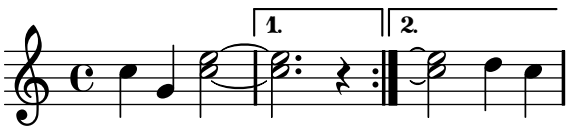
Wenn ein Bindebogen an einen Akkord gehängt wird, werden alle Noten dieses Akkordes übergebunden. Wenn kein Notenkopf passt, wird auch kein Bogen erzeugt. Noten in Akkorden können auch einzeln übergebunden werden, indem sie innerhalb des Akkordes hinter die entsprechende Note geschrieben werden.
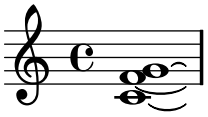
```
<c e g> ~ <c e g>
<c~ e g~ b> <c e g b>
```



Wenn die zweite Variante einer Wiederholung mit einer übergebundenen Note anfängt, muss der Bindebogen wie folgt notiert werden:

```
\repeat volta 2 { c g <c e>2 ~ }
\alternative {
  % First alternative: following note is tied normally
  { <c e>2. r4 }
  % Second alternative: following note has a repeated tie
  { <c e>2\repeatTie d4 c } }
```



So genannte *laissez vibrer*-Bögen werden verwendet um anzuzeigen, dass man die Musik ausklingen lassen soll. Sie werden in der Klavier-, Harfen-, anderer Saiteninstrument- und Schlagzeugnotation verwendet. Sie können folgendermaßen notiert werden:
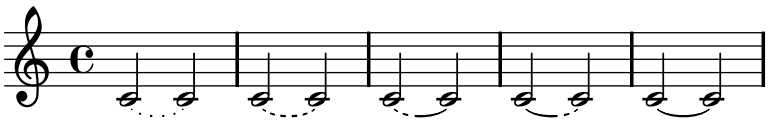
```
<c f g>1\laissezVibrer
```



Bindebögen können manuell über oder unter dem Notensystem gesetzt werden. Zu Einzelheiten siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.
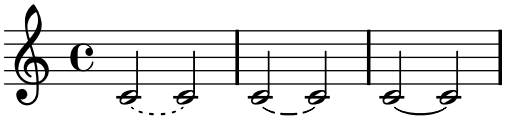
Bindebögen können durchgehend, gestrichelt, gepunktet oder in einer Kombination von Strichen und durchgehender Linie definiert werden.

```
\tieDotted
c2 ~ c
\tieDashed
c2 ~ c
\tieHalfDashed
c2 ~ c
\tieHalfSolid
c2 ~ c
\tieSolid
c2 ~ c
```



Eigene Strichelungsmuster können definiert werden:

```
\tieDashPattern #0.3 #0.75
c2 ~ c
\tieDashPattern #0.7 #1.5
c2 ~ c
\tieSolid
c2 ~ c
```



Die Definition von Muster für die Strichelung der Bindebögen hat die gleiche Struktur wie die Definition für Legatobögen. Zu weiterer Information zu komplizierten Strichelungsmustern, siehe die Schnipsel im Abschnitt [Legatobögen], Seite 116.

Vordefinierte Befehle
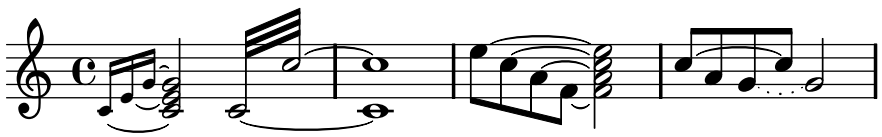
`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieDashPattern`, `\tieHalfDashed`, `\tieHalfSolid`, `\tieSolid`.

Ausgewählte Schnipsel

Überbindungen für Arpeggio benutzen

Überbindungen werden teilweise benutzt, um Arpeggios zu notieren. In diesem Fall stehen die übergebundenen Noten nicht unbedingt hintereinander. Das Verhalten kann erreicht werden, indem die `tieWaitForNote`-Eigenschaft auf `#t` gesetzt wird. Diese Funktion ist auch sinnvoll, um etwa ein Tremolo mit einem Akkord zu überbinden, kann aber prinzipiell auch für normale Überbindungen eingesetzt werden

```
\relative c' {
  \set tieWaitForNote = ##t
  \grace { c16[ ~ e ~ g] ~ } <c, e g>2
  \repeat tremolo 8 { c32 ~ c' ~ } <c c,>1
  e8 ~ c ~ a ~ f ~ <e' c a f>2
  \tieUp
  c8 ~ a
  \tieDown
  \tieDotted
  g8 ~ c g2
}
```



Bindebögen manuell setzen

Überbindungen können manuell gesetzt werden, indem man die `tie-configuration`-Eigenschaft des `TieColumn`-Objekts beeinflusst. Die erste Zahl zeigt den Abstand von der Mitte in Notensystemabständen an, die zweite Zahl zeigt die Richtung an (1 = nach oben, -1 = nach unten).

```
\relative c' {
  <c e g>2 ~ <c e g>
  \override TieColumn #'tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>2 ~ <c e g>
}
```



Siehe auch

Glossar: Abschnitt “tie” in *Glossar*, Abschnitt “laissez vibrer” in *Glossar*.

Notationsreferenz: [Legatobögen], Seite 116, [Automatische Aufteilung von Noten], Seite 71.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “LaissezVibrerTie” in *Referenz der Interna*, Abschnitt “LaissezVibrerTieColumn” in *Referenz der Interna*, Abschnitt “TieColumn” in *Referenz der Interna*, Abschnitt “Tie” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Der Wechsel zwischen Systemen bei aktiver Überbindung produziert keinen gekrümmten Bogen.

Änderung von Schlüssel oder Oktavierung zwischen übergebundenen Noten ist nicht richtig definiert. In diesen Fällen kann es besser sein, einen Legatobogen zu verwenden.

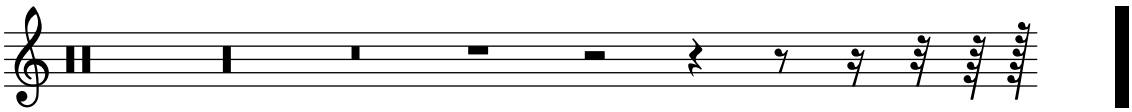
1.2.2 Pausen eingeben

Pausen werden als Teil der musikalischen Ausdrücke zusammen mit den Noten notiert.

Pausen

Pausen werden wie Noten eingegeben, ihre Bezeichnung ist `r`. Dauern, die länger als eine Ganze sind, haben die vordefinierten Befehle:
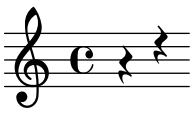
```
\new Staff {
  % These two lines are just to prettify this example
  \time 16/1
  \override Staff.TimeSignature #'stencil = ##f
  % Print a maxima rest, equal to four breves
  r\maxima
  % Print a longa rest, equal to two breves
  r\longa
  % Print a breve rest
  r\breve
  r1 r2 r4 r8 r16 r32 r64 r128
}
```



Pausen, die ganze Takte ausfüllen und in der Taktmitte zentriert werden sollen, müssen als mehrtaktige Pausen eingegeben werden. Sie können sowohl für einen einzigen Takt als auch für mehrere Takte verwendet werden, Näheres im Abschnitt [Ganztaktpausen], Seite 54.

Um die vertikale Position einer Pause explizit festzulegen, kann eine Note eingegeben werden, gefolgt vom Befehl `\rest`. Die Pause wird dann an die Stelle gesetzt, wo sich sonst die Note befinden würde. Damit wird die manuelle Formatierung von mehrstimmiger Musik sehr viel einfacher, da die Formatierungsfunktion zur automatischen Auflösung von Zusammenstößen diese Pausen nicht mit einbezieht.

```
a4\rest d4\rest
```



Ausgewählte Schnipsel

Pausenstile

Pausen können in verschiedenen Stilen dargestellt werden.

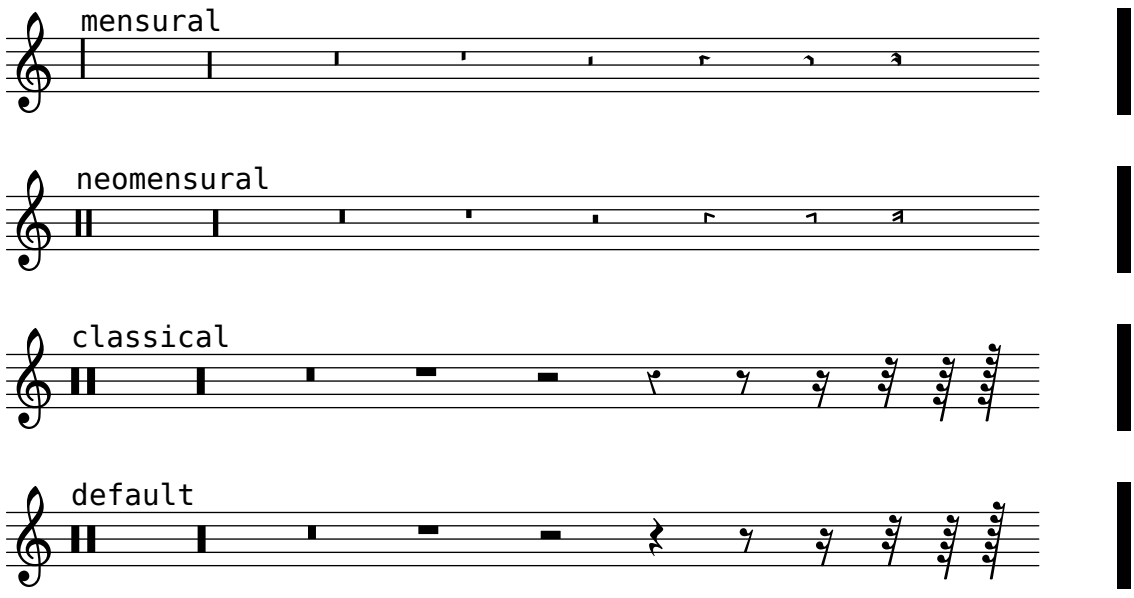
```
\layout {
  indent = 0
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}

\new Staff \relative c {
  \cadenzaOn
  \override Staff.Rest #'style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""

  \override Staff.Rest #'style = #'neomensural
  r\maxima^\markup \typewriter { neomensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""

  \override Staff.Rest #'style = #'classical
  r\maxima^\markup \typewriter { classical }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
  \bar ""

  \override Staff.Rest  #'style = #'default
  r\maxima^\markup \typewriter { default }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
}
```



Siehe auch

Glossar: Abschnitt “breve” in *Glossar*, Abschnitt “longa” in *Glossar*, Abschnitt “maxima” in *Glossar*.

Notationsreferenz: [Ganztaktpausen], Seite 54.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Rest” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Es gibt keine grundlegende Grenze für die Dauer von Pausen (sowohl kürzer als auch länger), aber die Anzahl von Symbolen ist begrenzt: Es gibt Zeichen für Pausen von einer 128 bis zu einer Maxima (8 Ganze).

Unsichtbare Pausen
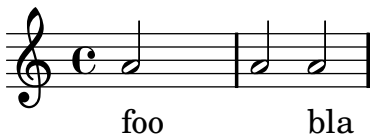
Eine unsichtbare Pause (auch als „skip“ oder Überspringung bezeichnet) kann wie eine Note eingegeben werden, die Notationsbezeichnung ist `s`.

```
a4 a4 s4 a4 \skip 1 a4
```



Die `s`-Syntax steht nur im Noten- oder Akkordmodus zur Verfügung. In anderen Situationen, z. B. innerhalb eines Liedtextes, muss der Befehl `\skip` benutzt werden. `\skip` benötigt eine explizite Dauerangabe.
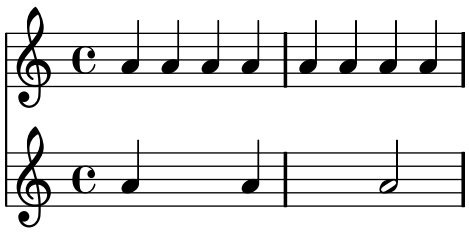
```
<<
  {
    a2 \skip2 a2 a2
  }
  \new Lyrics {
    \lyricmode {
      foo2 \skip 1 bla2
    }
  }
>>
```



Weil `\skip` ein Befehl ist, wirkt er sich nicht auf die Dauer der folgenden Noten aus, anders als `s`.

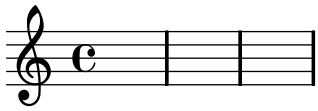
```
<<
  {
    \repeat unfold 8 { a4 }
  }
  {
    a4 \skip 2 a |
    s2 a
  }
```

```
>>
```



Die Platzhalterpause mit `s` erstellt `Staff`- und `Voice`-Kontext, wenn es erforderlich ist, genauso wie Noten und Pausen.

```
s1 s s
```



Der Überspringungsbefehl (`\skip`) ist einfach ein leerer Platzhalter. Durch ihn wird überhaupt nichts gesetzt, auch keine transparenten Objekte.

```
% This is valid input, but does nothing
\skip 1 \skip1 \skip 1
```

Siehe auch

Handbuch zum lernen: Abschnitt “Sichtbarkeit und Farbe von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: [Unsichtbare Noten], Seite 193, Abschnitt 5.4.7 [Sichtbarkeit von Objekten], Seite 546.

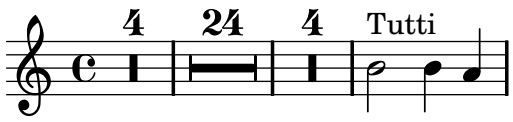
Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “SkipMusic” in *Referenz der Interna*

Ganztaktpausen

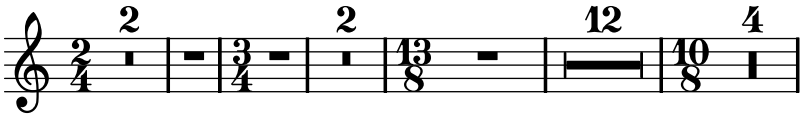
Pausen für einen oder mehrere ganze Takte werden wie Noten eingegeben, wobei die Bezeichnung ein Großbuchstabe `R` ist:

```
% Rest measures contracted to single measure
\compressFullBarRests
R1*4
R1*24
R1*4
b2^"Tutti" b4 a4
```
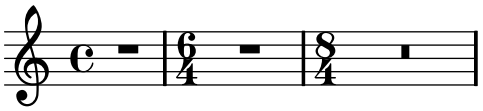


Die Dauer von Ganztaktpausen wird genauso angegeben wie die Dauer von Noten. Die Dauer einer Ganztaktpause muss immer eine ganze Anzahl an Taktlängen sein, weshalb Punktierungen und Brüche recht häufig eingesetzt werden müssen.

```
\compressFullBarRests
\time 2/4
R1 | R2 |
\time 3/4
R2. | R2.*2 |
\time 13/8
R1*13/8 | R1*13/8*12 |
\time 10/8
R4*5*4 |
```
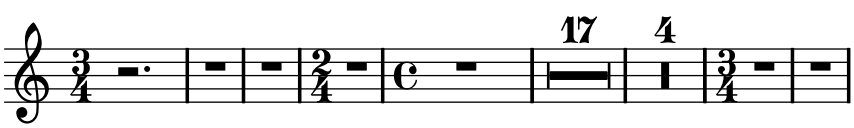


Eine Ganztaktpause wird abhängig von der Taktart entweder als Ganze oder Brevis-Pause gesetzt, zentriert im Takt.

```
\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |
```



In den Standardeinstellungen werden mehrtaktige Pausen ausgeschrieben gesetzt, sodass sie die entsprechende Anzahl von Takten einnehmen. Alternativ kann die mehrtaktige Pause aber auch nur in einem Takt angezeigt werden, der ein Mehrtaktpausensymbol beinhaltet, wobei die Anzahl der Takte der Pausendauer über dem Pausenzeichen ausgegeben wird:

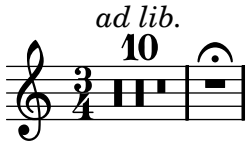
```
% Default behavior
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Rest measures contracted to single measure
\compressFullBarRests
r1 | R1*17 | R1*4 |
% Rest measures expanded
\expandFullBarRests
\time 3/4
R2.*2 |
```



Textbeschriftung kann Mehrtaktpausen mit `\markup` hinzugefügt werden. Ein vordefinierte Befehl `\fermataMarkup` fügt eine Fermate ein.

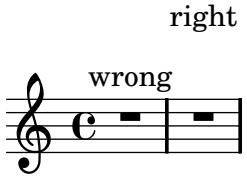
```
\compressFullBarRests
\time 3/4
```

```
R2.*10^\markup { \italic "ad lib." }
R2.^\fermataMarkup
```



Achtung: Beschriftungen, die an Mehrtaktpausen gehängt werden, sind Objekte vom Typ `MultiMeasureRestText`, nicht vom Typ `TextScript`. Änderungen etwa mit `\override` müssen auf das richtige Objekt gerichtet werden, damit sie nicht ignoriert werden. Siehe auch das folgende Beispiel.

```
% This fails, as the wrong object name is specified
\override TextScript #'padding = #5
R1^"wrong"
% This is the correct object name to be specified
\override MultiMeasureRestText #'padding = #5
R1^"right"
```



Wenn eine Mehrtaktpause direkt auf einen Auftakt mit `\partial` folgt, werden möglicherweise daraus resultierende Taktprüfungswarnungen nicht angezeigt.

Vordefinierte Befehle

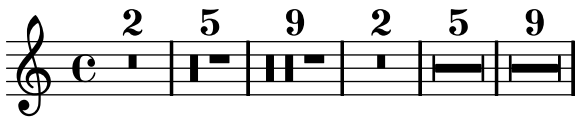
`\textLengthOn`, `\textLengthOff`, `\fermataMarkup`, `\compressFullBarRests`, `\expandFullBarRests`.

Ausgewählte Schnipsel

Die Erscheinung von Pausentakten ändern

Wenn zehn oder weniger Pausentakte vorkommen, wird eine Reihe von Longa- und Brevispausen (auch Kirchenpausen genannt) gesetzt, bei mehr Takten wird eine Line mit der Taktanzahl ausgegeben. Der vorgegebene Wert von zehn kann geändert werden, indem man die `expand-limit`-Eigenschaft setzt:

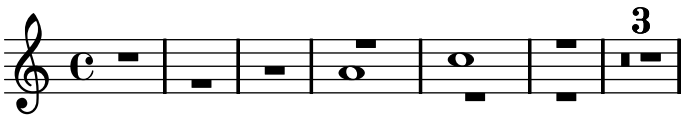
```
\relative c'' {
  \compressFullBarRests
  R1*2 | R1*5 | R1*9
  \override MultiMeasureRest #'expand-limit = #3
  R1*2 | R1*5 | R1*9
}
```



Positionierung von Ganztaktpausen

Anders als bei normalen Pausen gibt es keinen direkten Befehl, um die vertikale Position von Ganztaktpausen zu beeinflussen, indem man sie an eine Tonhöhe anhängt. In polyphoner Notation wird aber dennoch die Position der Pausen von geraden und ungeraden Stimmen voneinander unterschieden. Die Position von Ganztaktpausen kann wie folgt verändert werden:

```
\relative c'' {
  % MMR - Multi-Measure Rest
  % MMRs by default are set under the fourth line
  R1
  % They can be moved with an override
  \override MultiMeasureRest #'staff-position = #-2
  R1
  % A value of 0 is the default position;
  % the following trick moves the rest to the center line
  \override MultiMeasureRest #'staff-position = #-0.01
  R1
  % MMRs in odd-numbered voices are under the top line
  << { R1 } \\ { a1 } >>
  % MMRs in even-numbered voices are under the bottom line
  << { c1 } \\ { R1 } >>
  % They remain separated even in empty measures
  << { R1 } \\ { R1 } >>
  % This brings them together even though there are two voices
  \compressFullBarRests
  <<
    \revert MultiMeasureRest #'staff-position
    { R1*3 }
    \\
    \revert MultiMeasureRest #'staff-position
    { R1*3 }
  >>
}
```



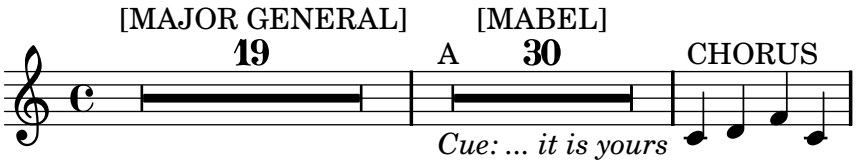
Textbeschriftung und Mehrtaktpausen

Textbeschriftungen, die an Mehrtaktpausen gehängt wird, wird über oder unter der Pause zentriert. Lange Beschriftungen lassen den Takt nicht breiter werden. Um eine Mehrtaktpause einer Beschriftung anzupassen, muss eine unsichtbare Pause mit der Beschriftung direkt vor der Mehrtaktpause eingesetzt werden.

Man sollte beachten, dass unsichtbare Pausen automatische Taktstriche nach sich ziehen. Text, der an eine unsichtbare Pause gehängt wird, ist links ausgerichtet an der Position, wo die Pause erscheinen würde. Wenn aber die Länge des Taktes durch die Länge des Textes bestimmt wird, sieht es so aus, als ob der Text zentriert gesetzt ist.

```
\relative c' {
  \compressFullBarRests
  \textLengthOn
  s1*0^\markup { [MAJOR GENERAL] }
  R1*19
```

```
  s1*0_\markup { \italic { Cue: ... it is yours } }
  s1*0^\markup { A }
  R1*30^\markup { [MABEL] }
  \textLengthOff
  c4^\markup { CHORUS } d f c
}
```



Siehe auch

Glossar: Abschnitt “multi-measure rest” in *Glossar*.

Notationsreferenz: [Tondauern], Seite 41, Abschnitt 1.8 [Text], Seite 201, Abschnitt 1.8.2 [Text formatieren], Seite 210, [Textarten], Seite 201.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “MultiMeasureRest” in *Referenz der Interna*, Abschnitt “MultiMeasureRestNumber” in *Referenz der Interna*, Abschnitt “MultiMeasureRestText” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Wenn man versucht, mit Fingersatz (etwa `R1*10-4` Zahlen über Ganztaktpausen zu setzen, kann die Zahl des Fingersatzes (4) mit der Taktanzahl (10) zusammenstoßen.

Es gibt keine Möglichkeit, normale Pausen automatisch zu Ganztaktpausen zu reduzieren.

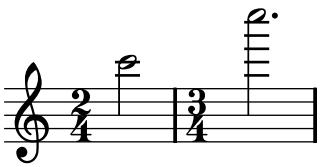
Ganztaktpausen werden bei der Vermeidung von Zusammenstößen nicht berücksichtigt.

1.2.3 Rhythmen anzeigen lassen

Taktangabe

Taktangaben werden wie folgt erstellt.

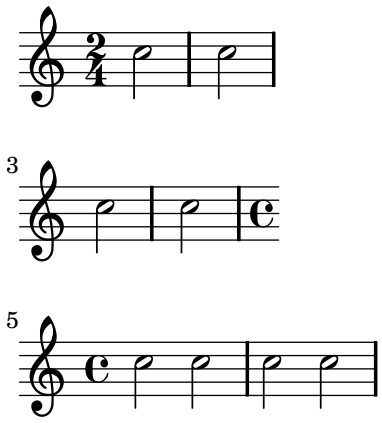
```
\time 2/4 c'2
\time 3/4 c'2.
```



Taktangaben werden zu Beginn eines Stückes gesetzt und immer dann, wenn sich die Taktart ändert. Wenn eine Änderung am Ende einer Zeile geschieht, wird eine warnende Taktangabe am Ende der Zeile ausgegeben. Dieses Verhalten kann verändert werden, siehe Abschnitt 5.4.7 [Sichtbarkeit von Objekten], Seite 546.
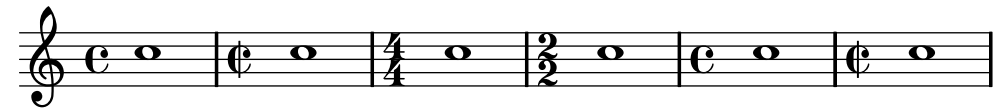
```
\time 2/4
c2 c
\break
c c
\break
```

```
\time 4/4
c c c c
```



Das Symbol für die Taktarten 2/2 und 4/4 kann in ein Zahlensymbol umgewandelt werden:

```
% Default style
\time 4/4 c1
\time 2/2 c1
% Change to numeric style
\numericTimeSignature
\time 4/4 c1
\time 2/2 c1
% Revert to default style
\defaultTimeSignature
\time 4/4 c1
\time 2/2 c1
```
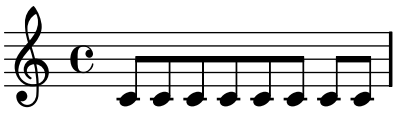


Symbole für Modus und Proprietas der mensuralen Notation werden behandelt unter [Mensurale Taktartenbezeichnungen], Seite 397.

Zusätzlich zu der gedruckten Taktart werden mit der Definition des Befehls `\time` auch die Standardwerte fürt die Eigenschaften `baseMoment`, `beatStructure` und `beamExtensions` gesetzt. Die vordefinierten Standardwerte für diese Eigenschaften finden sich in ‘`scm/time-signature-settings.scm`’. Die existierenden Standardwerte können verändert oder neue Standardwerte hinzugefügt werden.

```
\score {
  \new Staff {
    \relative c' {
      \overrideTimeSignatureSettings
        #'(4 . 4)  % timeSignatureFraction
        #'(1 . 4)  % baseMomentFraction
        #'(3 1)    % beatStructure
        #'()       % beamExceptions
      \time 4/4
      \repeat unfold 8 { c8 } |
    }
  }
```
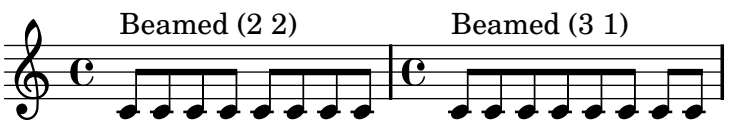
```
}
```



`\overrideTimeSignatureSettings` braucht fünf Argumente:

1. *`timeSignatureFraction`* (Taktart-Bruch), ein Scheme-Paar, das den Takt beschreibt.
2. *`baseMomentFraction`* (Grundmoment-Bruch), ein Scheme-Paar, das den Zähler und Nenner der Grundschlageinheit der Taktart enhält.
3. *`beatStructure`* (Taktzeit-Struktur), eine Scheme-Liste, die die Struktur der Taktschläge anzeigt, in Einheiten des Grundmoments.
4. *`beamExceptions`* (Balken-Ausnahmen), eine Aliste, die alle Bebalkungsregeln für die Taktart enthält, außer dem Balken, der zum Taktende endet, wie beschrieben in [Einstellung von automatischen Balken], Seite 77.

Der Kontext, der `\overrideTimeSignatureSettings` enthält, muss begonnen sein, bevor `\overrideTimeSignatureSettings` aufgerufen wird. Das heißt, dass er entweder explizit begonnen wird oder sich Noten in dem Kontext befinen müssen, bevor `\overrideTimeSignatureSettings` aufgerufen wird:
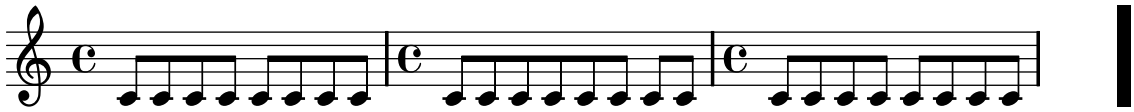
```
\score {
  \relative c' {
    % This call will fail because the context isn't yet instantiated
    \overrideTimeSignatureSettings
      #'(4 . 4)  % timeSignatureFraction
      #'(1 . 4)  % baseMomentFraction
      #'(3 1)    % beatStructure
      #'()       % beamExceptions
    \time 4/4
    c8^\markup {"Beamed (2 2)"}
    \repeat unfold 7 { c8 } |
    % This call will succeed
    \overrideTimeSignatureSettings
      #'(4 . 4)  % timeSignatureFraction
      #'(1 . 4)  % baseMomentFraction
      #'(3 1)    % beatStructure
      #'()       % beamExceptions
    \time 4/4
    c8^\markup {"Beamed (3 1)"}
    \repeat unfold 7 { c8 } |
  }
}
```



Veränderte Werte der Taktart-Eigenschaften können wieder auf den Standard zurückgesetzt werden:

```
\score{
  \relative c' {
```

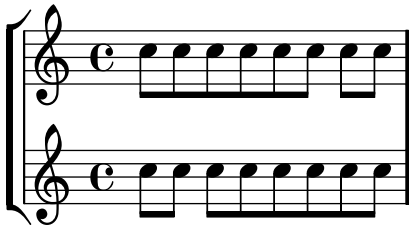
```
    \repeat unfold 8 { c8 } |
    \overrideTimeSignatureSettings
      #'(4 . 4)  % timeSignatureFraction
      #'(1 . 4)  % baseMomentFraction
      #'(3 1)    % beatStructure
      #'()       % beamExceptions
    \time 4/4
    \repeat unfold 8 { c8 } |
    \revertTimeSignatureSettings #'(4 . 4)
    \time 4/4
    \repeat unfold 8 { c8 } |
  }
}
```



Unterschiedliche Werte der Standard-Taktarteigenschaften für unterschiedliche Notensysteme können eingerichtet werden, indem man den `Timing_translator` und den `Default_bar_line_engraver` aus dem `Score`-Kontext in den `Staff`-Kontext verschiebt.

```
\score {
  \new StaffGroup <<
    \new Staff {
      \overrideTimeSignatureSettings
        #'(4 . 4)  % timeSignatureFraction
        #'(1 . 4)  % baseMomentFraction
        #'(3 1)    % beatStructure
        #'()       % beamExceptions
      \time 4/4
      \repeat unfold 8 {c''8}
    }
    \new Staff {
      \overrideTimeSignatureSettings
        #'(4 . 4)  % timeSignatureFraction
        #'(1 . 4)  % baseMomentFraction
        #'(1 3)    % beatStructure
        #'()       % beamExceptions
      \time 4/4
      \repeat unfold 8 {c''8}
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Timing_translator"
      \remove "Default_bar_line_engraver"
    }
    \context {
      \Staff
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
```

```
    }
  }
}
```



Vordefinierte Befehle

`\numericTimeSignature`, `\defaultTimeSignature`.

Ausgewählte Schnipsel

Die Taktart verändern ohne die Bebalkung zu beeinflussen

Der `\time`-Befehl verändert die Eigenschaften `timeSignatureFraction`, `beatLength`, `beatGrouping` und `measureLength` im `Timing`-Kontext, welcher normalerweise gleichbedeutend mit `Score` ist. Wenn der Wert von `timeSignatureFraction` verändert wird, wird die neue Taktart ausgegeben, ohne die anderen Eigenschaften zu beeinflussen:

```
\markup {
  This snippet is deprecated as of 2.13.5 and will be removed in 2.14
}
```
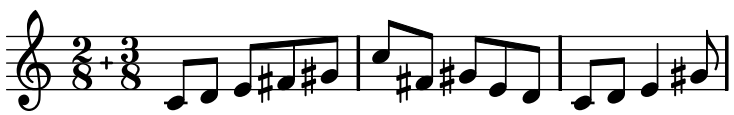
This snippet is deprecated as of 2.13.5 and will be removed in 2.14

Zusammengesetzte Taktarten

Ungerade Taktarten werden (wie etwa "5/8") werden oft als zusammengesetzte Taktarten interpretiert (bspw. "3/8 + 2/8"), in welchen zwei oder mehr Teiltakte unterschieden werden. LilyPond kann derartige Noten produzieren, indem entsprechende Taktarten gesetzt werden und die automatische Bebalkung angepasst wird.

```
#(define ((compound-time one two num) grob)
   (grob-interpret-markup grob
                          (markup #:override '(baseline-skip . 0) #:number
                                  (#:line ((#:column (one num))
                                           #:vcenter "+"
                                           (#:column (two num)))))))
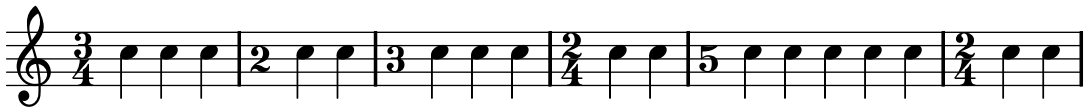
\relative c' {
  \override Staff.TimeSignature #'stencil = #(compound-time "2" "3" "8")
  \time 5/8
  \set Staff.beatStructure = #'(2 3)
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```



Time signature printing only the numerator as a number (instead of the fraction)

Sometimes, a time signature should not print the whole fraction (e.g. 7/4), but only the numerator (7 in this case). This can be easily done by using `\override Staff.TimeSignature #'style = #'single-digit` to change the style permanently. By using `\revert Staff.TimeSignature #'style`, this setting can be reversed. To apply the single-digit style to only one time signature, use the `\override` command and prefix it with a `\once`.

```
\relative c'' {
  \time 3/4
  c4 c c
  % Change the style permanently
  \override Staff.TimeSignature #'style = #'single-digit
  \time 2/4
  c4 c
  \time 3/4
  c4 c c
  % Revert to default style:
  \revert Staff.TimeSignature #'style
  \time 2/4
  c4 c
  % single-digit style only for the next time signature
  \once \override Staff.TimeSignature #'style = #'single-digit
  \time 5/4
  c4 c c c c
  \time 2/4
  c4 c
}
```



Siehe auch

Glossar: Abschnitt “time signature” in *Glossar*

Notationsreferenz: [Mensurale Taktartenbezeichnungen], Seite 397, [Verwaltung der Zeiteinheiten], Seite 105.
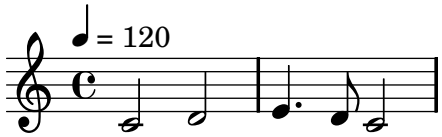
Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “TimeSignature” in *Referenz der Interna*, Abschnitt “Timing_translator” in *Referenz der Interna*.

Metronomangabe
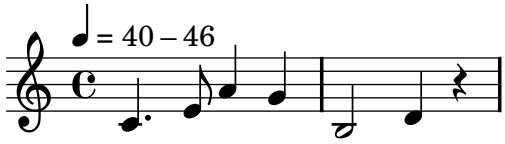
Eine Metronomanweisung wird wie folgt erstellt:

```
\tempo 4 = 120
c2 d
e4. d8 c2
```
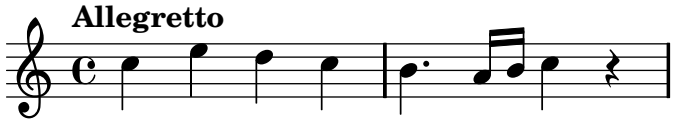


Metronombezeichnungen können auch für einen Zahlenbereich notiert werden:

```
\tempo 4 = 40 ~ 46
c4. e8 a4 g
b,2 d4 r
```
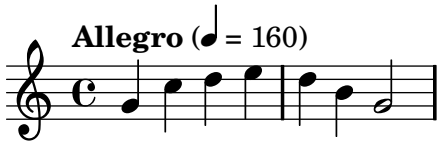


Anstelle dessen kann auch Text als Argument angegeben werden:

```
\tempo "Allegretto"
c4 e d c
b4. a16 b c4 r4
```
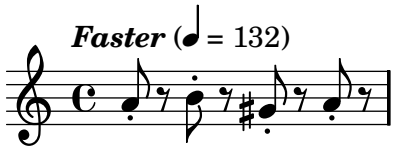


Wenn eine Metronombezeichnung und Text kombiniert wird, wird die Metronombezeichnung automatisch in Klammern gesetzt:

```
\tempo "Allegro" 4 = 160
g4 c d e
d4 b g2
```
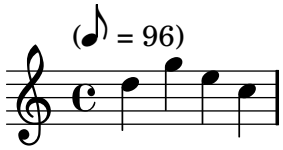


Der Text kann ein beliebiges Textbeschriftungsobjekt sein:

```
\tempo \markup { \italic Faster } 4 = 132
a8-. r8 b-. r gis-. r a-. r
```



Eine Metronombezeichnung in Klammern ohne Text kann erstellt werden, indem eine leere Zeichenkette hinzugefügt wird:

```
\tempo "" 8 = 96
d4 g e c
```



Ausgewählte Schnipsel

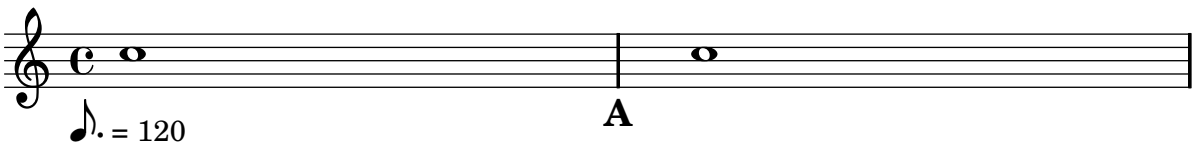
Metronom- und Übungszeichen unter das System setzen

Normalerweise werden Metronom- und Übungszeichen über dem Notensystem ausgegeben. Um sie unter das System zu setzen, muss die `direction`-Eigenschaft von `MetronomeMark` oder `RehearsalMark` entsprechend verändert werden.

```
\layout { ragged-right = ##f }

{
  % Metronome marks below the staff
  \override Score.MetronomeMark #'direction = #DOWN
  \tempo 8. = 120
  c''1
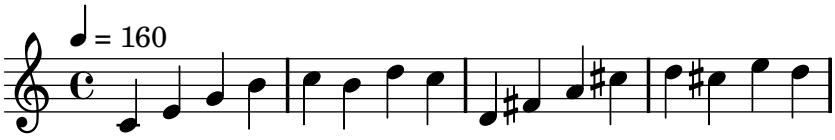
  % Rehearsal marks below the staff
  \override Score.RehearsalMark #'direction = #DOWN
  \mark \default
  c''1
}
```



Das Tempo ohne Metronom-Angabe verändern

Um das Tempo für die MIDI-Ausgabe zu ändern, ohne eine Tempoangabe in den Noten auszugeben, kann die Metronombezeichnung unsichtbar gemacht werden:
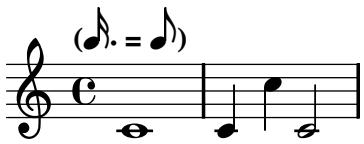
```
\score {
  \new Staff \relative c' {
    \tempo 4 = 160
    c4 e g b
    c4 b d c
    \set Score.tempoHideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```



Eine Metronombezeichnung als Textbeschriftung erstellen

Neue Metronombezeichnungen können als Textbeschriftung erstellt werden, aber sie ändern nicht das Tempo für die MIDI-Ausgabe.

```
\relative c' {
  \tempo \markup {
    \concat {
      (
      \smaller \general-align #Y #DOWN \note #"16." #1
      " = "
      \smaller \general-align #Y #DOWN \note #"8" #1
      )
    }
  }
  c1
  c4 c' c,2
}
```



Zu Einzelheiten siehe Abschnitt 1.8.2 [Text formatieren], Seite 210.

Siehe auch

Glossar: Abschnitt “metronome” in *Glossar*, Abschnitt “metronomic indication” in *Glossar*, Abschnitt “tempo indication” in *Glossar*, Abschnitt “metronome mark” in *Glossar*.

Notationsreferenz: Abschnitt 1.8.2 [Text formatieren], Seite 210, Abschnitt 3.5 [MIDI-Ausgabe], Seite 451.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “MetronomeMark” in *Referenz der Interna*.
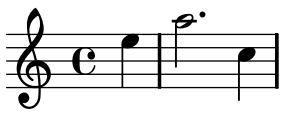
Auftakte

Verkleinerte Takte, wie etwa ein Auftakt, werden mit dem Befehl `\partial` notiert, dessen Syntax lautet:

```
\partial Dauer
```

wobei *Dauer* die rhythmische Länger der Noten darstellt, die vor dem ersten vollständigen Takt gesetzt werden sollen:

```
\partial 4 e4 |
a2. c,4 |
```



Intern wird `\partial Dauer` übersetzt nach:

```
\set Timing.measurePosition -Länge der Dauer
```

Zum Beispiel wird aus `\partial 8*3`:

```
\set Timing.measurePosition = #(ly:make-moment -3 8)
```

Die Eigenschaft `measurePosition` (Takt-Position) enthält eine rationale Zahl, die anzeigt, wie groß der Abstand zum Taktanfang ist. Deshalb ist sie eine negative Zahl; `\partial 4` wird also intern übersetzt zu `-4` was soviel bedeutet wie: „Eine Viertel bleibt übrig vom ganzen Takt.“

Siehe auch

Glossar: Abschnitt “anacrusis” in *Glossar*.

Notationsreferenz: [Verzierungen], Seite 99.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Timing_translator” in *Referenz der Interna*.
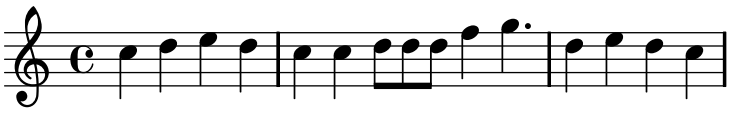
Bekannte Probleme und Warnungen

`\partial` ist nur für den Anfang eines Stückes vorgesehen. Wenn der Befehl innerhalb eines Stückes verwendet wird, können seltsame Warnungen auftreten. In solchem Fall sollten Sie `\set Timing.measurePosition` benutzen.

Musik ohne Metrum
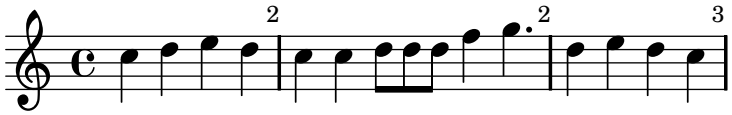
Taktlinien und Taktzahlen werden automatisch erzeugt. Für Musik ohne Metrum hingegen (etwa Kadenzen) ist das jedoch nicht erwünscht. Mit den Befehlen `\cadenzaOn` und `\cadenzaOff` kann dieses Verhalten ausgeschaltet und wieder angeschaltet werden.

```
c4 d e d
\cadenzaOn
c4 c d8[ d d] f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```
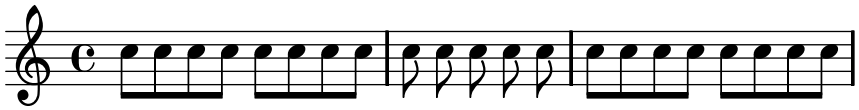


Taktnummerierung wird am Ende der Kadenz wieder aufgenommen, als ob es die Kadenz nicht gegeben hätte:

```
% Show all bar numbers
\override Score.BarNumber #'break-visibility = #all-visible
c4 d e d
\cadenzaOn
c4 c d8[ d d] f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```




```
\repeat unfold 8 { c8 }
\cadenzaOn
\repeat unfold 5 { c8 }
\bar"|"
\cadenzaOff
\repeat unfold 8 { c8 }
```



Automatische Bebalkung wird durch `\cadenzeOn` ausgestellt und durch `\cadenzaOff` wieder angestellt. Darum müssen alle Balken in Kadenzen manuell eingegeben werden (siehe [Manuelle Balken], Seite 84).

Diese vordefinierten Befehle wirken sich auf alle Systeme in der Partitur aus, auch wenn sie nur in einer einzigen Stimme notiert werden. Um dieses Verhalten zu ändern, müsen Sie `Timing_translator` aus dem `Score`-Kontext in den `Staff`-Kontext verschieben, wie gezeigt in [Polymetrische Notation], Seite 68.

Vordefinierte Befehle

`\cadenzaOn`, `\cadenzaOff`.

Siehe auch

Glossar: Abschnitt “cadenza” in *Glossar*.

Notationsreferenz: Abschnitt 5.4.7 [Sichtbarkeit von Objekten], Seite 546, [Polymetrische Notation], Seite 68 [Manuelle Balken], Seite 84.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Bekannte Probleme und Warnungen

LilyPond fügt Zeilen- und Seitenumbrüche nur an einer Taktlinie ein. Wenn die Kadenz nicht vor einem Umbruch endet, müssen Sie selber unsichtbare Taktlinien mit

```
\bar ""
```

einfügen, um anzuzeigen, wo umgebrochen werden darf.

Sie müssen explizit einen `Voice`-Kontext erstellen, wenn Sie ein Stück mit `cadenzaOn` beginnen wollen, weil sonst ein seltsamer Fehler auftreten kann.

```
\new Voice {
  \relative c' {
    \cadenzaOn
    c16[^"Solo Free Time" d e f] g2.
    \bar "||"
    \cadenzaOff
  }
}
```

Polymetrische Notation

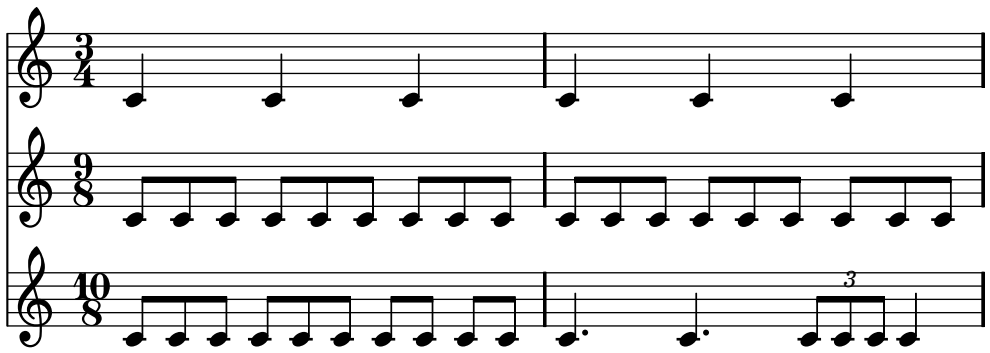
Polymetrische Notation ist unterstützt, entweder direkt, oder indem man das sichtbare Taktart-Symbol verändert und zusätzlich die Notendauern skaliert.

Systeme mit unterschiedlichen Taktarten, gleiche Taktlänge

Diese Art der Notation kann erstellt werden, indem für jedes System eine identische Taktart eingestellt wird, aber manuell für jeden Takt durch Einstellung von `timeSignatureFraction` auf den gewünschten Bruch geändert und dann die Länge der Noten entsprechenden skaliert wird, siehe auch [Taktangabe], Seite 58. Die Skalierung geschieht mit dem Befehl `\scaleDurations`, der auf ähnliche Weise wie `\times` benutzt wird, aber keine Klammer über den Noten ausgibt. Siehe auch [Tondauern skalieren], Seite 47.

In diesem Beispiel werden Noten mit den Taktarten 3/4, 9/8 und 10/8 parallel benutzt. Im zweiten System werden die gezeigten Dauern mit 2/3 multipliziert, da 2/3x9/8=3/4, und im dritten System werden die gezeigten Dauern mit 3/5 multipliziert, da 3/5x10/8=3/4. Oft wird es nötig sein, Balken manuell zu setzen, weil die Skalierung sich auch auf die automatische Bebalkung auswirkt.

```
\relative c' <<
  \new Staff {
    \time 3/4
    c4 c c |
    c c c |
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = #'(9 . 8)
    \scaleDurations #'(2 . 3)
    \repeat unfold 6 { c8[ c c] }
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = #'(10 . 8)
    \scaleDurations #'(3 . 5) {
      \repeat unfold 2 { c8[ c c] }
      \repeat unfold 2 { c8[ c] } |
      c4. c4. \times 2/3 { c8[ c c] } c4
    }
  }
>>
```



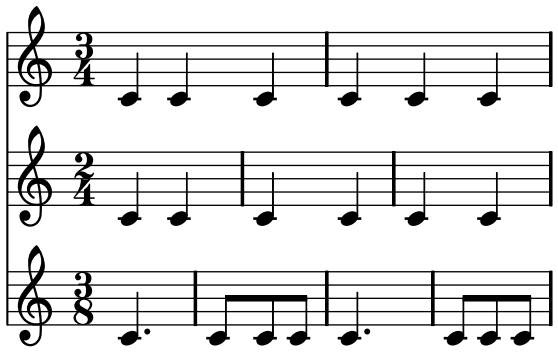
Systeme mit unterschiedlichen Taktarten, unterschiedliche Taktlänge

Jedes System kann auch eine eigene unabhängige Taktart erhalten. Dazu muss der `Timing_translator` und der `Default_bar_line_engraver` in den `Staff`-Kontext verschoben werden.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
}

% Now each staff has its own time signature.
```

```
\relative c' <<
  \new Staff {
    \time 3/4
    c4 c c |
    c4 c c |
  }
  \new Staff {
    \time 2/4
    c4 c |
    c4 c |
    c4 c |
  }
  \new Staff {
    \time 3/8
    c4. |
    c8 c c |
    c4. |
    c8 c c |
  }
>>
```



Ausgewählte Schnipsel
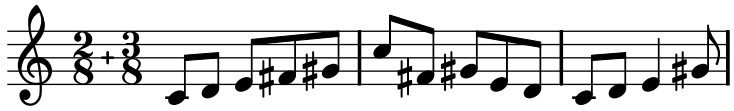
Zusammengesetzte Taktarten

Ungerade Taktarten werden (wie etwa "5/8") werden oft als zusammengesetzte Taktarten interpretiert (bspw. "3/8 + 2/8"), in welchen zwei oder mehr Teiltakte unterschieden werden. LilyPond kann derartige Noten produzieren, indem entsprechende Taktarten gesetzt werden und die automatische Bebalkung angepasst wird.

```
#(define ((compound-time one two num) grob)
   (grob-interpret-markup grob
                          (markup #:override '(baseline-skip . 0) #:number
                                  (#:line ((#:column (one num))
                                           #:vcenter "+"
                                           (#:column (two num)))))))

\relative c' {
  \override Staff.TimeSignature #'stencil = #(compound-time "2" "3" "8")
  \time 5/8
  \set Staff.beatStructure = #'(2 3)
  c8 d e fis gis
```

```
  c8 fis, gis e d
  c8 d e4 gis8
}
```



Siehe auch

Glossar: Abschnitt “polymetric” in *Glossar*, Abschnitt “polymetric time signature” in *Glossar*, Abschnitt “meter” in *Glossar*.

Notationsreferenz: [Taktangabe], Seite 58, [Tondauern skalieren], Seite 47.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “TimeSignature” in *Referenz der Interna*, Abschnitt “Timing_translator” in *Referenz der Interna*, Abschnitt “Default_bar_line_engraver” in *Referenz der Interna*, Abschnitt “Staff” in *Referenz der Interna*.

Bekannte Probleme und Warnungen
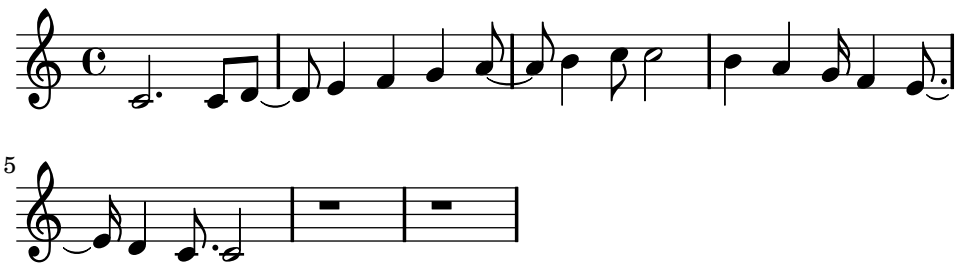
Wenn unterschiedliche Taktarten parallel benutzt werden, werden Noten auf demselben musikalischen Moment horizontal auf die gleiche Position gesetzt. Die unterschiedlichen Taktlinien führen allerdings dazu, dass die Noten nicht ganz so regelmäßig gesetzt werden, wie es ohne unterschiedliche Taktarten der Fall wäre.

Automatische Aufteilung von Noten

Lange Noten, die über Taktlinien hinüberreichen, können automatisch in übergebundene Noten aufgeteilt werden. Dieses Verhalten erreicht man, indem der Abschnitt “Note_heads_engraver” in *Referenz der Interna* mit dem Abschnitt “Completion_heads_engraver” in *Referenz der Interna* ausgetauscht wird. Auf gleiche Art können lange Pausen, die über Taktgrenzen reichen, automatisch aufgeteilt werden, indem man den `Rest_engraver`mit dem `Completion_rest_engraver` ersetzt. Im nächsten Beispiel werden Noten und Pausen, die über die Taktlinie dauern, aufgeteilt; Noten werden auch übergebunden.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
  \remove "Rest_engraver"
  \consists "Completion_rest_engraver"
}

{ c2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2 }
```



Dieser Engraver teilt alle Noten und Pausen auf, die über eine Taktlinie dauern und fügt für Noten Bindebögen hinzu. Er kann unter Anderem dann nützlich sein, wenn man komplexe

Partituren auf Fehler überprüfen möchte: Wenn die Takte nicht vollständig gefüllt sind, zeigt die Überbindung genau an, wie viele Notenwerte noch in dem jeweiligen Takt fehlen.

Siehe auch

Glossar: Abschnitt “tie” in *Glossar*

Handbuch zum Lernen: Abschnitt “Was sind Engraver?” in *Handbuch zum Lernen*, Abschnitt “Engraver hinzufügen und entfernen” in *Handbuch zum Lernen*.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Note_heads_engraver” in *Referenz der Interna*, Abschnitt “Completion_heads_engraver” in *Referenz der Interna*, Abschnitt “Rest_engraver” in *Referenz der Interna*, Abschnitt “Completion_rest_engraver” in *Referenz der Interna*. Abschnitt “Forbid_line_break_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

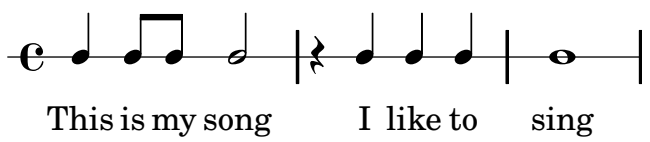
Nicht alle Notenwerte (besonders wenn sie andere rhythmische Aufteilungen beinhalten) können exakt durch normale Noten und Punktierungen wiedergegeben werden. Der Engraver setzt aber trotzdem keine Triolen etc.

`Completion_heads_engraver` wirkt sich nur auf Noten aus; Pausen werden nicht aufgeteilt.

Melodierhythmus anzeigen

Manchmal soll nur der Rhythmus einer Melodie dargestellt werden. Das erreicht man mit einem Rhythmus-Notensystem. Alle Tonhöhen werden auf eine Linie reduziert und das System hat auch nur eine einzige Linie.

```
<<
  \new RhythmicStaff {
    \new Voice = "myRhythm" {
      \time 4/4
      c4 e8 f g2
      r4 g g f
      g1
    }
  }
  \new Lyrics {
    \lyricsto "myRhythm" {
      This is my song
      I like to sing
    }
  }
>>
```
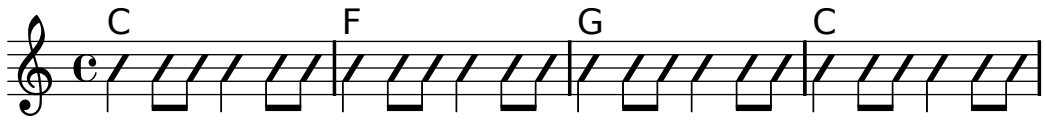


Akkordnotation für Gitarren bezeichnet auch oft zusätzlich den geschlagenen Rhythmus. Das kann notiert werden unter Verwendung des `Pitch_squash_engraver` und indem Tonhöhenimprovisation eingeschaltet wird mit `\improvisationOn`.

```
<<
  \new ChordNames {
    \chordmode {
```

```
      c1 f g c
    }
  }

  \new Voice \with {
    \consists Pitch_squash_engraver
  } \relative c'' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c
  }
>>
```



Vordefinierte Befehle

`\improvisationOn`, `\improvisationOff`.

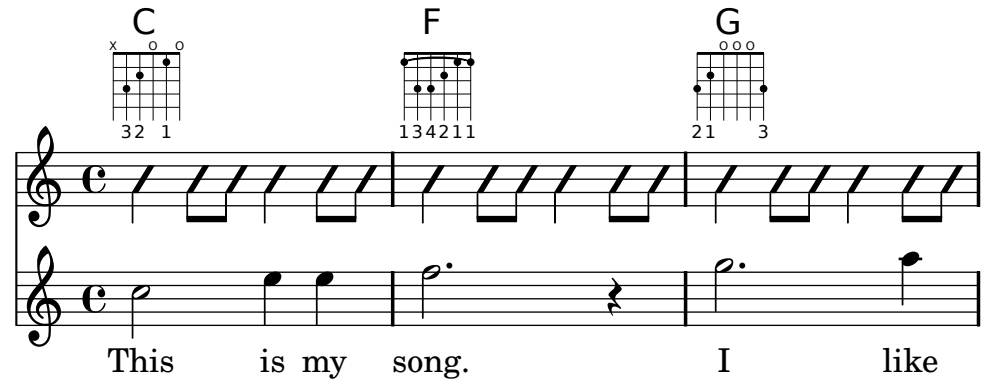
Ausgewählte Schnipsel

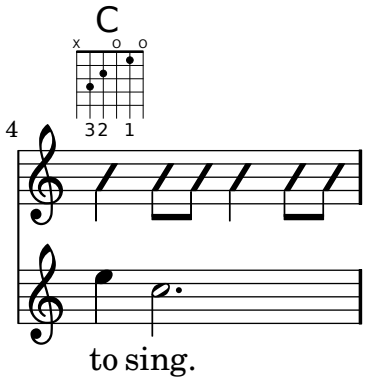
Schlagrhythmus für Gitarren

In Gitarrennotation kann neben Melodie, Akkordbezeichnungen und Bunddiagrammen auch der Schlagrhythmus angegeben werden.

```
\include "predefined-guitar-fretboards.ly"
<<
  \new ChordNames {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new FretBoards {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } {
    \relative c'' {
      \improvisationOn
      c4 c8 c c4 c8 c
      f4 f8 f f4 f8 f
      g4 g8 g g4 g8 g
      c4 c8 c c4 c8 c
    }
  }
  \new Voice = "melody" {
```

```
    \relative c'' {
      c2 e4 e4
      f2. r4
      g2. a4
      e4 c2.
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      This is my song.
      I like to sing.
    }
  }
>>
```





Siehe auch

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.
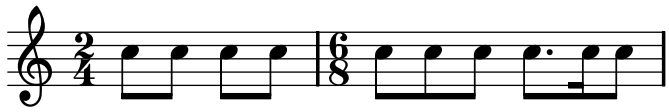
Referenz der Interna: Abschnitt “RhythmicStaff” in *Referenz der Interna*, Abschnitt “Pitch_squash_engraver” in *Referenz der Interna*.

1.2.4 Balken

Automatische Balken

LilyPond setzt Balken (engl. beam) automatisch.
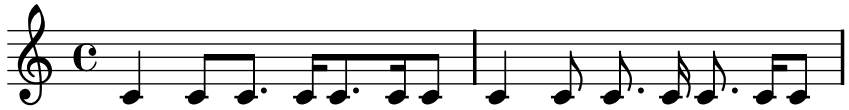
```
\time 2/4 c8 c c c
\time 6/8 c8 c c c8. c16 c8
```



Wenn diese automatischen Entscheidungen nicht gut genug sind, können die Balken auch explizit eingegeben werden, siehe [Manuelle Balken], Seite 84. Balken *müssen* auch auf diese Weise eingegeben werden, wenn sie über Pausen hinwegreichen sollen.

Wenn automatische Bebalkung nicht benötigt wird, kann sie mit dem Befehl `\autoBeamOff` aufgehoben werden und mit dem Befehl `\autoBeamOn` wieder eingeschaltet werden.

```
c4 c8 c8. c16 c8. c16 c8
\autoBeamOff
c4 c8 c8. c16 c8.
\autoBeamOn
c16 c8
```



Achtung: Wenn Balken eingesetzt werden, um Melismen in Gesang zu notieren, sollte die automatische Bebalkung mit `\autoBeamOff` ausgeschaltet werden und die Balken manuell notiert werden.

Achtung: Wenn `\partcombine` mit `\autoBeamOff` benutzt wird, können ungewünschte Resultate auftreten. Siehe die Schnipsel unten für mehr Information.

Balkenmuster, die sich von den automatisch erstellen unterscheiden, können erstellt werden, siehe [Einstellung von automatischen Balken], Seite 77.

Vordefinierte Befehle

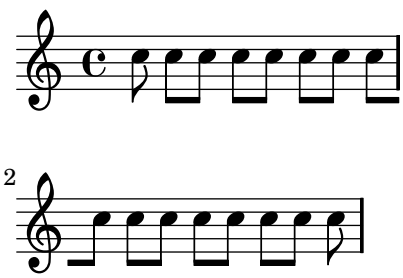
`\autoBeamOff`, `\autoBeamOn`.

Ausgewählte Schnipsel

Balken über Zeilenumbrüche

Zeilenumbrüche sind normalerweise während Balken verboten. Das kann geändert werden.

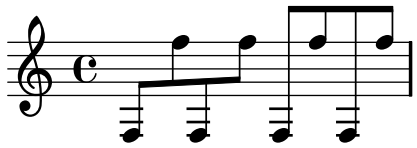
```
\relative c'' {
  \override Beam #'breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
  c8] c[ c] c[ c] c[ c] c
}
```



Balken für weit auseinander liegende Noten ändern

Balken mit Hälsen in unterschiedliche Richtungen werden automatisch erstellt, wenn ein großer Sprung zwischen Tonhöhen gefunden wird. Dieses Verhalten kann durch die `auto-knee-gap`-Eigenschaft beeinflusst werden. Ein derartiger Knie-Balken wird erstellt, wenn der Abstand größer ist als der Wert von `auto-knee-gap` plus der Dicke des Balkens (was von der Notendauer und der Neigung des Balkens abhängt). Der Standardwert von `auto-knee-gap` ist 5.5 Notensystemabstände.

```
{
  f8 f''8 f8 f''8
  \override Beam #'auto-knee-gap = #6
  f8 f''8 f8 f''8
}
```



Partcombine und autoBeamOff

Die Funktionsweise von `\autoBeamOff`, wenn es zusammen mit `\partcombine` eingesetzt wird, kann schwer zu verstehen sein. Es kann besser sein, anstatt dessen

```
\set Staff.autobeaming = ##f
```
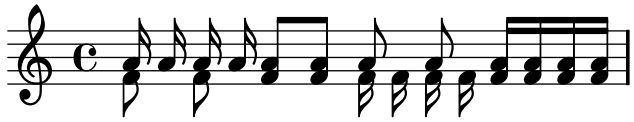
zu benutzen, um sicherzustellen, dass die automatische Bebalkung für das gesamte System ausgeschaltet ist.

`\partcombine` funktioniert offensichtlich mit 3 Stimme (Hals nach oben einfach, Hals nach unten einfach, Hals nach oben kombiniert).

Ein `\autoBeamOff`-Befehl im ersten Argument von `\partcombine` gilt für die Stimme, die zu dem Zeitpunkt aktiv ist, an dem der Befehl verarbeitet wird, entweder für Hals nach oben, nach unten oder Hals nach oben kombiniert. Ein `\autoBeamOff`-Befehl im zweiten Argument gilt für die Stimme, die mit Hals nach unten einfach ist.

Um `\autoBeamOff` zu benutzen, damit alle automatischen Balken aufhören, wenn man es mit `\partcombine` verwendet, muss `\autoBeamOff` *dreimal* aufgerufen werden.

```
{
  %\set Staff.autoBeaming = ##f % turns off all autobeaming
  \partcombine
  {
    \autoBeamOff % applies to split up stems
    \repeat unfold 4 a'16
    %\autoBeamOff % applies to combined up stems
    \repeat unfold 4 a'8
    \repeat unfold 4 a'16
  }
  {
    \autoBeamOff % applies to down stems
    \repeat unfold 4 f'8
    \repeat unfold 8 f'16 |
  }
}
```



Siehe auch

Notationsreferenz: [Manuelle Balken], Seite 84, [Einstellung von automatischen Balken], Seite 77.

Installierte Dateien: ‘`scm/auto-beam.scm`’.
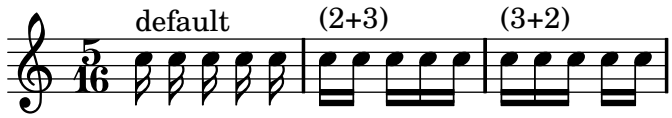
Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Auto_beam_engraver” in *Referenz der Interna*, Abschnitt “Beam_engraver” in *Referenz der Interna*, Abschnitt “Beam” in *Referenz der Interna*, Abschnitt “BeamEvent” in *Referenz der Interna*, Abschnitt “BeamForbidEvent” in *Referenz der Interna*, Abschnitt “beam-interface” in *Referenz der Interna*, Abschnitt “unbreakable-spanner-interface” in *Referenz der Interna*.

Einstellung von automatischen Balken

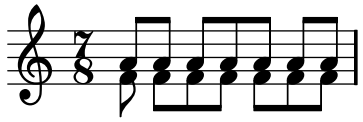
In den meisten Fällen enden automatische Balken am Ende eines Taktes. Die Endpunkte für Schläge werden durch die Kontexteigenschaften `baseMoment` und `beatStructure` bestimmt. `beatStructure` ist eine Scheme-Liste, die die Länge jedes Schlages im Takt in Einheiten von `baseMoment` angibt. Der Standard von `baseMoment` ist Eins durch den Numerator der Taktangabe. Der Standardwert jeder Längeneinheit `baseMoment` ist ein einzelner Taktschlag.

```
\time 5/16
c16^"default" c c c c |
\set Timing.beatStructure = #'(2 3)
c16^"(2+3)" c c c c |
\set Timing.beatStructure = #'(3 2)
c16^"(3+2)" c c c c |
```



Balkenregelveränderungen können auf bestimmte Kontexte beschränkt werden. Wenn keine Regeln in einen unteren Kontext definiert sind, gelten die Regeln des höheren Kontext, in dem sich der niedrigere befindet.
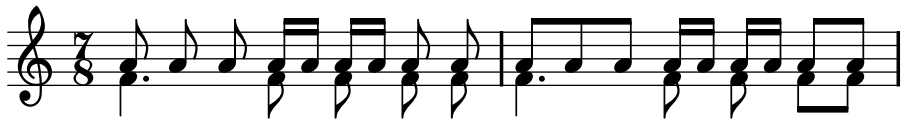
```
\new Staff <<
  \time 7/8
  \set Staff.beatStructure = #'(2 3 2)
  \new Voice = one {
    \relative c'' {
        a8 a a a a a a
    }
  }
  \new Voice  = two {
    \relative c' {
      \voiceTwo
      \set Voice.beatStructure = #'(1 3 3)
      f8 f f f f f f
    }
  }
>>
```



Wenn mehrere Stimmen eingesetzt werden, muss der `Staff`-Kontext definiert werden, wenn die Balkenregeln auf alle Stimmen des Systems angewendet werden sollen:

```
\time 7/8
% rhythm 3-1-1-2
% Context applied to Voice by dafault --  does not work correctly
% Because of autogenerated voices, all beating will
% be at baseMoment (1 . 8)
\set beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \\ {f4. f8 f f f} >>

% Works correctly with context Staff specified
\set Staff.beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \\ {f4. f8 f f f} >>
```



Der Wert von `baseMoment` kann angepasst werden, um das Bebalkungsverhalten zu ändern, wenn gewünscht. In diesem Fall muss der Wert von `beatStructure` so gesetzt werden, dass er kompatibel mit dem neuen Wert von `baseMoment` ist.

```
\time 5/8
\set Timing.baseMoment = #(ly:make-moment 1 16)
\set Timing.beatStructure = #'(7 3)
\repeat unfold 10 { a16 }
```
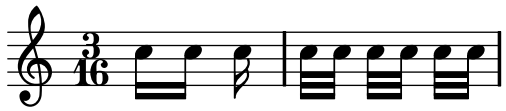


`baseMoment` ist ein *Moment*, eine Einheit an musikalischer Dauer. Eine Anzahl vom Typus *Moment* wird durch die Scheme-Funktion `ly:make-moment` erstellt. Zu mehr Information über diese Funktion siehe [Verwaltung der Zeiteinheiten], Seite 105.

Der Standardwert von `baseMoment` ist Eins durch den Denominator der Taktangabe. Alle Ausnahmen dieses Standards finden sich in der Datei '`scm/time-signature-settings.scm`'.

Besondere automatische Bebalkungsregeln (außer dass ein Balken auf einem Taktschlag aufhört) sind in der `beamExceptions`-Eigenschaft definiert.

```
\time 3/16
\set Timing.beatStructure = #'(2 1)
\set Timing.beamExceptions =
  #'(                          ;start of alist
     (end .                    ;entry for end of beams
      (                        ;start of alist of end points
       ((1 . 32) . (2 2 2))    ;rule for 1/32 beams -- end each 1/16
      )))                      %close all entries
c16 c c |
\repeat unfold 6 { c32 } |
```



`beamExceptions` ist eine Aliste mit einem Schlüssel der Regeltypen (rule-type) und einem Wert der Bebalkungsregeln (beaming-rules).

Im Moment ist der einzige mögliche rule-type `#'end` für ein Balkenende.

Beaming-rules ist eine Scheme-Aliste (oder eine paarige Liste), die den Balkentyp und die Gruppierung anzeigt, die auf Balken angewendet werden, welche Noten mit einer kürzesten Dauer des Balkentyps enthalten.

```
#'((beam-type1 . grouping-1)
   (beam-type2 . grouping-2)
   (beam-type3 . grouping-3))
```

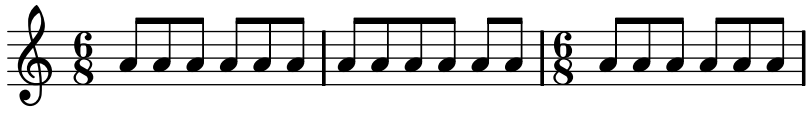
Beam-type ist ein Scheme-Paar, das die Dauer eines Balkens anzeigt, etwa `(1 . 16)` für ein Sechszehntel.

Grouping ist eine Scheme-Liste, die die auf den Balken anzuwendene Gruppierung anzeigt. Die Gruppierung wird in Einheiten des Balkentyps angegeben.

Achtung: Ein `beamExceptions`-Wert muss eine *vollständige* Ausnahme-Liste sein. Das heißt, dass jede Ausnahme, die angewendet werden soll, auch in die Einstellungen mit aufgenommen werden muss. Es ist nicht möglich, nur eine der Einstellungen zu ändern, zu entfernen oder hinzuzufügen. Das mag seltsam erscheinen, bedeutet aber, dass die aktuellen Balkenregeln bekann sein müssen, um ein neues Bebalkungsmuster definieren zu können.

Wenn die Taktart geändert wird, werden neue Standardwerte für `Timing.baseMoment`, `Timing.beatStructure` und `Timing.beamExceptions` definiert. Wenn die Taktart definiert wird, werden die automatischen Bebalkungsregeln für den `Timing`-Kontext auf den Standard zurückgesetzt.

```
\time 6/8
\repeat unfold 6 { a8 }
% group (4 + 2)
\set Timing.beatStructure = #'(4 2)
\repeat unfold 6 { a8 }
% go back to default behavior
\time 6/8
\repeat unfold 6 { a8 }
```
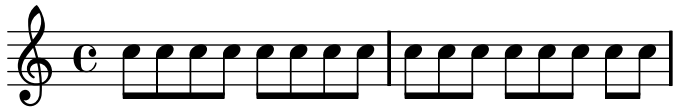


Die automatischen Standardeinstellungen für die Bebalkung einer Taktart werden in der Datei '`scm/time-signature-settings.scm`' bestimmt. Änderungen der automatischen Bebalkungsregeln für eine Taktart werden in [Taktangabe], Seite 58 beschrieben.

Die meisten automatischen Bebalkungsregeln für eine Taktart enthalten einen Eintrag für `beamExceptions`. Beispielsweise wird in einem 4/4-Takt versucht, den Takt in zwei Hälften zu teilen, wenn nur Achtelnoten vorkommen. Die `beamExceptions`-Regel kann die `beatStructure`-Einstellung überschreiben, wenn `beamExceptions` nicht zurückgesetzt wird:

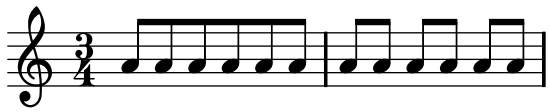
```
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1 8)
\set Timing.beatStructure = #'(3 3 2)
% This won't beam (3 3 2) because of beamExceptions
```

```
\repeat unfold 8 {c8} |
% This will beam (3 3 2) because we clear beamExceptions
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c8}
```



Auf die gleiche Art werden im 3/4-Takt Achtelnoten als ganzer Takt bebalkt. Um Achtelnoten im 3/4-Takt auf jeder Taktzeit zu bebalken, muss `beamExceptions` zurückgesetzt werden:

```
\time 3/4
% by default we beam in (3) due to beamExceptions
\repeat unfold 6 {a8} |
% This will beam (1 1 1) due to beatLength
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a8}
```



Wie die automatische Bebalkung funktioniert

Wenn die automatische Bebalkung aktiviert ist, wird die Platzierung der automatischen Balken durch die Kontexteigenschaften `baseMoment`, `beatStructure` und `beamExceptions` bestimmt.

Die folgenden Regeln, in der Reihenfolge ihrer Priorität, gelten, wenn das Aussehen der Balken bestimmt wird:

- Wenn ein manueller Balken mit `[...]` definiert ist, wird er gesetzt, andernfalls
- wenn eine Balkenendung-Regel für den Balkentyp in `beamExceptions` definiert ist, wird sie verwendet, um die gültigen Plätze für Balkenenden zu berechnen, andernfalls
- wenn eine Balkenendung-Regel für einen größeren Balkentyp in `beamExceptions` definiert ist, wird sie verwendet, um die gültigen Plätze für Balkenenden zu berechnen, andernfalls
- benutze die Werte von `baseMoment` und `beatStructure`, um die Enden der Balken im Takt zu definieren und beende Balken am Ende jedes Taktes.

In den oben genannten Regeln ist der Balkentyp die Dauer der kürzesten Note der bebalkten Gruppe.

Die Standardbebalkungsregeln finden sich in der Datei '`scm/time-signature-settings.scm`'.

Ausgewählte Schnipsel
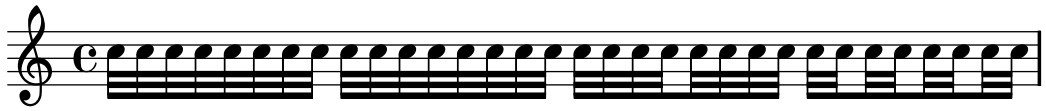
Subdividing beams

The beams of consecutive 16th (or shorter) notes are, by default, not subdivided. That is, the three (or more) beams stretch unbroken over entire groups of notes. This behavior can be modified to subdivide the beams into sub-groups by setting the property `subdivideBeams`. When set, multiple beams will be subdivided at intervals defined by the current value of `baseMoment` by reducing the multiple beams to just one beam between the sub-groups. Note that `baseMoment` defaults to one over the denominator of the current time signature if not set explicitly. It must be set to a fraction giving the duration of the beam sub-group using the `ly:make-moment` function, as shown in this snippet. Also, when `baseMoment` is changed, `beatStructure` should also be changed to match the new `baseMoment`:

```
\relative c'' {
  c32[ c c c c c c c]
  \set subdivideBeams = ##t
  c32[ c c c c c c c]

  % Set beam sub-group length to an eighth note
  \set baseMoment = #(ly:make-moment 1 8)
  \set beatStructure = #'(2 2 2 2)
  c32[ c c c c c c c]

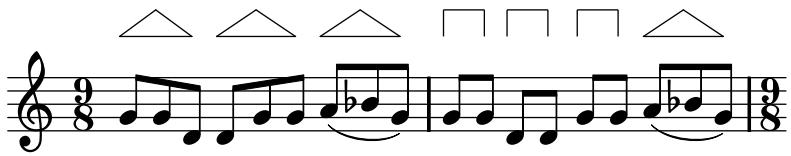
  % Set beam sub-group length to a sixteenth note
  \set baseMoment = #(ly:make-moment 1 16)
  \set beatStructure = #'(4 4 4 4)
  c32[ c c c c c c c]
}
```

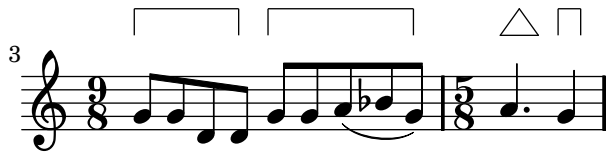


Dirigierzeichen Taktgruppenzeichen

Optionen, mit denen die Balken in einem Takt gruppiert werden, sind durch die Scheme-Funktion `set-time-signature` erhältlich, die drei Argumente braucht: Die Zahl der Taktschläge, die Länge des Schlages und die interne gruppieren von Balken in dem Takt. Wenn der `Measure_grouping_engraver` hinzugefügt worden ist, erstellt diese Funktion auch `MeasureGrouping`-(Taktgruppen)-Zeichen. Derartige Zeichen erleichtern das Lesen von rhythmisch komplexer Musik. In dem Beispiel ist der 9/8-Takt in 2, 2, 2 und 3 aufgeteilt. Das wird der `set-time-signature`-Funktion als das dritte Argument mitgegeben: `'(2 2 2 3)`:

```
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = #'(2 2 2 3)
    g8 g d d g g a( bes g) |
    #(set-time-signature 9 8 '(4 5))
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```
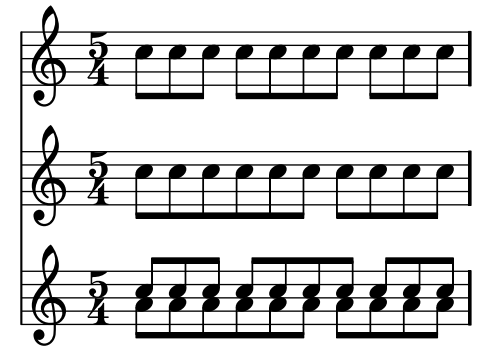




Balkenenden auf Score-Ebene

Balkenenderegeln, die im `Score`-Kontext definiert werden, wirken sich auf alle Systeme aus, können aber auf `Staff`- und `Voice`-Ebene neu verändert werden:

```
\relative c'' {
  \time 5/4
  % Set default beaming for all staves
  \set Score.baseMoment = #(ly:make-moment 1 8)
  \set Score.beatStructure = #'(3 4 3)
  <<
    \new Staff {
      c8 c c c c c c c c c
    }
    \new Staff {
      % Modify beaming for just this staff
      \set Staff.beatStructure = #'(6 4)
      c8 c c c c c c c c c
    }
    \new Staff {
      % Inherit beaming from Score context
      <<
        {
          \voiceOne
          c8 c c c c c c c c c
        }
        % Modify beaming for this voice only
        \new Voice {
          \voiceTwo
          \set Voice.beatStructure = #'(6 4)
          a8 a a a a a a a a a
        }
      >>
    }
  >>
}
```



Siehe auch

Installierte Dateien: ‘`scm/beam-settings.scm`’.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

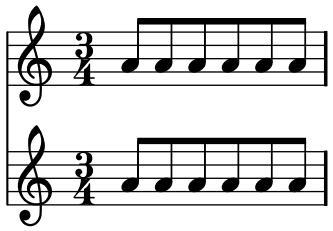
Referenz der Interna: Abschnitt “Auto_beam_engraver” in *Referenz der Interna*, Abschnitt “Beam” in *Referenz der Interna*, Abschnitt “BeamForbidEvent” in *Referenz der Interna*, Abschnitt “beam-interface” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Wenn eine Partitur endet, während ein automatischer Balken noch nicht beendet wurde und weiterhin Noten erwartet, wird dieser letzte Balken nicht ausgegeben. Das Gleiche gilt auch für polyphone Stimmen, die mit der `<< ... \\ ... >>`-Konstruktion notiert wurden. Wenn eine polyphone Stimme endet, während ein Balken noch weitere Noten erwartet, wird der Balken nicht gesetzt. Eine Notlösung für dieses Probelm ist, den letzten Balken in der Stimme oder Partitur manuell zu setzen.

Die Standardeinstellungen weisen den dem `Score`-Kontext zu. Das bedeutet, dass das Setzen der Taktart (time signature) in einem System sich auch auf die Bebalkung der anderen Systeme auswirkt. Darum verändert eine neue Taktart in einem späteren System auch alle früher vorgenommenen eigenen Einstellungen der Bebalkung eines anderen Systems. Eine Möglichkeit, dieses Problem zu vermeiden, ist es, die Taktart nur in einem System zu setzen.

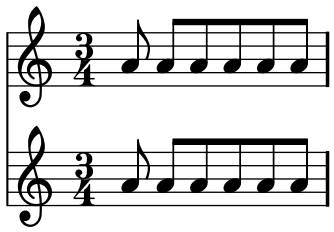
```
<<
  \new Staff {
    \time 3/4
    \set Timing.baseMoment = #(ly:make-moment 1 8)
    \set Timing.beatStructure = #'(1 5)
    \repeat unfold 6 { a8 }
  }
  \new Staff {
    \repeat unfold 6 { a8 }
  }
>>
```



Die Standardbebalkungsregeln für die Taktart können auch verändert werden, sodass die gewünschte Bebalkung immer benützt wird. Veränderungen der automatischen Bebalkungsregeln für eine Taktart sind in [Taktangabe], Seite 58 beschrieben.

```
<<
  \new Staff {
    \overrideTimeSignatureSettings
      #'(3 . 4)         % timeSignatureFraction
      #'(1 . 8)         % baseMomentFraction
      #'(1 5)           % beatStructure
      #'()              % beamExceptions
    \time 3/4
    \repeat unfold 6 { a8 }
  }
  \new Staff {
    \time 3/4
```
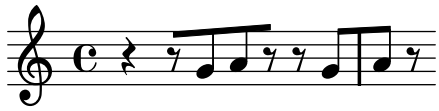
```
    \repeat unfold 6 { a8 }
  }
>>
```



Manuelle Balken

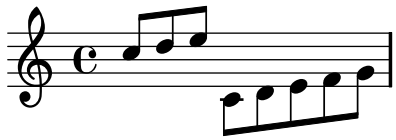
In einigen Fällen kann es nötig sein, den automatischen Algorithmus für die Balken zu überschreiben. Die automatischen Balken werden beispielsweise nicht über Pausen oder Taktlinien hinweg gesetzt, und in Gesang werden die Balken oft nach dem Rhythmus des Textes und nicht dem der Musik gesetzt. Manuell definierte Balken werden mit den Zeichen `[` und `]` (AltGr+8 bzw. 9) markiert.

```
r4 r8[ g' a r] r g[ | a] r
```
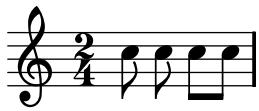


Die Richtung von Balken kann mit den Richtungszeichen verändert werden:

```
c8^[ d e] c,_[ d e f g]
```
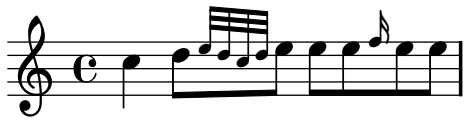


Einzelne Noten können mit dem Befehl `\noBeam` markiert werden, damit sie nicht mit einem Balken versehen werden.

```
\time 2/4 c8 c\noBeam c c
```
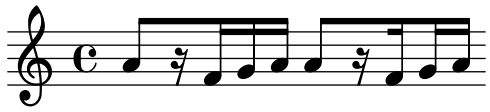


Balken von Verzierungsnoten und normale Balken können gleichzeitig gesetzt werden. Unbebalkte Verzierungen werden nicht innerhalb von normalen Balken gesetzt.

```
c4 d8[
\grace { e32[ d c d] }
e8] e[ e
\grace { f16 }
e8 e]
```



Noch bessere manuelle Kontrolle über die Balken kann durch Setzen der Eigenschaften `stemLeftBeamCount` und `stemRightBeamCount` erreicht werden. Sie bestimmen die Anzahl von Balken, die rechts und links vom Hals der nächsten Note gesetzt werden sollen. Wenn eine Eigenschaften gesetzt ist, wird ihr Wert nur einmal eingesetzt und dann wieder auf Null gesetzt. Im folgenden Beispiel hat das letzte `f` nur einen Balken an seiner linken Seite (der als Achtelbalken der gesamten Gruppe gewertet wird).

```
a8[ r16 f g a]
a8[ r16
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #1
f16
\set stemLeftBeamCount = #1
g16 a]
```



Ausgewählte Schnipsel

Gerade Fähnchen und überstehende Balkenenden

Gerade Fähnchen an einzelnen Noten und überstehende Balkenenden bei bebalkten Notengruppen sind möglich mit einer Kombination aus `stemLeftBeamCount`, `stemRightBeamCount` und Paaren von `[]`-Balkenbegrenzungen.

Für gerade Fähnchen, die nach rechts zeigen, kann `[]` eingesetzt werden und `stemLeftBeamCount` auf Null gesetzt werden (wie Bsp. 1).

Für gerade Fähnchen, die nach links zeigen, muss `stemRightBeamCount` eingesetzt werden (Bsp. 2).

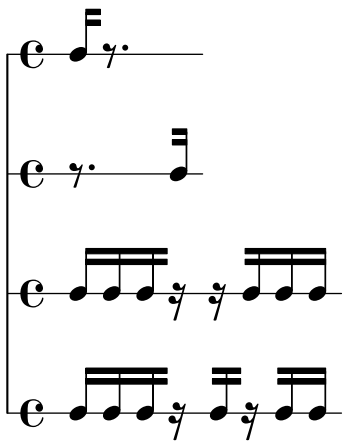
Für überstehende Balkenenden nach rechts muss `stemRightBeamCount` auf einen positiven Wert gesetzt werden, für Balkenenden, die nach links zeigen benutzt man `stemLeftBeamCount` (Bsp. 3).

Manchmal können einzelne Noten, die von Pausen umgeben sind, auch Balkenenden in beide Richtungen tragen. Das geschieht mit `[]`-Klammern (Bsp. 4).

(`\set stemLeftBeamCount` entspricht immer dem Befehl `\once \set`. Anders gesagt müssen die Einstellungen immer wieder wiederholt werden und die Fähnchen des letzten Sechzehntels im letzten Beispiel haben nichts mit dem `\set`-Befehl zwei Noten vorher zu tun.)

```
\score {
  <<
    % Example 1
    \new RhythmicStaff {
      \set stemLeftBeamCount = #0
      c16[]
      r8.
    }
    % Example 2
    \new RhythmicStaff {
      r8.
      \set stemRightBeamCount = #0
      c16[]
    }
```

```
    % Example 3
    \new RhythmicStaff {
      c16 c
      \set stemRightBeamCount = #2
      c16 r r
      \set stemLeftBeamCount = #2
      c16 c c
    }
    % Example 4
    \new RhythmicStaff {
      c16 c
      \set stemRightBeamCount = #2
      c16 r
      c16[]
      r16
      \set stemLeftBeamCount = #2
      c16 c
    }
  >>
}
```



Siehe auch

Notationsreferenz: Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538, [Verzierungen], Seite 99.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Beam” in *Referenz der Interna*, Abschnitt “BeamEvent” in *Referenz der Interna*, Abschnitt “Beam_engraver” in *Referenz der Interna*, Abschnitt “beam-interface” in *Referenz der Interna*, Abschnitt “Stem_engraver” in *Referenz der Interna*.

Gespreizte Balken
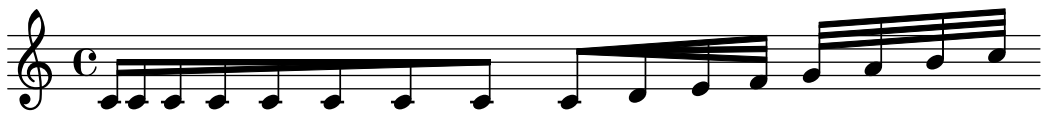
Gespreizte Balken werden teilweise eingesetzt um anzuzeigen, dass kleine Notengruppen in beschleunigendem oder verlangsamendem Tempo gespielt werden sollen, ohne dass sich das Tempo des Stückes verändert. Die Reichweite der gespreizten Balken muss manuell mit `[` und `]` angegeben werden und die Spreizung wird kontrolliert, indem der Balken-Eigenschaft `grow-direction` eine Richtung zugewiesen wird.

Wenn die Anordnung der Noten und die MIDI-Ausgabe das *Ritardando* oder *Accelerando*, wie es die Spreizung angibt, reflektieren soll, müssen die Noten als ein musikalischer Ausdruck notiert werden, der von geschweiften Klammern umgeben ist und dem ein `featheredDurations`-(gespreizteDauern)-Befehl vorangestellt ist, der das Verhältnis der ersten und letzten Dauer definiert.

Die eckigen Klammern geben die Reichweite des Balkens an und die geschweiften Klammern zeigen, auf welche Noten sich die Veränderung der Dauern auswirkt. Normalerweise bezieht sich das auf die selbe Notengruppe, aber das ist nicht unbedingt erforderlich: beide Befehle sind unabhängig voneinander.

Im folgenden Beispiel nehmen die acht 16-Noten exakt die gleiche Zeit ein wie eine halbe Note, aber die erste Note ist halb so lang wie die letzte der Gruppe, und die Noten dazwischen werden stufenweise verlängert. Die ersten vier 32-Noten beschleunigen stufenweise das Tempo, während die darauffolgenden vier 32-Noten ein gleichmäßiges Tempo haben.

```
\override Beam #'grow-direction = #LEFT
\featherDurations #(ly:make-moment 2 1)
{ c16[ c c c c c c c] }
\override Beam #'grow-direction = #RIGHT
\featherDurations #(ly:make-moment 2 3)
{ c32[ d e f] }
% revert to non-feathered beams
\override Beam #'grow-direction = #'()
{ g32[ a b c] }
```



Die Platzierung der Noten im Druckbild entspricht den Notendauern nur annähernd, aber die MIDI-Ausgabe ist exakt.

Vordefinierte Befehle

`\featherDurations`.

Siehe auch

Snippets: Abschnitt “Rhythms” in *Schnipsel*.

Bekannte Probleme und Warnungen

Der `\featherDurations`-Befehl funktioniert nur mit kurzen Notenabschnitten, und wenn die Zahlen in den Brüchen klein sind.
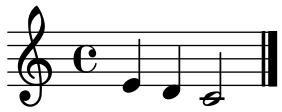
1.2.5 Takte

Taktstriche

Taktstriche trennen Takte voneinander, werden aber auch verwendet, um Wiederholungen anzuzeigen. Normalerweise werden sie automatisch nach Vorgabe der aktuellen Taktart eingefügt.

Die einfachen, automatisch eingefügten Taktstriche können mit dem `\bar`-Befehl geändert werden. Eine doppelter Taktstrich etwa wird normalerweise am Ende eines Stückes gesetzt:

```
e4 d c2 \bar "|."
```



Es ist kein Fehler, wenn die letzte Note in einem Takt nicht zum automatisch eingefügten Taktstrich aufhört: es wird angenommen, dass die Note im nächsten Takt weitergeht. Wenn aber eine ganze Reihe solcher überlappenden Takte auftritt, können die Noten gedrungen aussehen oder sogar über den Seitenrand hinausragen. Das kommt daher, dass Zeilenumbrüche nur dann vorgenommen werden, wenn ein vollständiger Takt auftritt, also ein Takt, an dem alle Noten vor dem Taktstrich zu Ende sind.

Achtung: Eine falsche Dauer kann bewirken, dass Zeilenumbrüche verhindert werden, woraus resultiert, dass die Noten entweder sehr stark gedrängt auf der Zeile notiert werden, oder die Zeile über den Seitenrand hinausragt.

Zeilenumbrüche werden erlaubt, wenn ein Taktstrich manuell eingefügt wird, auch, wenn es sich um keinen vollständigen Takt handelt. Um einen Zeilenumbruch zu erlauben, ohne den Taktstrich auszugeben, kann

```
\bar ""
```

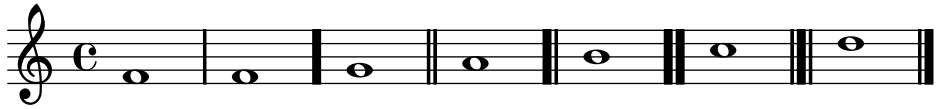
benutzt werden. Damit wird ein unsichtbarer Taktstrich an dieser Stelle eingefügt und damit ein Zeilenumbruch erlaubt (aber nicht erzwungen), ohne dass sich die Anzahl der Takte erhöhen würde. Um einen Zeilenumbruch zu erzwingen, siehe Abschnitt 4.3.1 [Zeilenumbrüche], Seite 473.
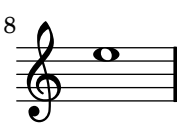
Diese Taktstrichart und auch andere besondere Taktstriche können manuell an jeder Stelle in der Partitur eingefügt werden. Wenn sie mit dem Ende eines Taktes übereinstimmen, wird der automatische Taktstrich durch den manuellen ersetzt. Diese manuellen Einfügungen haben keine Auswirkung auf die Zählung und Position der folgenden automatischen Taktstriche.

Dabe gilt zu beachten, dass manuell gesetzten Taktstriche nur visuell sichtbar sind. Sie wirken sich auf keine der Eigenschaften aus, die ein normaler Taktstrich beeinflussen würde, wie etwa Taktzahlen, Versetzungszeichen, Zeilenumbrüche usw. Sie beeinflussen auch nicht die Berechnung und Platzierung von weiteren automatischen Taktstrichen. Wenn ein manueller Taktstrich dort gesetzt wird, wo ein automatischer Taktstrich sowieso wäre, werden die Auswirkungen des originalen Taktstriches nicht verändert.

Manuell können zwei einfache Taktstriche und zusätzlich fünf Arten eines doppelten Taktstriches gesetzt werden:

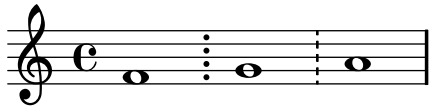
```
f1 \bar "|"
f1 \bar "."
g1 \bar "||"
a1 \bar ".|"
b1 \bar ".|."
c1 \bar "|.|"
d1 \bar "|."
e1
```





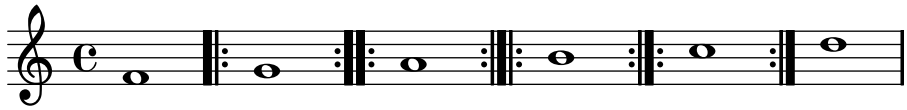
Zusätzlich gibt es noch punktierte und gestrichelte Taktstriche:

```
f1 \bar ":"
g1 \bar "dashed"
a1
```



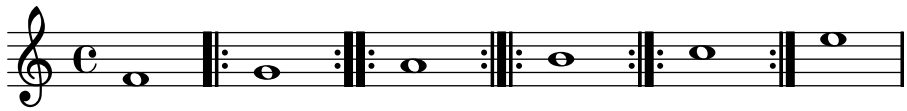
und fünf unterschiedliche Wiederholungstaktstriche:

```
f1 \bar "|:" g \bar ":|:" a \bar ":|.|:" b \bar ":|.:" c \bar ":|" d
```



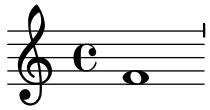
Zusätzlich kann eine Taktlinie mit einem einfachen Apostroph gesetzt werden:

```
f1 \bar "|:"
g1 \bar ":|:"
a1 \bar ":|.|:"
b1 \bar ":|.:"
c1 \bar ":|"
e1
```



Zusätzliche kann ein Taktstrich auch nur als kleines Komma gesetzt werden:

```
f1 \bar "'"
```

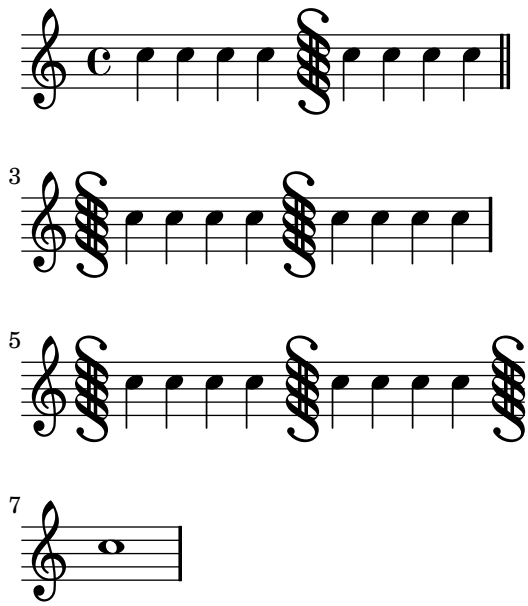


Derartige Apostrophe werden allerdings vor allem im gregorianischen Choral eingesetzt, und es wird empfohlen, anstatt dessen `\divisioMinima` zu benutzen, wie beschrieben im Abschnitt [Divisiones], Seite 405.

Für *segno*-Zeichen innerhalb des Systems gibt es drei Taktstricharten, die sich in ihrem Verhalten an Zeilenumbrüchen unterscheiden:

```
c4 c c c
\bar "S"
c4 c c c \break
\bar "S"
c4 c c c
\bar "|S"
c4 c c c \break
\bar "|S"
c4 c c c
\bar "S|"
c4 c c c \break
\bar "S|"
```

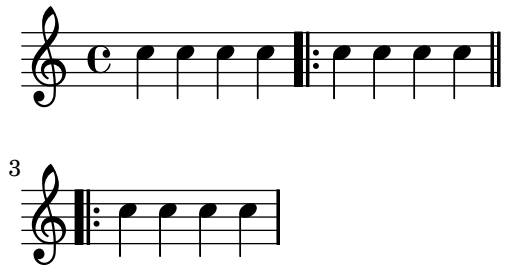
```
c1
```



Auch wenn die Taktlinien, die Wiederholungen angeben, manuell eingefügt werden können, wird die Wiederholung dadurch nicht von LilyPond erkannt. Wiederholte Stellen werden besser notiert, indem man die Wiederholungs-Befehle einsetzt, die automatische die richtigen Taktlinien setzen. Das ist beschrieben inAbschnitt 1.4 [Wiederholungszeichen], Seite 128.

Zusätzlich kann noch `"||:"` verwendet werden, dass sich genauso wie `"|:"` verhält, außer bei Zeilenumbrüchen, wo ein doppelter Taktstrich am Ende der Zeile ausgegeben wird und ein öffnender Wiederholungsstrich am Anfang der nächsten Zeile.

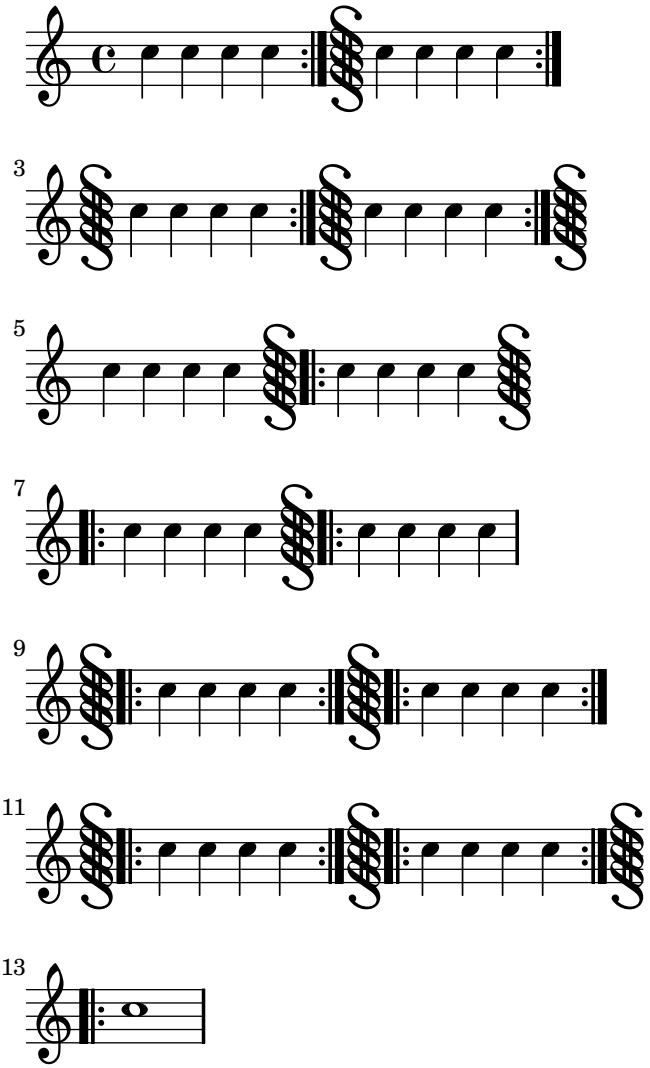
```
c4 c c c
\bar "||:"
c4 c c c \break
\bar "||:"
c4 c c c
```



Für Kombinationen von Wiederholungen mit dem segno-Zeichen gibt es sechs verschiedene Variationen:

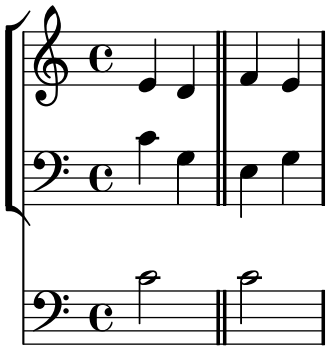
```
c4 c c c
\bar ":|S"
c4 c c c \break
\bar ":|S"
c4 c c c
\bar ":|S."
c4 c c c \break
\bar ":|S."
```

```
c4 c c c
\bar "S|:"
c4 c c c \break
\bar "S|:"
c4 c c c
\bar ".S|:"
c4 c c c \break
\bar ".S|:"
c4 c c c
\bar ":|S|:"
c4 c c c \break
\bar ":|S|:"
c4 c c c
\bar ":|S.|:"
c4 c c c \break
\bar ":|S.|:"
c1
```



In Partituren mit vielen Systemen wird ein `\bar`-Befehl in einem System automatisch auf alle anderen Systeme angewendet. Die resultierenden Taktstriche sind miteinander verbunden innerhalb einer Gruppe (`StaffGroup`) oder einem Klaviersystem (`PianoStaff` bzw. (`GrandStaff`).

```
<<
  \new StaffGroup <<
    \new Staff {
      e4 d
      \bar "||"
      f4 e
    }
    \new Staff { \clef bass c4 g e g }
  >>
  \new Staff { \clef bass c2 c2 }
>>
```



Ausgewählte Schnipsel

Der Befehl `\bar` *Taktart* ist eine Kurzform von: `\set Timing.whichBar =` *Taktart*. Immer, wenn `whichBar` auf einen Wert gesetzt wird, wird ein Taktstrich dieses Typs erzeugt.

Der automatisch erzeugte Taktstrich ist `"|"`. Das kann jederzeit durch den Befehl `\set Timing.defaultBarType =` *Taktstrichart* geändert werden.

Siehe auch

Notationsreferenz: Abschnitt 4.3.1 [Zeilenumbrüche], Seite 473, Abschnitt 1.4 [Wiederholungszeichen], Seite 128, [Systeme gruppieren], Seite 162.

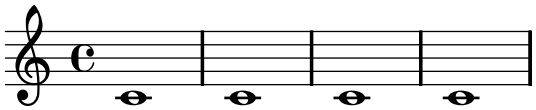
Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

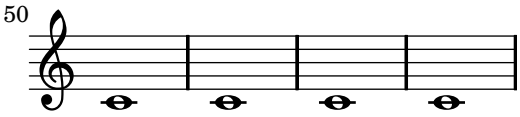
Referenz der Interna: Abschnitt “BarLine” in *Referenz der Interna* (erstellt auf Abschnitt “Staff” in *Referenz der Interna*-Ebene), Abschnitt “SpanBar” in *Referenz der Interna* (über Systeme), Abschnitt “Timing_translator” in *Referenz der Interna* (für Timing-Eigenschaften).

Taktzahlen

Taktzahlen werden standardmäßig zu Beginn eines jeden Systems ausgegeben, ausgenommen ist die erste Zeile. Die Zahl selber wird in der `currentBarNumber`-Eigenschaft gespeichert, die normalerweise für jeden Takt aktualisiert wird. Sie kann aber auch manuell gesetzt werden:

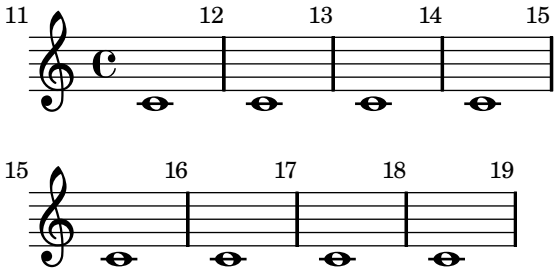
```
c1 c c c
\break
\set Score.currentBarNumber = #50
c1 c c c
```





Taktnummern können in regelmäßigem Abstand ausgegeben werden, anstatt dass sie nur am Beginn des Systems erscheinen. Um das zu erreichen, muss die Standardeinstellung verändert werden, um zu erlauben, dass Taktnummern an anderen Stellen als dem Beginn von Systemen ausgegeben werden. Das wird mit der Eigenschaft `break-visibility` von `BarNumber` vorgenommen. Sie braucht drei Werte, die auf `#t` (wahr) oder `#f` (falsch) gestellt werden können, womit angegeben wird, ob die Taktnummer an der entsprechenden Stelle sichtbar ist. Die Reihenfolge der Werte ist: *Ende der Zeile*, *Mitte der Zeile* und *Beginn der Zeile*. Im folgenden Beispiel werden die Taktlinien überall ausgegeben:

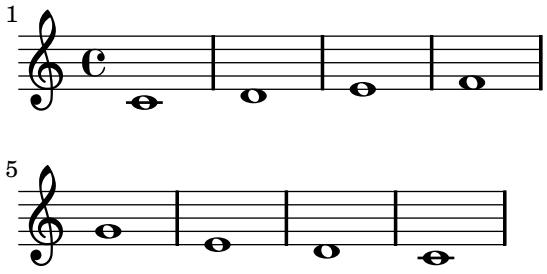
```
\override Score.BarNumber #'break-visibility = #'#(#t #t #t)
\set Score.currentBarNumber = #11
% Permit first bar number to be printed
\bar ""
c1 | c | c | c
\break
c1 | c | c | c
```



Setzen der Taktnummer für den ersten Takt

Standardmäßig wird die erste Taktzahl einer Partitur nicht gesetzt, wenn sie weniger oder gleich '1' ist. Indem man `barNumberVisibility` auf `all-bar-numbers-visible` setzt, kann eine beliebige Taktzahl für den ersten und die folgenden Takte gesetzt werden. Eine leere Taktlinie muss jedoch vor der ersten Note eingefügt werden, damit das funktioniert.

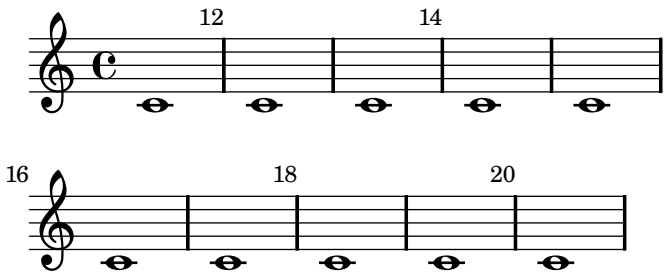
```
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Setzen der Taktnummern in regelmäßigen Intervallen

Taktnummern können in regelmäßigen Intervallen gesetzt werden, indem man die Eigenschaft `barNumberVisibility` definiert. In diesem Beispiel werden die Taktnummern jeden zweiten Takt gesetzt, außer am Ende einer Zeile.

```
\relative c' {
  \override Score.BarNumber #'break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```



Setzen von Taktnummern in Kästen oder Kreisen

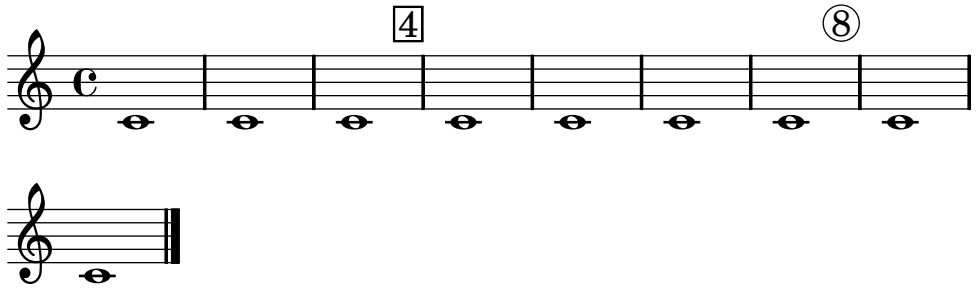
Taktnummern können auch in Boxen oder Kreisen gesetzt werden.

```
\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber #'break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % Increase the size of the bar number by 2
  \override Score.BarNumber #'font-size = #2

  % Draw a box round the following bar number(s)
  \override Score.BarNumber #'stencil
    = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
  \repeat unfold 5 { c1 }

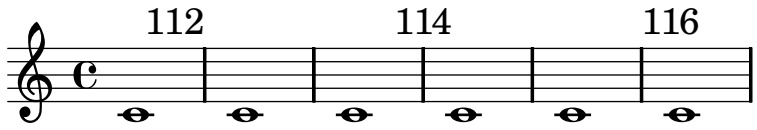
  % Draw a circle round the following bar number(s)
  \override Score.BarNumber #'stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 4 { c1 } \bar "|."
}
```



Taktnummern ausrichten

Taktnummern sind standardmäßig links an ihrem Ursprungsobjekt ausgerichtet. Das ist normalerweise die linke Ecke einer Linie oder, wenn die Nummern innerhalb einer Zeile gesetzt werden, auf der linken Seite eines Taktstrichs. Die Nummern können auch direkt über dem Taktstrich positioniert werden oder rechts vom Taktstrich gesetzt werden.

```
\relative c' {
  \set Score.currentBarNumber = #111
  \override Score.BarNumber #'break-visibility = #all-visible
  % Increase the size of the bar number by 2
  \override Score.BarNumber #'font-size = #2
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c1
  % Center-align bar numbers
  \override Score.BarNumber #'self-alignment-X = #CENTER
  c1 | c1
  % Left-align bar numbers
  \override Score.BarNumber #'self-alignment-X = #LEFT
  c1 | c1
}
```
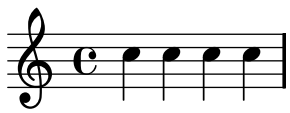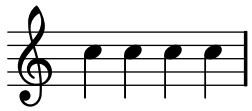


Entfernung von Taktnummern in einer Partitur

Taktnummern können vollkommen aus den Noten entfernt werden, indem man den `Bar_number_engraver` aus dem `Score`-Kontext entfernt.

```
\layout {
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}

\relative c'' {
  c4 c c c \break
  c4 c c c
}
```





Siehe auch

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “BarNumber” in *Referenz der Interna*, Abschnitt “Bar_number_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Taktnummern können mit der oberen Ecke der Klammer zu Beginn des Systems zusammenstoßen. Um das zu verhindert, kann die `padding`-Eigenschaft von `BarNumer` verwendet werden, um die Zahl zu verschieben. Für mehr Information siehe Abschnitt “StaffgGroup” in *Referenz der Interna* und Abschnitt “BarNumber” in *Referenz der Interna*.

Takt- und Taktzahlüberprüfung

Die Taktüberprüfung hilft, Fehler in den Notendauern zu entdecken. Eine Taktüberprüfung wird mit dem Taktstrichsymbol „`|`“ (Taste AltGr+<) eingegeben. Immer, wenn LilyPond bei der Ausgabe des Notendrucks auf dieses Zeichen stößt, sollte hier in den Noten auch ein Taktstrich erscheinen. Wenn das nicht der Fall ist, wird eine Warnung ausgegeben. Im nächsten Beispiel resultiert die zweite Taktüberprüfung in einer Fehlermeldung.

```
\time 3/4 c2 e4 | g2 |
```

Taktüberprüfungen können auch in Gesangstexten verwendet werden:
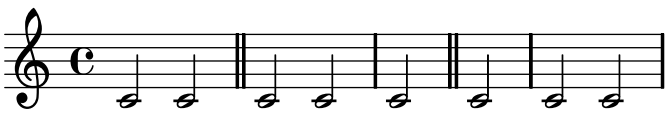
```
\lyricmode {
  \time 2/4
  Twin -- kle | Twin -- kle
}
```

Besonders in mehrstimmiger komplizierter Musik können falschen Notenwerte die ganze Partitur durcheinander bringen. Es lohnt sich also, die Fehlersuche damit zu beginnen, nicht bestandene Taktüberprüfungen zu kontrollieren.

Wenn aufeinander folgende Taktüberprüfungen mit dem gleichen Abstand Fehler produzieren, wird eventuell nur die erste Warnung ausgegeben. Damit wird die Warnung auf den Ursprung de Fehlers fokussiert.

Es ist auch möglich, die Bedeutung des Symbols `|` (Pipe) umzudefinieren, so dass hiermit eine andere Aktion als eine Taktüberprüfung erreicht wird. Das geschieht, indem man der Pipe (`pipeSymbol` einen musikalischen Ausdruck zuweist. Im nächsten Beispiel wird `|` dazu verwendet, eine doppelte Taktlinie auszugeben, woimmer man das Zeichen auch setzt. Gleichzeitig hört das Zeichen auf, als Taktüberprüfung zu funktionieren.

```
pipeSymbol = \bar "||"
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



Wenn man größere Musikstücke kopiert, kann es hilfreich sein, wenn LilyPond überprüft, ob die Taktnummer, in der Sie gerade kopieren, mit der des Originales übereinstimmt. Das kann mit dem Befehl `\barNumberCheck` folgenderweise überprüft werden:

`\barNumberCheck #123`

Eine Warnung wird ausgegeben, wenn der interne Zähler `currentBarNumber` von LilyPond nicht mit dem Wert 123 übereinstimmt.

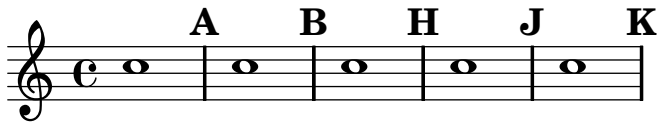
Siehe auch

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Übungszeichen

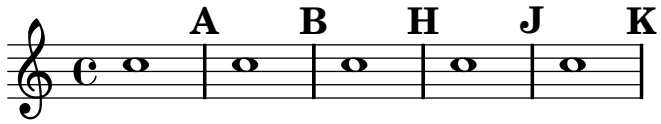
Übungszeichen können mit dem `\mark`-Befehl ausgegeben werden:

```
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



Das Zeichen wird automatisch um einen Wert heraufgesetzt, wenn man `\mark \default` benutzt, aber man kann auch eine Ganzzahl als Argument einsetzen, wenn man das Zeichen manuell setzen will. Der Wert, der eingesetzt werden soll, wird in der Eigenschaft `rehearsalMark` gespeichert.

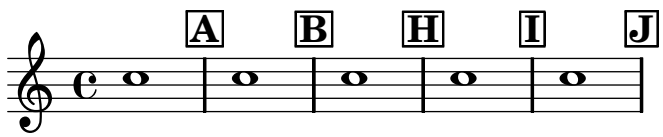
```
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



Der Buchstabe „I“ wird ausgelassen, was den allgemeinen Notensatzregeln entspricht. Wenn Sie dennoch den Buchstaben „I“ benutzen, wollen, müssen Sie einen der folgenden Stile benutzen, je nachdem, was für einen Übungszeichenstil Sie wollen (Buchstaben, Buchstaben in einem Kasten, Buchstaben in einem Kreis).

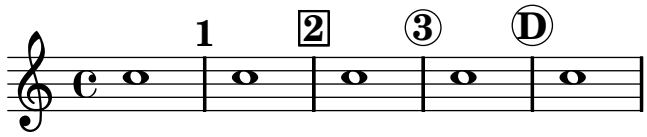
```
\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet
```

```
\set Score.markFormatter = #format-mark-box-alphabet
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



Der Stil der Übungszeichen wird von der Eigenschaft `markFormatter` definiert. Das ist eine Funktion, die das aktuelle Zeichen und den aktuellen Kontext als Argument annimmt. Sie gibt dann ein Textbeschriftungsobjekt aus. Im folgenden Beispiel ist `markFormatter` so definiert, dass eine Zahl ausgegeben wird. Dann wird ein Übungszeichen in einem Kasten produziert.

```
\set Score.markFormatter = #format-mark-numbers
c1 \mark \default
c1 \mark \default
\set Score.markFormatter = #format-mark-box-numbers
c1 \mark \default
\set Score.markFormatter = #format-mark-circle-numbers
c1 \mark \default
\set Score.markFormatter = #format-mark-circle-letters
c1
```



Die Datei '`scm/translation-functions.scm`' beinhaltet die Definitionen für `format-mark-numbers` (erstelle-Zeichen-Nummern), `format-mark-box-numbers` (erstelle-Zeichen-Kasten-Nummern), `format-mark-letters` (erstelle-Zeichen-Buchstaben) und `format-mark-box-letters` (erstelle-Zeichen-Kasten-Buchstaben). Sie können als Anleitung für eigene Formatierungsfunktionen dienen.

Die Funktionen `format-mark-barnumbers`, `format-mark-box-barnumbers`und `format-mark-circle-barnumbers` könenn eingesetzt werden, um Taktnummern anstelle der fortlaufenden Zahlen bzw. Buchstaben zu erhalten.

Andere Übungszeichenstile können auch manuell gesetzt werden:

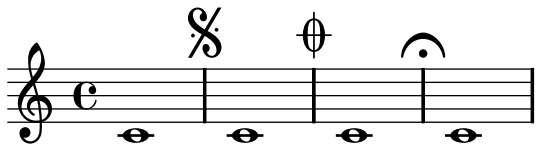
```
\mark "A1"
```

`Score.markFormatter` hat keine Auswirkungen auf solcherart definierte Zeichen. Man kann aber auch mit `\markup` Textbeschriftungsobjekte zu dem selbstdefinierten Zeichen hinzufügen:

```
\mark \markup{ \box A1 }
```

Musikbuchstaben (wie etwa das Segno-Zeichen) können mit dem Befehl `\musicglyph` als ein `\mark`-Zeichen definierte werden:

```
c1 \mark \markup { \musicglyph #"scripts.segno" }
c1 \mark \markup { \musicglyph #"scripts.coda" }
c1 \mark \markup { \musicglyph #"scripts.ufermata" }
c1
```



Siehe Abschnitt A.7 [Die Feta-Schriftart], Seite 578, wo alle Symbole gezeigt sind, die mit dem Befehl `\musicglyph` ausgegeben werden können.

Übliche Veränderungen der Positionierung von Übungszeichen finden sich in Abschnitt 1.8.2 [Text formatieren], Seite 210. Zu noch präzieserer Kontrolle siehe `break-alignable-interface` in Abschnitt 5.5.1 [Objekte ausrichten], Seite 553.

Siehe auch

Notationsreferenz: Abschnitt A.7 [Die Feta-Schriftart], Seite 578, Abschnitt 1.8.2 [Text formatieren], Seite 210, Abschnitt 5.5.1 [Objekte ausrichten], Seite 553.

Installierte Dateien: ‘`scm/translation-functions.scm`’ beinhaltet die Definition von `format-mark-numbers` und `format-mark-letters`. Sie können als Anleitung für eigene Funktionen benutzt werden.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.
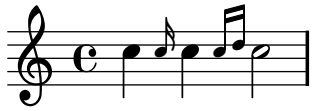
Referenz der Interna: Abschnitt “MarkEvent” in *Referenz der Interna*, Abschnitt “Mark_engraver” in *Referenz der Interna*, Abschnitt “RehearsalMark” in *Referenz der Interna*.

1.2.6 Besondere rhythmische Fragen

Verzierungen
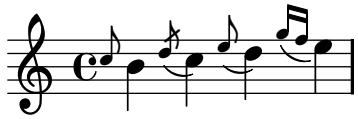
Verzierungen, mit dem Befehl `\grace` notiert, sind ausgeschriebene Ornamente. Sie werden in einer kleineren Schriftgröße gesetzt und nehmen keine logische Zeit im Takt ein.

```
c4 \grace c16 c4
\grace { c16[ d16] } c2
```
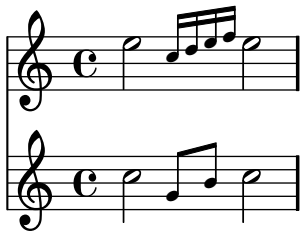


LilyPond hat auch Unterstützung für zwei besondere Verzierungen, den Vorschlag (engl. acciaccatura) und den Vorhalt (engl appogiatura). Der Vorschlag wird durch eine verkleinerte Note mit Schrägstrich und Bogen notiert. Der Vorhalt dagegen ist eine Verzierung, die einen bestimmten Notenwert der Hauptnote für sich beansprucht. Er wird als verkleinerte Note ohne Schrägstrich notiert.

```
\grace c8 b4
\acciaccatura d8 c4
\appoggiatura e8 d4
\acciaccatura { g16[ f] } e4
```
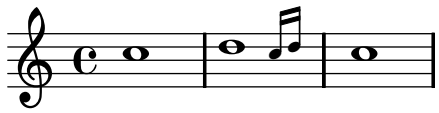


Die Position von Verzierungen ist zwischen Notensystemen synchronisiert. Im nächsten Beispiel stehen im einen System zwei 16-Noten für jede 8-Note des zweiten Systems:

```
<< \new Staff { e2 \grace { c16[ d e f] } e2 }
   \new Staff { c2 \grace { g8[ b] } c2 } >>
```
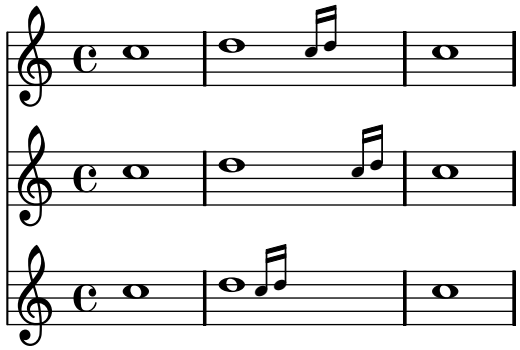


Wenn Sie eine Note mit einer Verzierung abschließen wollen, müssen Sie den `\afterGrace`-Befehl benutzen. Er benötigt zwei Argumente: die Hauptnote und die Verzierung, die nach der Hauptnote folgen soll:

```
c1 \afterGrace d1 { c16[ d] } c1
```
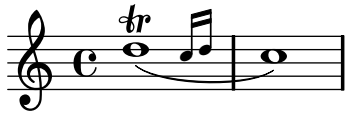


Damit wird die Verzierung mit einem Abstand von der Hauptnote gesetzt, der 3/4 der Dauer der Hauptnote entspricht. Dieser Standard kann durch Definition von `afterGraceFraction` verändert werden. Das nächste Beispiel zeigt, wie sich der Abstand verändert, wenn der Wert 3/4, 15/16 und 1/2 der Hauptnote beträgt.

```
<<
  \new Staff {
    c1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff {
    #(define afterGraceFraction (cons 15 16))
    c1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff {
    #(define afterGraceFraction (cons 1 2))
    c1 \afterGrace d1 { c16[ d] } c1
  }
>>
```



Der Abstand zwischen der Hauptnote und der Verzierung kann auch mit unsichtbaren Noten beeinflusst werden. Im nächsten Beispiel wird die Verzierung mit einem Abstand von 7/8 zur Hauptnote gesetzt.

```
\new Voice {
  << { d1^\trill_( }
     { s2 s4. \grace { c16[ d] } } >>
  c1)
}
```



Ein `\grace`-Notenabschnitt wird nach besonderen Satzregeln gesetzt, um z. B. kleinere Noten zu benutzen und die Richtung der Hälse einzustellen. Veränderungen am Layout müssen also innerhalb des Verzierungsausdrucks gesetzt werden, damit sie auch eine Auswirkung haben. Die Veränderungen müssen auch innerhalb des Verzierungsausdrucks rückgängig gemacht werden. In diesem Fall wird die Richtung der Hälse geändert und dann wieder der Standard eingestellt:

```
\new Voice {
  \acciaccatura {
    \stemDown
    f16->
    \stemNeutral
  }
  g4 e c2
}
```
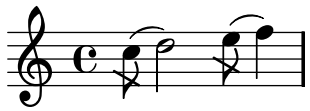


Ausgewählte Schnipsel

Using grace note slashes with normal heads

The slash through the stem found in acciaccaturas can be applied in other situations.

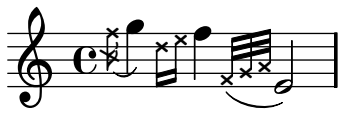
```
\relative c'' {
  \override Stem #'stroke-style = #"grace"
  c8( d2) e8( f4)
}
```



Veränderung des Layouts von Verzierungen innerhalb der Noten

Das Layout von Verzierungsausdrücken kann in der Musik verändert werden mit den Funktionen `add-grace-property` und `remove-grace-property`. Das folgende Beispiel definiert die Richtung von Hälsen (Stem) für diese Verzierung, sodass die Hälse nicht immer nach unten zeigen, und ändert den Standardnotenkopf in ein Kreuz.

```
\relative c'' {
  \new Staff {
    #(remove-grace-property 'Voice 'Stem 'direction)
    #(add-grace-property 'Voice 'NoteHead 'style 'cross)
    \new Voice {
       \acciaccatura { f16 } g4
       \grace { d16[ e] } f4
       \appoggiatura { f,32[ g a] } e2
    }
  }
}
```



Globale Umdefinition von Verzierungsnoten
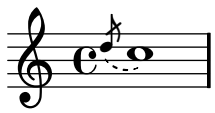
Die globalen Standardeinstellungen für Verzierungsnoten werden in den Variablen`startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`,

`stopAcciaccaturaMusic`, `startAppoggiaturaMusic` und `stopAppoggiaturaMusic` gespeichert, die in der Datei '`ly/grace-init.ly`' definiert sind. Wenn man sie umdefiniert, können andere Effekte erreicht werden.

```
startAcciaccaturaMusic = {
  s1*0(
  \override Stem #'stroke-style = #"grace"
  \slurDashed
}

stopAcciaccaturaMusic = {
  \revert Stem #'stroke-style
  \slurSolid
  s1*0)
}

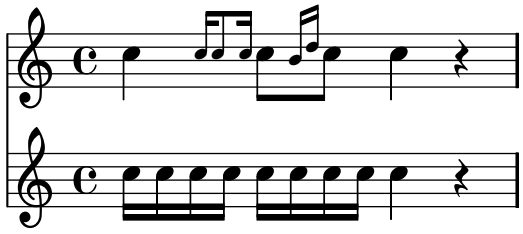
\relative c'' {
  \acciaccatura d8 c1
}
```



Positionierung von Verzierungen mit verschiebbarem Platz

Wenn man die Eigenschaft `'strict-grace-spacing` aktiviert, werden die Verzierungsnoten "fließend" gemacht, d.h. sie sind von den normalen Noten los gekoppelt: Zuerst werden die normalen Noten platziert, dann erst die Verzierungen links von der Hauptnote gesetzt.

```
\relative c'' {
  <<
    \override Score.SpacingSpanner #'strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16[ d] } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}
```



Siehe auch

Glossar: Abschnitt “grace notes” in *Glossar*, Abschnitt “acciaccatura” in *Glossar*, Abschnitt “appoggiatura” in *Glossar*.

Notationsreferenz: [Manuelle Balken], Seite 84.

Installierte Dateien: ‘`ly/grace-init.ly`’.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.
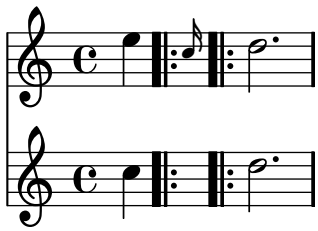
Referenz der Interna: Abschnitt “GraceMusic” in *Referenz der Interna*, Abschnitt “Grace_beam_engraver” in *Referenz der Interna*, Abschnitt “Grace_engraver” in *Referenz der Interna*, Abschnitt “Grace_spacing_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Ein Vorschlag (*acciaccatura*) mit mehreren Noten und Balken wird ohne den Schrägstrich gesetzt und sieht einem Vorhalt (*appoggiatura*) sehr ähnlich.
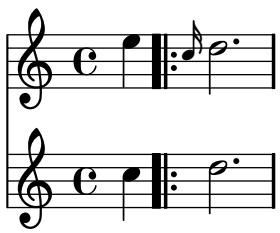
Die Synchronisation von Verzierungen kann auch zu Überraschungen führen. Auch andere Symbole der Systeme, wie Vorzeichen, Taktlinien usw., werden synchronisiert. Vorsicht ist geboten, wenn nur in bestimmten Systemen Verzierungen vorkommen:

```
<<
  \new Staff { e4 \bar "|:" \grace c16 d2. }
  \new Staff { c4 \bar "|:" d2. }
>>
```



Dem kann abgeholfen werden, indem unsichtbare Verzierungsnoten der selben Länge in die anderen Systeme gesetzt werden. Im obigen Beispiel müsste also

```
<<
  \new Staff { e4 \bar "|:" \grace c16 d2. }
  \new Staff { c4 \bar "|:" \grace s16 d2. }
>>
```



gesetzt werden.

Der Einsatz von Verzierungsnoten innerhalb von Stimmen-Kontexten kann den Notensatz der Stimme verwirren. Dieses Problem kann umgangen werden, indem man eine Note oder Pause zwischen dem `Voice`-Befehl und der Verzierungsnote setzt.

```
accMusic = {
  \acciaccatura { f8 } e8 r8 \acciaccatura { f8 } e8 r4
}
```

```
\new Staff {
  <<
    \new Voice {
      \relative c'' {
        r8 r8 \voiceOne \accMusic \oneVoice r8 |
        r8 \voiceOne r8 \accMusic \oneVoice r8 |
      }
    }
    \new Voice {
      \relative c' {
        s8 s8 \voiceTwo \accMusic \oneVoice s8 |
        s8 \voiceTwo r8 \accMusic \oneVoice s8 |
      }
    }
  >>
}
```



Verzierungsabschnitte sollten nur innerhalb von sequentiellen musikalischen Ausdrücken benützt werden. Wenn sie ineinandergeschachtelt werden, kann es zu Fehlermeldungen oder Abstürzen kommen.

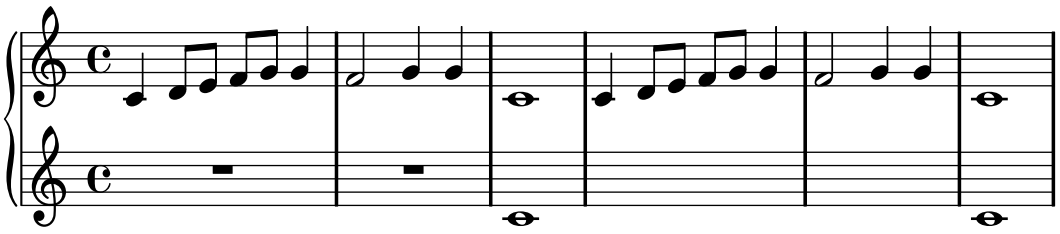
An Kadenzen ausrichten

In Orchesterpartituren stellen Kadenzen ein besonderes Problem dar: Wenn in der Partitur ein Instrument eine Kadenz spielt, die notiert wird, müssen die anderen Stimmen genau die entsprechende Anzahl Noten überspringen, damit sie nicht zu früh oder zu spät einsetzen.

Eine Lösung ist es, die Funktionen `mmrest-of-length` oder `skip-of-length` zu benutzen. Diese Scheme-Funktionen brauchen einen definierten Notenabschnitt (eine Variable) als Argument und produzieren entweder Ganztaktpausen oder leere Takte, die genauso lang sind wie der Notenabschnitt.

```
MyCadenza = \relative c' {
  c4 d8 e f g g4
  f2 g4 g
}

\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(ly:export (mmrest-of-length MyCadenza))
    c'1
    #(ly:export (skip-of-length MyCadenza))
    c'1
  }
>>
```



Siehe auch

Glossar: Abschnitt “cadenza” in *Glossar*.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Verwaltung der Zeiteinheiten

Die Zeit in einer Partitur wird vom `Timing_translator` verwaltet, der sich in den Standardeinstellungen im `Score`-Kontext befindet. Eine Parallelbezeichnung, `Timing`, wird dem Kontext hinzugefügt, in dem sich `Timing_translator` befindet. Um sicherzugehen, dass `Timing` erhältlich ist, muss man eventuell den enthaltenden Kontext manuell erstellen (also etwa einen `Voice`- oder `Staff`-Kontext.

Die folgenden Eigenschaften von `Timing` werden eingesetzt, um die Zeit in Partituren zu verwalten.

`currentBarNumber (aktuelle Taktnummer)`
: Die gerade aktuelle Taktzahl. Für ein Beispiel, das die Benutzung dieser Eigenschaft zeigt, siehe [Taktzahlen], Seite 92.

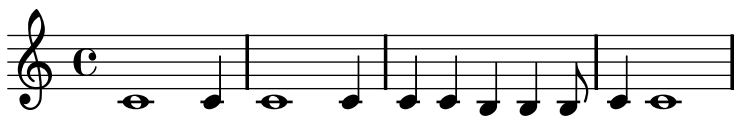
`measureLength (Taktlänge)`
: Die Länge der Takte mit der aktuellen Taktart. In einem 4/4-Takt ist sie 1, in einem 6/8-Takt 3/4. Dieser Wert bestimmt, wann eine Taktlinie gezogen wird und wie automatische Balken erstellt werden sollen.

`measurePosition (Taktposition)`
: Der Schlag im Takt zum aktuellen Moment. Dieser Wert wird zurückgesetzt, indem `measureLength` (die Taktlänge) abgezogen wird, wenn der Wert von `measureLength` erreicht oder überschritten wird. Wenn das passiert, wird der Zähler `currentBarNumber` (aktuelle Taktnummer) erhöht.

`timing (Zeitberechnung)`
: Wenn auf wahr gesetzt, werden die oben genannten Variablen zu jedem Zeitpunkt aktualisiert. Wenn auf falsch gesetzt, bleibt der Engraver unendlich lange im aktuellen Takt.

Zeitverwaltung kann geändert werden, indem man diese Variablen direkt beeinflusst. Im nächsten Beispiel wird die normale Taktart mit 4/4 angegeben, aber `measureLength` wird auf 5/4 gesetzt. An der Stelle 4/8 des dritten Taktes wird die Taktposition (`measurePosition`) um 1/8 auf 5/8 erhöht, so dass der Takt im Ergebnis 1/8 kürzer ist. Die nächste Taktlinie wird dann auch bei 9/8 gezogen und nicht bei 5/4.

```
\set Score.measureLength = #(ly:make-moment 5 4)
c1 c4
c1 c4
c4 c4
\set Score.measurePosition = #(ly:make-moment 5 8)
b4 b4 b8
c4 c1
```



Wie das Beispiel zeigt, erstellt `ly:make-moment n m` die Dauer Zähler/Nenner einer ganzen Note. Zum Beispiel heißt `ly:make-moment 1 8` die Dauer einer Achtelnote, und `ly:make-moment 7 16` die Dauer von sieben Sechszehntelnoten.
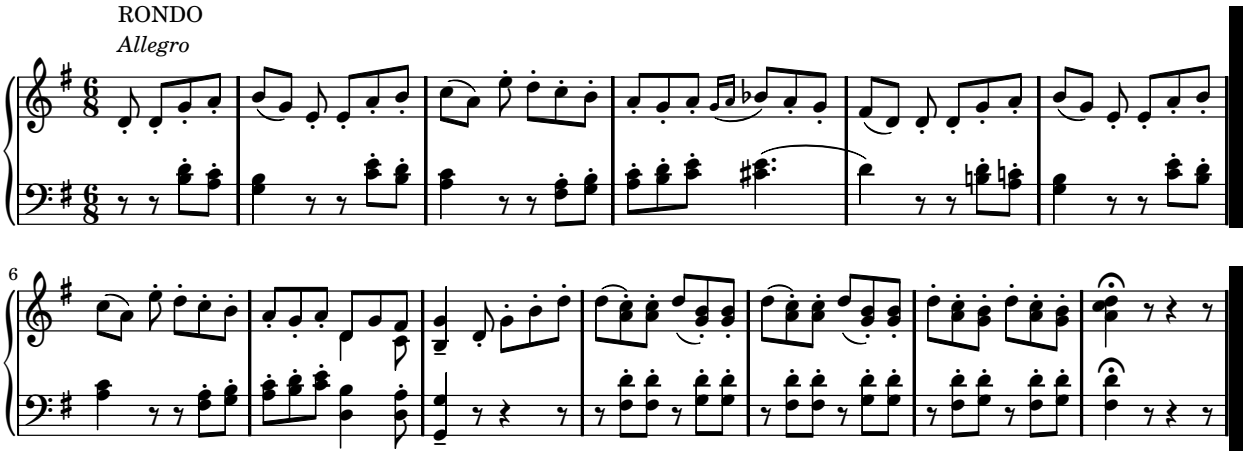
Siehe auch

Notationsreferenz: [Taktzahlen], Seite 92, [Musik ohne Metrum], Seite 67.

Schnipsel: Abschnitt “Rhythms” in *Schnipsel*.

Referenz der Interna: Abschnitt “Timing_translator” in *Referenz der Interna*, Abschnitt “Score” in *Referenz der Interna*

1.3 Ausdrucksbezeichnungen



Dieser Abschnitt zeigt verschiedene Ausdrucksbezeichnungen, die zur Partitur hinzugefügt werden können.

1.3.1 Ausdrucksbezeichnungen an Noten angehängt

Dieser Abschnitt erklärt, wie man Ausdrucksbezeichnungen erstellt, die an Noten gebunden sind: Artikulationszeichen, Ornamente und Dynamikzeichen. Es werden auch Methoden gezeigt, eigene Ausdrucksbezeichnungen zu erstellen.
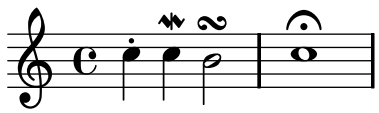
Artikulationszeichen und Verzierungen

Eine Vielfalt an Symbolen kann über und unter den Noten erscheinen, um zu markieren, auf welche Art die Note ausgeführt werden soll. Hierzu wird folgende Syntax benutzt:

```
Note\Bezeichnung
```
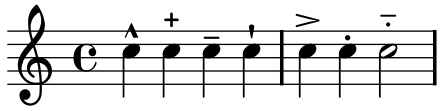
Die möglichen Werte für *Bezeichnung* sind aufgelistet in Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640. Ein Beispiel:

```
c4\staccato c\mordent b2\turn
c1\fermata
```



Einige dieser Artikulationszeichen haben eine Abkürzung, damit es einfacher ist, sie zu schreiben. Die Abkürzung wird an die Notenbezeichnung gehängt, wobei ihre Syntax aus einem Minuszeichen `-` besteht, gefolgt von dem Symbol, das dem Artikulationszeichen zugeordnet ist. Es gibt diese Abkürzungen für *marcato*, *stopped* (gedämpft), *tenuto*, *staccatissimo*, *accent*, *staccato*, und *portato*. Die ihnen entsprechenden Symbole werden also folgendermaßen notiert:

```
c4-^ c-+ c-- c-|
c4-> c-. c2-_
```
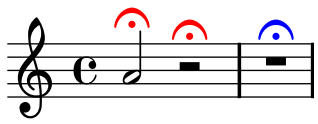


Die Regeln für die standardmäßige Platzierung von Artikulationszeichen werden in der Datei '`scm/script.scm`' definiert. Artikulationszeichen und Ornamente können manuell über oder unter dem System gesetzt werden, siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

Artikulationszeichen sind `Script`-Objekte. Ihre Eigenschaften werden ausführlich in Abschnitt "Script" in *Referenz der Interna* beschrieben.

Artikulationen können neben Noten auch an Pausen gehängt werden, aber sie können nicht an Mehrtaktpausen gehängt werden. Ein besonderer Befehl, `fermataMarkup`, wurde definiert, damit man eine Fermate an eine Mehrtaktpause anfügen kann (und nur hieran). Damit wird ein `MultiMeasureRestText`-Objekt erstellt.

```
\override Script #'color = #red
\override MultiMeasureRestText #'color = #blue
a2\fermata r\fermata
R1\fermataMarkup
```



Zusätzlich zu den Artikulationszeichen können auch Text und Beschriftung an Noten angehängt werden. Siehe auch [Textarten], Seite 201.

Zu weiterer Information über die Reihenfolge von Scripten und TextScripten, die an Noten angehängt werden, siehe Abschnitt "Positionierung von Objekten" in *Handbuch zum Lernen*.

Ausgewählte Schnipsel
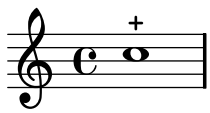
Die Standardwerte der Abkürzungen von Artikulationen verändern
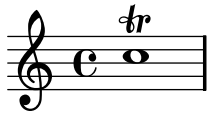
Die Abkürzungen sind in der Datei '`ly/script-init.ly`' definiert, wo den Variablen `dashHat`, `dashPlus`, `dashDash`, `dashBar`, `dashLarger`, `dashDot` und `dashUnderscore` Standardwerte zugewiesen werden. Diese Standardwerte können verändert werden. Um zum Beispiel die Abkürzung `-+` (`dashPlus`) mit dem Triller anstatt mit dem `+`-Symbol zu assoziieren, muss der Wert `trill` der Variable `dashPlus` zugewiesen werden:

```
\relative c'' { c1-+ }

dashPlus = "trill"

\relative c'' { c1-+ }
```
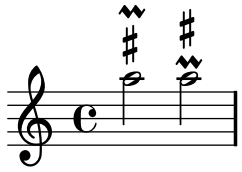




Die vertikale Anordnung von Beschriftungen kontrollieren

Die vertikale Anordnung von Beschriftungen wird mit der `'script-priority`-Eigenschaft kontrolliert. Um so kleiner die Zahl, umso näher wird die Beschriftung in Bezug auf die Note gesetzt. In diesem Beispiel hat das `TextScript`-Objekt (das Kreuz) zuerst die niedrigste Priorität, wird also auch am niedrigsten in dem ersten Beispiel gesetzt. Im zweiten Fall hat der Praller (das `Script`) die niedrigste Priorität, darum wird er am nächsten zum System gesetzt. Wenn zwei Objekte die gleiche Priorität haben, wird ihre Reihenfolge anhand ihres Auftretens in der Quelldatei entschieden.

```
\relative c''' {
  \once \override TextScript #'script-priority = #-100
  a2^\prall^\markup { \sharp }
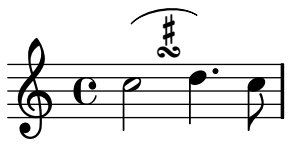
  \once \override Script #'script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```



Einen Doppelschlag mit Vorhalt erstellen

Einen Doppelschlag mit Vorhalt zu erstellen, wobei die untere Note das Vorzeichen benutzt, erfordert einige Einstellungsänderungen. Die `outside-staff-priority`-Eigenschaft muss auf falsch (`#f`) gesetzt werden, weil sie sonst über die Eigenschaft `avoid-slur property` dominieren würde. Der Wert von `halign` wird benutzt, um den Doppelschlag horizontal zu positionieren.

```
\relative c'' {
  \once \override TextScript #'avoid-slur = #'inside
  \once \override TextScript #'outside-staff-priority = ##f
  c2(^\markup \tiny \override #'(baseline-skip . 1) {
    \halign #-4
    \center-column {
      \sharp
      \musicglyph #"scripts.turn"
    }
  }
  d4.) c8
}
```



Siehe auch

Glossar: Abschnitt “tenuto” in *Glossar*, Abschnitt “accent” in *Glossar*, Abschnitt “staccato” in *Glossar*, Abschnitt “portato” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Positionierung von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: [Textarten], Seite 201, Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538, Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640, [Triller], Seite 126.

Installierte Dateien: ‘`scm/script.scm`’.

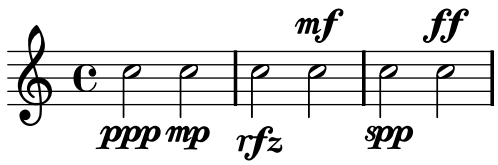
Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “Script” in *Referenz der Interna*, Abschnitt “TextScript” in *Referenz der Interna*.

Dynamik

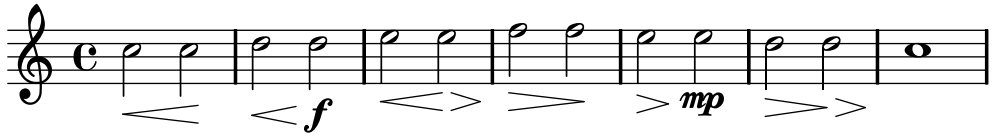
Absolute Dynamikbezeichnung wird mit Befehlen nach den Noten angezeigt, etwa `c4\ff`. Die vordefinierten Befehle lauten: `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, and `\rfz`. Die Dynamikzeichen können manuell unter- oder oberhalb des Systems platziert werden, siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

```
c2\ppp c\mp
c2\rfz c^\mf
c2_\spp c^\ff
```
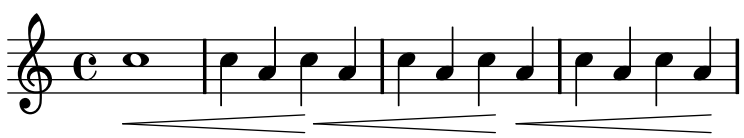


Eine *Crescendo*-Klammer wird mit dem Befehl `\<` begonnen und mit `\!`, einem absoluten Dynamikbefehl oder einer weiteren Crescendo- oder Decrescendo-Klammer beendet. Ein *Decrescendo* beginnt mit `\>` und wird auch beendet mit `\!`, einem absoluten Dynamikbefehl oder einem weiteren Crescendo oder Decrescendo. `\cr` und `\decr` können anstelle von `\<` und `\>` benutzt werden. Die Befehle ergeben standardmäßig Crescendo-Klammern.

```
c2\< c\!
d2\< d\f
e2\< e\>
f2\> f\!
e2\> e\mp
d2\> d\>
c1\!
```



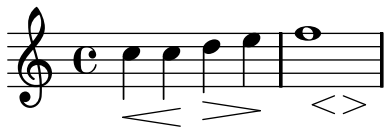
Eine Crescendo-Klammer, die mit `\!` beendet wird, endet an der rechten Seite der Note, welcher `\!` zugeordnet ist. In dem Fall, dass es durch den Beginn eines anderen *crescendo*- oder *decrescendo*-Zeichens beendet wird, endet es in der Mitt der Note, welche das nächste `\<` oder `\>` angehängt hat. Die nächste Klammer beginnt dann am rechten Rand der selben Note anstatt dem normalerweise linken Rand, wenn die vorherige Klammer mit `\!` beendet worden wäre.

```
c1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
```
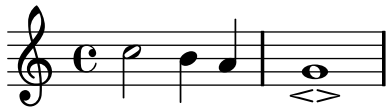


Leere Pausenzeichen werden benötigt, um mehrere Zeichen für eine Note zu notieren. Das ist insbesondere nützlich, wenn man *crescendo* und *decrescendo* zu der selben Note hinzufügen will:

```
c4\< c\! d\> e\!
<< f1 { s4 s4\< s4\> s4\! } >>
```
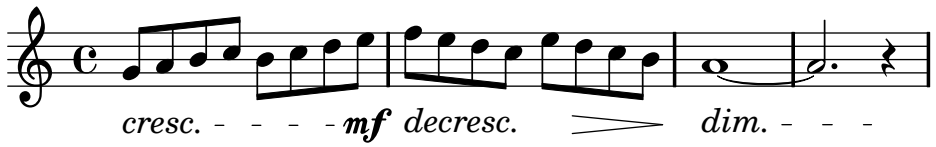


Der `\espressivo`-Befehl kann eingesetzt werden, um crescendo und decrescendo für die selbe Note anzuzeigen. Dieser Befehl ist jedoch als Artikulation, nicht als Dynamikzeichen implementiert.

```
c2 b4 a
g1\espressivo
```
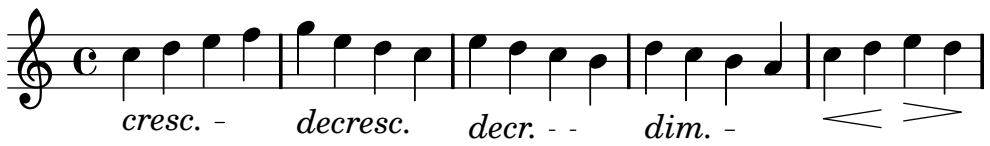


Mit Text gesetzte Crescendo-Bezeichnungen beginnen mit `\cresc`. Mit Text gesetzte Decrescendo-Bezeichnungen beginnen mit `\decresc` oder `\dim`. Fortsetzungslinien werden gesetzt, wenn sie benötigt werden.

```
g8\cresc a b c b c d e\mf |
f8\decresc e d c e\> d c b |
a1\dim ~ |
a2. r4\! |
```



Als Text gesetzte Dynamik-Bezeichnungen können auch die Crescendo-Klammern ersetzen:

```
\crescTextCresc
c4\< d e f\! |
\dimTextDecresc
g4\> e d c\! |
\dimTextDecr
e4\> d c b\! |
\dimTextDim
d4\> c b a\! |
\crescHairpin
\dimHairpin
c4\< d\! e\> d\! |
```

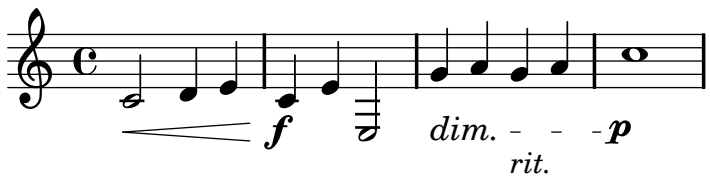


Um neue absolute Dynamikzeichen oder Text, der mit ihnen angeordnet wird, zu erstellen, siehe [Neue Lautstärkezeichen], Seite 114.

Vertikale Position der Zeichen wird von der Funktion Abschnitt “DynamicLineSpanner” in *Referenz der Interna* verwaltet.

Es gibt einen besonderen `Dynamics`-Kontext, um Crescendi und Decrescendi auf einer eigenen Zeile zu notieren. Mit leeren Pausen (`s`) werden die Dauern gesetzt. (Auch Noten in einem `Dynamics`-Kontext nehmen eine Dauer ein, werden aber nicht gesetzt.) Der `Dynamics`-Kontext ist sehr nützlich, um andere Elemente wie Textbeschriftung, Text-Strecker und Klavierpedalbezeichnungen aufzunehmen.

```
<<
  \new Staff \relative c' {
    c2 d4 e |
    c4 e e,2 |
    g'4 a g a |
    c1 |
  }
  \new Dynamics {
    s1\< |
    s1\f |
    s2\dim s2-"rit." |
    s1\p |
  }
>>
```



Vordefinierte Befehle

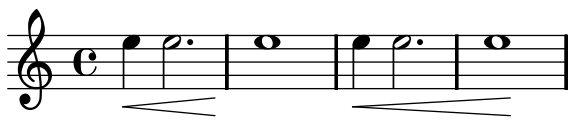
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

Ausgewählte Schnipsel

Das Verhalten von Crescendo-Klammern an Taktlinien beeinflussen

Wenn die Note, an welcher eine Crescendo-Klammer endet, die erste Note eines Taktes ist, wird die Klammer an der vorhergehenden Taktlinie beendet. Dieses Verhalten kann auch mit der Eigenschaft `'to-barline` geändert werden:
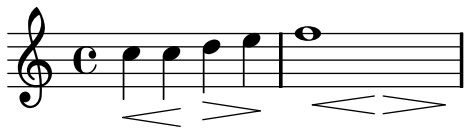
```
\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin #'to-barline = ##f
  e4\< e2.
  e1\!
}
```



Die Mindestlänge von Crescendo-Klammern bestimmen

Wenn Crescendo-Klammern zu kurz sind, können sie verlängert werden, indem die `minimum-length`-Eigenschaft des `Hairpin`-Objektes verändert wird.
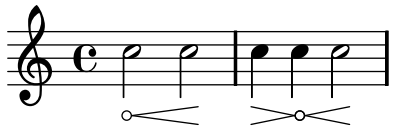
```
\relative c'' {
  c4\< c\! d\> e\!
  \override Hairpin #'minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}
```



Crescendo Klammern al niente schreiben

Crescendo-Klammern können mit einem kleinen Kreis vor der Spitze notiert werden (al niente = bis zum Nichts), indem die `circled-tip`-Eigenschaft des `Hairpin`-Objekts auf `#t` gesetzt wird.

```
\relative c'' {
  \override Hairpin #'circled-tip = ##t
  c2\< c\!
  c4\> c\< c2\!
}
```



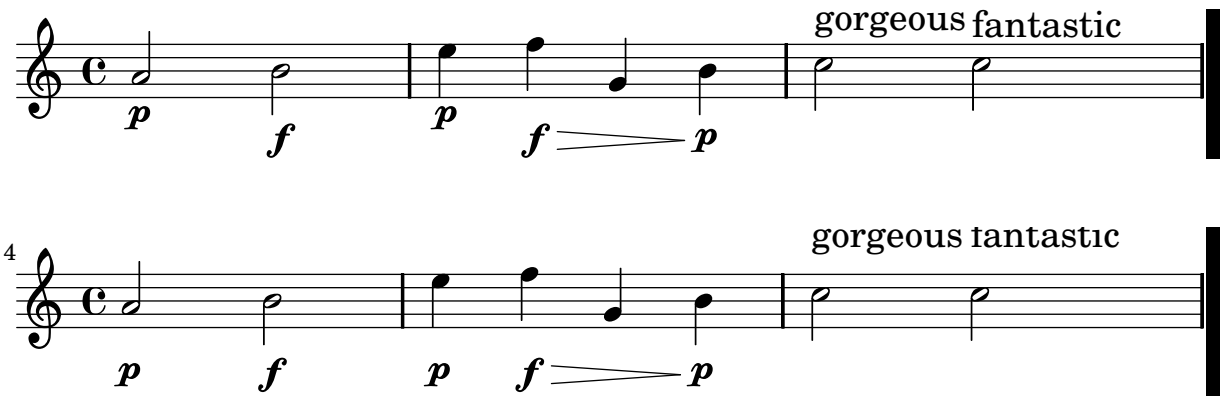
Vertikale Ausrichtung von Dynamik und Textbeschriftung beeinflussen

Indem man die `'Y-extent`-Eigenschaft auf einen passenden Wert setzt, können alle `DynamicLineSpanner`-Objekte (Crescendo-Klammern und Dynamik-Texte) (hairpins and dynamic texts) unabhängig von ihrer wirklichen Ausdehnung an einem gemeinsamen Referenzpunkt ausgerichtet werden. Auf diese Weise ist jedes Element vertikal ausgerichtet und der Notensatz sieht ansprechender aus.

Die gleiche Idee wird benutzt, um Textbeschriftungen an ihrer Grundlinie auszurichten.

```
music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

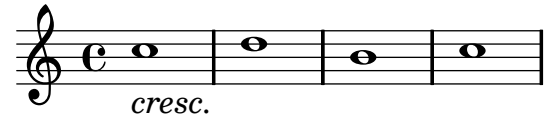
{
  \music
  \break
  \override DynamicLineSpanner #'staff-padding = #2.0
  \override DynamicLineSpanner #'Y-extent = #'(-1.5 . 1.5)
  \override TextScript #'Y-extent = #'(-1.5 . 1.5)
  \music
}
```



Crescendo-Linien von Dynamik-Texten unterdrücken

Dynamik-Texte (wie cresc. und dim.) werden mit einer gestrichelten Linie gesetzt, die ihre Dauer anzeigt. Diese Linie kann auf folgende Weise unterdrückt werden:
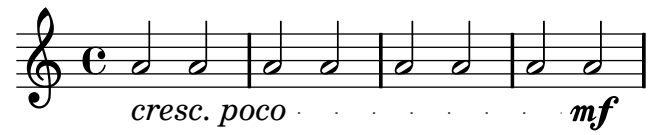
```
\relative c'' {
  \override DynamicTextSpanner #'style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```



Text und Strecker-Stile für Dynamik-Texte ändern

Der Text, der für Crescendo und Decrescendo gesetzt wird, kann geändert werden, indem man die Eigenschaften `crescendoText` und `decrescendoText` verändert. Der Stil des Streckers kann auch geändert werden, indem die `'style`-Eigenschaft des `DynamicTextSpanner` beeinflusst wird. Der Standardwert ist `'hairpin`, andere Möglichkeiten sind `'line`, `'dashed-line` und `'dotted-line`.

```
\relative c'' {
  \set crescendoText = \markup { \italic { cresc. poco } }
  \set crescendoSpanner = #'text
  \override DynamicTextSpanner #'style = #'dotted-line
  a2\< a
  a2 a
  a2 a
  a2 a\mf
}
```



Siehe auch

Glossar: Abschnitt “al niente” in *Glossar*, Abschnitt “crescendo” in *Glossar*, Abschnitt “decrescendo” in *Glossar*, Abschnitt “hairpin” in *Glossar*. Handbuch zum Lernen: Abschnitt “Artikulationszeichen und Lautstärke” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538, [Neue Lautstärkezeichen], Seite 114, Abschnitt 3.5.3 [Was geht in die MIDI-Ausgabe], Seite 454, Abschnitt 3.5.5 [MIDI-Lautstärke kontrollieren], Seite 455.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.
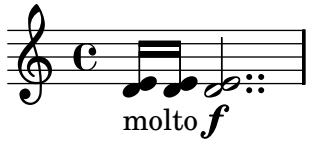
Referenz der Interna: Abschnitt “DynamicText” in *Referenz der Interna*, Abschnitt “Hairpin” in *Referenz der Interna*, Abschnitt “DynamicLineSpanner” in *Referenz der Interna*, Abschnitt “Dynamics” in *Referenz der Interna*.

Neue Lautstärkezeichen

Die einfachste Art, eigene Dynamikbezeichnungen zu erstellen, ist die Benutzung von `\markup`-(Textbeschriftungs)-Objekten.

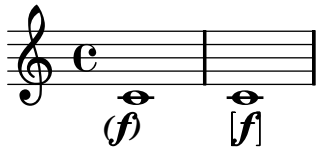
```
moltoF = \markup { molto \dynamic f }

\relative c' {
  <d e>16_\moltoF <d e>
  <d e>2..
}
```
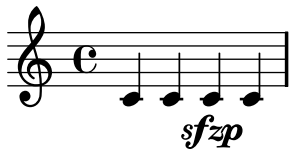


Mit einer Textbeschriftung können editorische Dynamikzeichen (in runden oder eckigen Klammern) erstellt werden. Die Syntax für den Textbeschriftungsmodus wird erklärt in Abschnitt 1.8.2 [Text formatieren], Seite 210.

```
roundF = \markup {
    \center-align \concat { \bold { \italic ( }
           \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative c' {
  c1_\roundF
  c1_\boxF
}
```



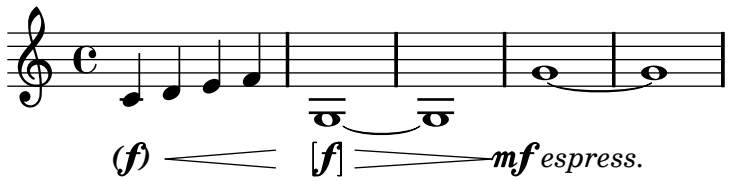
Einfache, mittig gesetzte Dynamikzeichen können schnell mit der `make-dynamic-script`-Funktion erstellt werden.

```
sfzp = #(make-dynamic-script "sfzp")
\relative c' {
  c4 c c\sfzp c
}
```



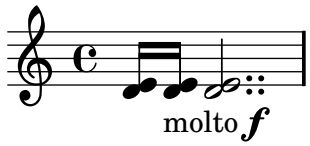
Allgemein gesagt kann `make-dynamic-script` jegliches Textbeschriftungsobjekt als Argument haben. Die Schriftart für Dynamikzeichen enthält nur die Buchstaben `f,m,p,r,s` sowie `z`; ein Dynamikzeichen, das anderen Text oder Satzzeichen enthalten soll, benötigt Textbeschriftungsbefehle, die die normale Schriftart einschalten, etwa `\normal-text`. Die Funktion `make-dynamic-script` sollte anstelle einer normalen Textbeschriftung vor allem deshalb benutzt werden, weil auf diese Weise die vertikale Ausrichtung von den Textbeschriftungen (engl. markup) und den spitzen Klammern an der selben Linie gewährleistet wird.

```
roundF = \markup { \center-align \concat {
           \normal-text { \bold { \italic ( } }
           \dynamic f
           \normal-text { \bold { \italic ) } } } }
boxF = \markup { \bracket { \dynamic f } }
mfEspress = \markup { \center-align \line {
              \hspace #3.7 mf \normal-text \italic espress. } }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative c' {
  c4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g1
  g'1~\mfEspressDynamic
  g1
}
```



Anstelle dessen kann auch die Scheme-Form des Beschriftungs-Modus verwendet werden. Seine Syntax ist erklärt in Abschnitt “Beschriftungskonstruktionen in Scheme” in *Extending*.

```
moltoF = #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative c' {
  <d e>16 <d e>
  <d e>2..\moltoF
}
```



Die Auswahl von Schriftarten in Textbeschriftungen ist erklärt in [Überblick über die wichtigsten Textbeschriftungsbefehle], Seite 211.

Siehe auch

Notationsreferenz: Abschnitt 1.8.2 [Text formatieren], Seite 210, [Überblick über die wichtigsten Textbeschriftungsbefehle], Seite 211, Abschnitt 3.5.3 [Was geht in die MIDI-Ausgabe], Seite 454, Abschnitt 3.5.5 [MIDI-Lautstärke kontrollieren], Seite 455.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Erweitert: ⟨undefined⟩ [Beschriftungskonstruktionen in Scheme], Seite ⟨undefined⟩.

1.3.2 Ausdrucksbezeichnungen als Bögen
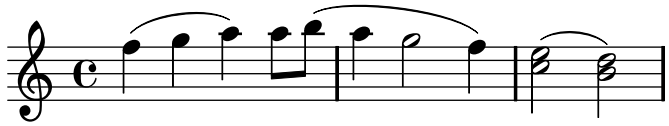
Dieser Abschnitt erklärt, wie man verschiedene gebogene Ausdrucksbezeichnungen erstellt: Legato- und Phrasierungsbögen, Atemzeichen und Glissandos zu unbestimmten Tonhöhen.

Legatobögen

Ein Legatobogen (engl. slur) zeigt an, dass die Noten *legato* gespielt werden sollen. Er wird mit Klammern hinter den Notenwerten notiert.

Achtung: In polyphoner Musik muss ein Legatobogen in der gleichen Stimme beendet werden, in der er begonnen wurde.

```
f4( g a) a8 b(
a4 g2 f4)
<c e>2( <b d>2)
```
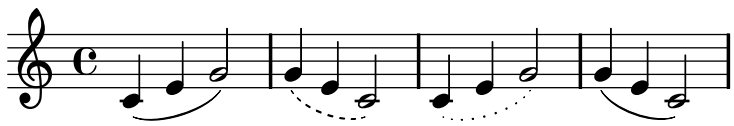


Legatobögen können manuell ober- oder unterhalb des Notensystems besetzt werden, siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

Gleichzeitige, überlappende Legatobögen sind nicht erlaubt, aber ein Phrasierungsbogen kann einen Legatobogen überlappen. Damit können zwei Bögen gleichzeitig ausgegeben werden. Siehe auch [Phrasierungsbögen], Seite 119.
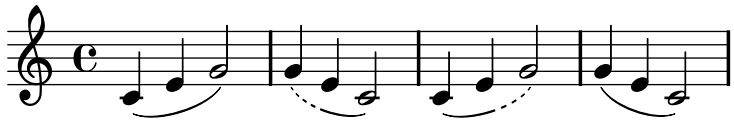
Legatobögen können durchgehend, gepunktet oder gestrichelt dargestellt werden. Standard ist der durchgehende Bogen:

```
c4( e g2)
\slurDashed
g4( e c2)
\slurDotted
c4( e g2)
\slurSolid
g4( e c2)
```
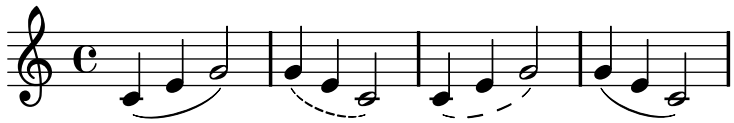


Bögen können auch halb gestrichelt (die erste Hälfte gestrichelt, die zweite Hälfte durchgehend) erstellt werden, oder als halb durchgehend (die erste Hälfte durchgehend, die zweite Hälfte gestrichelt:

```
c4( e g2)
\slurHalfDashed
g4( e c2)
\slurHalfSolid
c4( e g2)
\slurSolid
g4( e c2)
```



Eigene Muster für die Strichelung können definiert werden:

```
c4( e g2)
\slurDashPattern #0.7 #0.75
g4( e c2)
\slurDashPattern #0.5 #2.0
c4( e g2)
\slurSolid
g4( e c2)
```



Vordefinierte Befehle

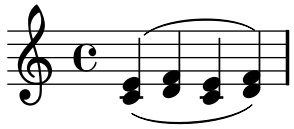
`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`, `\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

Ausgewählte Schnipsel

Doppelte Bögen für Legato-Akkorde benutzen

Einige Komponisten schreiben doppelte Bögen, wenn Legato-Akkorde notiert werden. Das kann mit der Eigenschaft `doubleSlurs` erreicht werden.

```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```
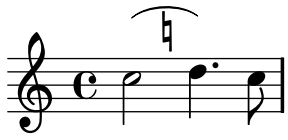


Textbeschriftung innerhalb von Bögen positionieren

Textbeschriftung kann innerhalb von Bögen gesetzt werden, wenn die `outside-staff-priority`-Eigenschaft auf falsch gesetzt wird.

```
\relative c'' {
  \override TextScript #'avoid-slur = #'inside
  \override TextScript #'outside-staff-priority = ##f
```

```
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```
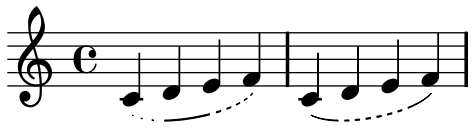


*Legatobögen mit kompliziertem Strichelmuster*definieren

Legatobögen können mit einem komplizierten Strichelmuster gesetzt werden, indem die `dash-definition`-Eigenschaft definiert wird. `dash-definition` ist eine Liste bestehend aus `dash-elements`-Elementen. Ein `dash-element` ist eine Liste an Parametern, die das Strichverhalten für einen Abschnitt des Legatobogens definieren.

Der Bogen wird nach dem Bezierparameter t definiert, welcher von 0 am linken Ende des Bogens zu 1 am rechten Ende des Bogens reicht. `dash-element` ist eine Liste `(start-t stop-t dash-Unterbrechung dash-Abschnitt)`. Die Region des Bogens von `start-t` bis `stop-t` hat eine Unterbrechung von `dash-Unterbrechung` von jedem `dash-Abschnitt`-Schwarzabschnitt. `dash-Abschnitt` ist in Notenlinienzwischenräumen definiert. `dash-Abschnitt` ist auf 1 für einen durchgehenden Bogen gesetzt.

```
\relative c' {
  \once \override
    Slur #'dash-definition = #'((0 0.3 0.1 0.75)
                                (0.3 0.6 1 1)
                                (0.65 1.0 0.4 0.75))
  c4( d e f)
  \once \override
    Slur #'dash-definition = #'((0 0.25 1 1)
                                (0.3 0.7 0.4 0.75)
                                (0.75 1.0 1 1))
  c4( d e f)
}
```



Siehe auch

Glossar: Abschnitt “slur” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Über die Nicht-Schachtelung von Klammern und Bindebögen” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538, [Phrasierungsbögen], Seite 119.
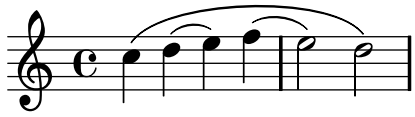
Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “Slur” in *Referenz der Interna*.

Phrasierungsbögen

Ein Phrasierungsbogen verbindet Noten und wird verwendet, um einen musikalischen Ausdruck anzuzeigen. Er wird mit den Befehlen `\(` und `\)` eingegeben.

```
c4\( d( e) f(
e2) d\)
```
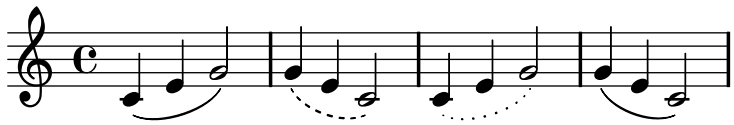


Im typographischen Sinne verhalten sich Phrasierungsbögen genauso wie Legatobögen. Sie werden aber als eigene Objekte behandelt. Ein `\slurUp` hat also keine Auswirkung auf die Phrasierungsbögen. Phrasierungsbögen können manuell oberhalb oder unterhalb des Notensystems gesetzt werden, siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

Simultane oder überlappende Phrasierungsbögen sind nicht erlaubt.

Phrasierungsbögen können durchgehend, gepunktet oder gestrichelt dargestellt werden. Standard ist der durchgehende Bogen:
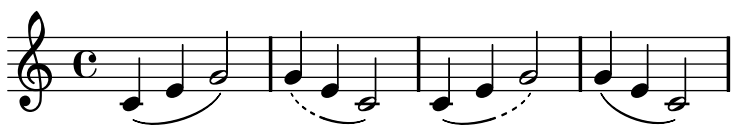
```
c4\( e g2\)
\phrasingSlurDashed
g4\( e c2\)
\phrasingSlurDotted
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```



funindex phrasingSlurHalfDashed

Phrasierungsbögen können auch als halbgestrichelt dargestellt werden (die erste Hälfte gestrichelt, die zweite Hälfte durchgehend, oder halb durchgehend (die erste Hälfte durchgehend, die zweite gestrichelt):
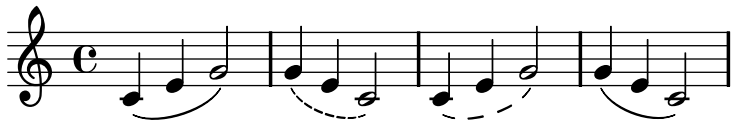
```
c4\( e g2\)
\phrasingSlurHalfDashed
g4\( e c2\)
\phrasingSlurHalfSolid
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```



Eigene Strichelmuster für Phrasierungsbögen können definiert werden:

```
c4\( e g2\)
\phrasingSlurDashPattern #0.7 #0.75
g4\( e c2\)
```

```
\phrasingSlurDashPattern #0.5 #2.0
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```



Strichelmusterdefinitionen für Phrasierungsbögen haben die gleiche Struktur wie die Definitionen für Legatobögen. Zu mehr Information über komplizierte Strichelmuster, siehe die Schnipsel im Abschnitt [Legatobögen], Seite 116.

Vordefinierte Befehle

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`, `\phrasingSlurDashed`, `\phrasingSlurDotted`, `\phrasingSlurHalfDashed`, `\phrasingSlurHalfSolid`, `\phrasingSlurDashPattern`, `\phrasingSlurSolid`.

Siehe auch

Handbuch zum Lernen: Abschnitt “Über die Nicht-Schachtelung von Klammern und Bindebögen” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.
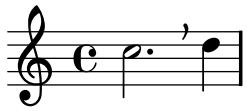
Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “PhrasingSlur” in *Referenz der Interna*.

Atemzeichen
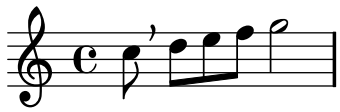
Atemzeichen werden mit dem Befehl `\breathe` eingegeben.

```
c2. \breathe d4
```



Ein Atemzeichen bezeichnet gleichzeitig das Ende eines automatischen Balkens. Um das Verhalten zu verändern siehe [Manuelle Balken], Seite 84.

```
c8 \breathe d e f g2
```



Musikalische Zeichen für Atemzeichen in Alter Notation, auch Divisiones genannt, sind unterstützt. Für Einzelheiten siehe [Divisiones], Seite 405.

Ausgewählte Schnipsel

Das Atemzeichen-Symbol verändern

Das Schriftzeichen für das Atemzeichen kann verändert werden, indem die Text-Eigenschaft des `BreathingSign`-Layoutobjekts mit einer beliebigen Textbeschriftung definiert wird.

```
\relative c'' {
  c2
  \override BreathingSign #'text = \markup { \musicglyph #"scripts.rvarcomma" }
  \breathe
  d2
}
```
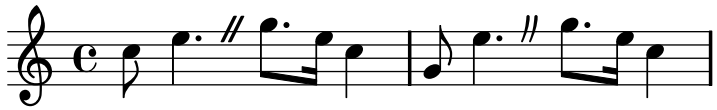


Eine Zäsur einfügen

Zäsurzeichen können erstellt werden, indem die `'text`-Eigenschaft des `BreathingSign`-Objektes verändert wird. Ein gekrümmtes Zäsurzeichen ist auch möglich.

```
\relative c'' {
  \override BreathingSign #'text = \markup {
    \musicglyph #"scripts.caesura.straight"
  }
  c8 e4. \breathe g8. e16 c4

  \override BreathingSign #'text = \markup {
    \musicglyph #"scripts.caesura.curved"
  }
  g8 e'4. \breathe g8. e16 c4
}
```



Siehe auch

Glossar: Abschnitt “caesura” in *Glossar*.

Notationsreferenz: [Divisiones], Seite 405.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “BreathingEvent” in *Referenz der Interna*, Abschnitt “BreathingSign” in *Referenz der Interna*, Abschnitt “Breathing_sign_engraver” in *Referenz der Interna*.

Glissando zu unbestimmter Tonhöhe

Gleiten nach oben und unten kann mit dem Befehl `\bendAfter` notiert werden. Die Richtung des Glissandos wird mit einem Plus oder Minus (nach oben bzw. nach unten) angezeigt. Die Zahl zeigt die Intervallgröße an, über die sich das Glissando *nach* der Note erstreckt.

```
c2-\bendAfter #+4
c2-\bendAfter #-4
c2-\bendAfter #+6.5
c2-\bendAfter #-6.5
c2-\bendAfter #+8
c2-\bendAfter #-8
```



Das Minuszeichen (-) direkt vor dem `\bendAfter`-Befehl ist *notwendig* um unbestimmte Glissandos zu notieren.

Ausgewählte Schnipsel

Das Aussehen von unbestimmten Glissandi anpassen

Die `shortest-duration-space`-Eigenschaft kann verändert werden, um das Aussehen von unbestimmten Glissandi anzupassen.

```
\relative c'' {
  \override Score.SpacingSpanner #'shortest-duration-space = #4.0
  c2-\bendAfter #5
  c2-\bendAfter #-4.75
  c2-\bendAfter #8.5
  c2-\bendAfter #-6
}
```



Siehe auch

Glossar: Abschnitt “fall” in *Glossar*, Abschnitt “doit” in *Glossar*.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.
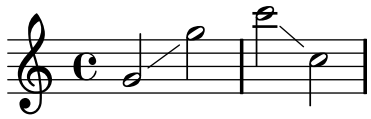
1.3.3 Ausdrucksbezeichnungen als Linien

Dieser Abschnitt zeigt, wie man verschiedene Ausdrucksbezeichnungen erstellt, die sich linear erstrecken: Glissando, Arpeggio und Triller.

Glissando

Ein *Glissando* wird mit dem Befehl `\glissando` auf eine Note folgend notiert:

```
g2\glissando g'
c2\glissando c,
```



Verschiedene Glissando-Stile sind möglich. Für Einzelheiten siehe Abschnitt 5.4.8 [Linienstile], Seite 551.
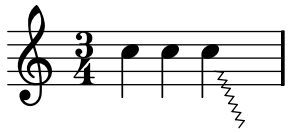
Ausgewählte Schnipsel

Moderne Glissandi

Ein modernes Glissando ohne eine Endnote kann gesetzt werden, indem eine Kadenz eingesetzt wird und die Endnote unsichtbar gemacht wird.

```
\relative c'' {
  \time 3/4
  \override Glissando #'style = #'zigzag
```

```
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
  \bar "|"
}
```



Siehe auch

Glossar: Abschnitt “glissando” in *Glossar*.

Notationsreferenz: Abschnitt 5.4.8 [Linienstile], Seite 551.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “Glissando” in *Referenz der Interna*.

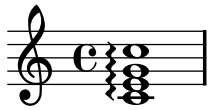
Bekannte Probleme und Warnungen

Printing text over the line (such as *gliss.*) is not supported.

Arpeggio

Ein *Arpeggio* als Zeichen, dass ein Akkord gebrochen gespielt werden soll, kann mit dem Befehl `\arpeggio` hinter der Akkord-Konstruktion erzeugt werden.

```
<c e g c>1\arpeggio
```
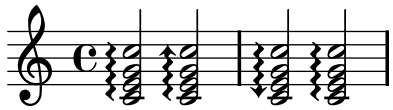


Unterschiedliche Arpeggio-Typen können benutzt werden. `\arpeggioNormal` stellt wieder das normale Verhalten her:

```
<c e g c>2\arpeggio

\arpeggioArrowUp
<c e g c>2\arpeggio

\arpeggioArrowDown
<c e g c>2\arpeggio

\arpeggioNormal
<c e g c>2\arpeggio
```



Besondere Arpeggios mit Klammern können erstellt werden:

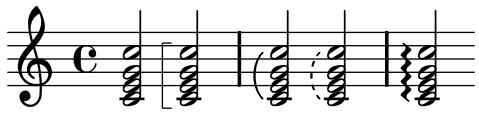
```
<c e g c>2

\arpeggioBracket
<c e g c>2\arpeggio

\arpeggioParenthesis
<c e g c>2\arpeggio

\arpeggioParenthesisDashed
<c e g c>2\arpeggio

\arpeggioNormal
<c e g c>2\arpeggio
```



Die `dash`-Eigenschaft der Arpeggioklammern werden von der `'dash-details`-Eigenschaft kontrolliert, die beschrieben ist in [Legatobögen], Seite 116.

Ein Arpeggio kann auch explizit ausgeschrieben werden, indem Überbindungsbögen benutzt werden. Für mehr Information siehe [Bindebögen], Seite 48.

Vordefinierte Befehle

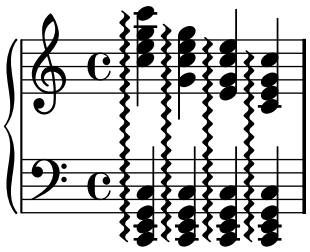
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis`, `\arpeggioParenthesisDashed`.

Ausgewählte Schnipsel

Arpeggio über mehrere Systeme in anderen Kontexten

Arpeggio über mehrere Systeme können in anderen Kontexten als dem `PianoStaff` erstellt werden, wenn der `Span_arpeggio_engraver` in den `Score`-Kontext eingefügt wird.

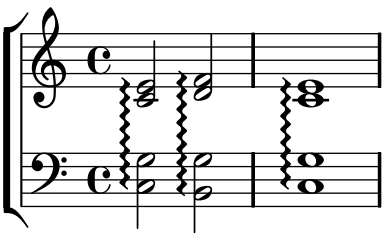
```
\new PianoStaff \relative c'' <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff {
    <c e g c>4\arpeggio
    <g c e g>4\arpeggio
    <e g c e>4\arpeggio
    <c e g c>4\arpeggio
  }
  \new Staff {
    \clef bass
    \repeat unfold 4 {
      <c,, e g c>4\arpeggio
    }
  }
>>
```



Arpeggio zwischen Systemen in einem Klaviersystem erstellen

In einem Klaviersystem (`PianoStaff`) ist es möglich, ein Arpeggio zwischen beiden Systemen zu verbinden, indem die `PianoStaff.connectArpeggios`-Eigenschaft gesetzt wird.

```
\score {
  \new ChoirStaff {
    \set Score.connectArpeggios = ##t
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
  }
  \layout {
    \context {
      \Score
      \consists "Span_arpeggio_engraver"
    }
  }
}
```

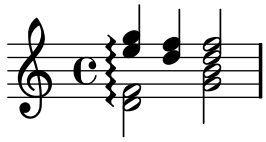


Arpeggios zwischen unterschiedlichen Stimmen erzeugen

Ein Arpeggio kann zwischen Noten aus unterschiedlichen Stimmen auf demselben System gezogen werden, wenn der `Span_arpeggio_engraver` in den `Staff`-Kontext verschoben wird:

```
\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' {
  \set Staff.connectArpeggios = ##t
  <<
```

```
    { <e' g>4\arpeggio <d f> <d f>2 }
    \\
    { <d, f>2\arpeggio <g b>2 }
  >>
}
```



Siehe auch

Glossar: Abschnitt “arpeggio” in *Glossar*.

Notationsreferenz: [Legatobögen], Seite 116, [Bindebögen], Seite 48.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “Arpeggio” in *Referenz der Interna*, Abschnitt “Slur” in *Referenz der Interna*, Abschnitt “PianoStaff” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Es ist nicht möglich, Arpeggios zwischen Systemen und solche, die sich nur auf ein System erstrecken, zum gleichen Zeitpunkt in einem Klaviersystem (`PianoStaff`) zu benutzen.

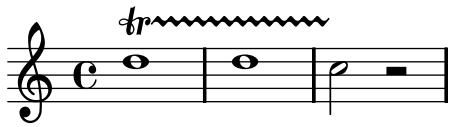
Die Arpeggios im Klammer-Stil funktionieren nicht über mehrere Notensysteme.

Triller

Kurze Triller ohne eine Dauer werden mit dem Befehl `\trill` notiert, siehe auch [Artikulationszeichen und Verzierungen], Seite 106.

Längere Triller mit einer Dauer werden mit den Befehlen `\startTrillSpan` zu Beginn und `\stopTrillSpan` am Ende erstellt.

```
d1\startTrillSpan
d1
c2\stopTrillSpan
r2
```



Ein Triller-Strekcer, der über einen Zeilenumbruch geht, beginnt genau über der ersten Note auf der neue Zeile erneut.

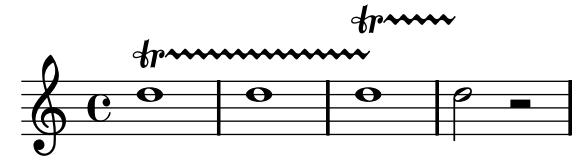
```
d1\startTrillSpan
\break
d1
c2\stopTrillSpan
r2
```
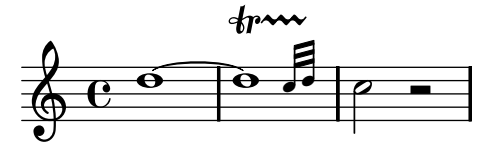




Aufeinanderfolgende Trillerstrecker funktieren ohne einen `+\stopTrillSpan`-Befehl, weil ein folgender Strecker automatisch die rechte Begrenzung des vorhergehenden beendet.

```
d1\startTrillSpan
d1
d1\startTrillSpan
d2\stopTrillSpan
r2
```



Triller können auch mit Vorschlägen kombiniert werden. Die Syntax für diese Konstruktion und die Methode, um die Position der Vorschläge präzise zu positionieren, wird gezeigt in [Verzierungen], Seite 99.

```
d1~\afterGrace
d1\startTrillSpan { c32[ d]\stopTrillSpan }
c2 r2
```



Triller, die auf einer bestimmten Note ausgeführt werden sollen, können mit dem Befehl `pitchedTrill` notiert werden. Das erste Argument ist die Hauptnote, das zweite die Note, auf der getrillert wird. Sie wird als Note ohne Hals in Klammern ausgegeben.

```
\pitchedTrill
d2\startTrillSpan fis
d2
c2\stopTrillSpan
r2
```



Aufeinanderfolgende Versetzungszeichen der selben Note im selben Takt müssen selbst hinzugefügt werden. Nur das Versetzungszeichen des ersten Trillers mit Tonhöhe innerhalb eines Taktes wird ausgegeben.

```
\pitchedTrill
eis4\startTrillSpan fis
eis4\stopTrillSpan
\pitchedTrill
eis4\startTrillSpan cis
eis4\stopTrillSpan
\pitchedTrill
eis4\startTrillSpan fis
```

```
eis4\stopTrillSpan
\pitchedTrill
eis4\startTrillSpan fis!
eis4\stopTrillSpan
```



Vordefinierte Befehle

`\startTrillSpan`, `\stopTrillSpan`.

Siehe auch
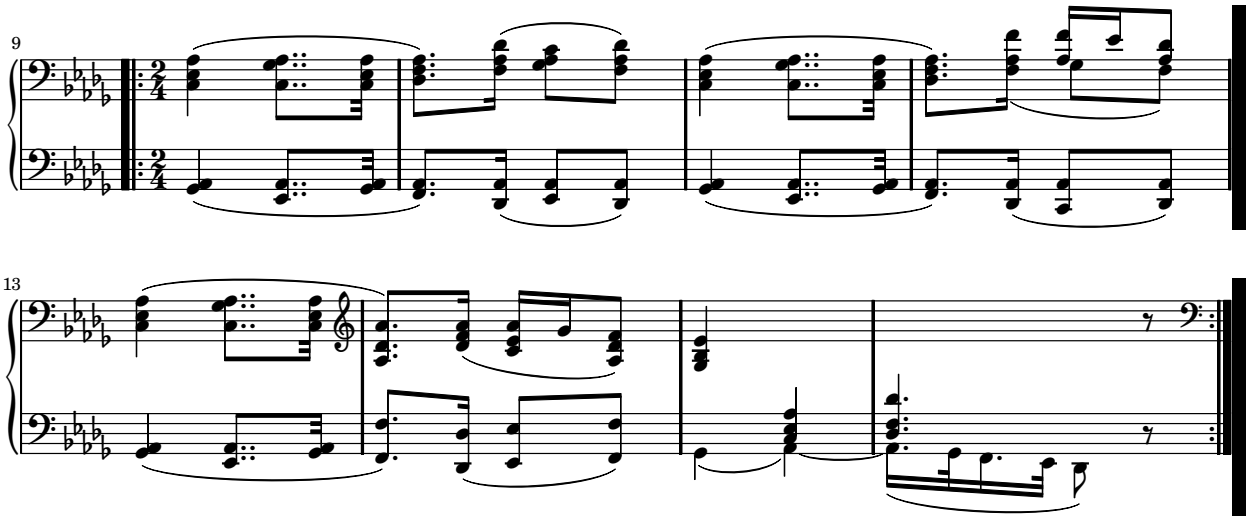
Glossar: Abschnitt “trill” in *Glossar*.

Notationsreferenz: [Artikulationszeichen und Verzierungen], Seite 106, [Verzierungen], Seite 99.

Schnipsel: Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “TrillSpanner” in *Referenz der Interna*.

1.4 Wiederholungszeichen



Wiederholung ist ein zentrales Konzept in der Musik, und es gibt eine ganze Vielzahl von Notationsmöglichkeiten für Wiederholungen. LilyPond unterstützt folgende Arten von Wiederholungen:

`volta (Wiederholungsklammer)`
: Die wiederholte Musik wird nicht geschrieben, sondern zwischen zwei Wiederholungstaktstrichen eingeschlossen. Wenn die Wiederholung am Anfang eines Stückes beginnt, wird nur am Ende der Wiederholung eine Wiederholungstaktlinie gesetzt. Alternative Schlüsse (Volta) werden von links nach rechts mit Klammern gesetzt. Das ist die Standardnotationspraxis für Wiederholungen mit alternativen Schlüssen.

`unfold (aufklappen)`
: Die wiederholte Musik wird ausgeschrieben, so oft, wie es durch *`Wiederholungszähler`* definiert wird. Das erspart Arbeit, wenn repetitive Musik notiert wird.

`percent (Prozent-Wiederholung)`
: Das sind Noten- oder Taktwiederholungen, sie sehen aus wie ein Schrägstrich bzw. wie ein Prozentzeichen.

`tremolo` Das wird benutzt, um Tremolo-Wiederholungen am Notenhals zu notieren.

1.4.1 Lange Wiederholungen

Normale Wiederholungen
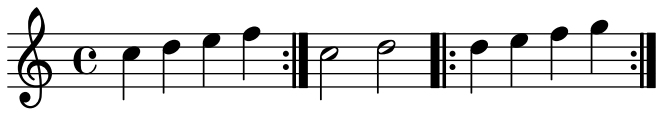
Die Syntax für normale Wiederholungen ist

`\repeat` *`Typ Wiederholungszähler musikAusdr`*

wobei *`musikAusdr`* ein musikalischer Ausdruck ist.

Wiederholung ohne alternativen Schluß:

```
\repeat volta 2 { c4 d e f }
c2 d
\repeat volta 2 { d4 e f g }
```



Alternative Schlüsse können mit `\alternative` gesetzt werden. Damit die alternativen Schlüsse von den wiederholten Noten abgegrenzt werden, müssen sie in geschweiften Klammern zusammengefasst werden.
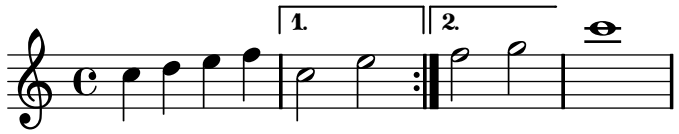
```
\repeat volta Wiederholungszähler musikAusdr
\alternative {
  { musikAusdr }
}
```

wobei *`musikAusdr`* ein musikalischer Ausdruck ist.

Wenn es mehr Wiederholungen gibt, als Alternativen angegeben sind, erhalten die ersten Wiederholungen den ersten Schluss.
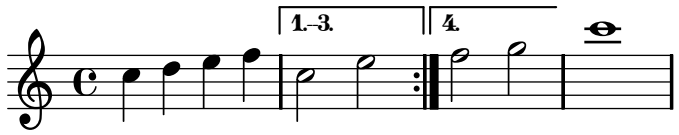
Eine einfache Wiederholung mit einer Alternative:

```
\repeat volta 2 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
}
c1
```



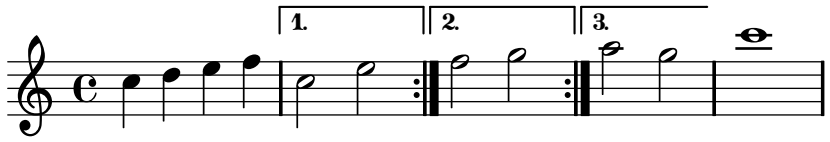
Eine einfache Wiederholung mit mehr als einer Alternative:

```
\repeat volta 4 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
}
c1
```



Mehrfache Wiederholungen mit mehr als einer Alternative:

```
\repeat volta 3 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
  { a2 g | }
}
c1
```
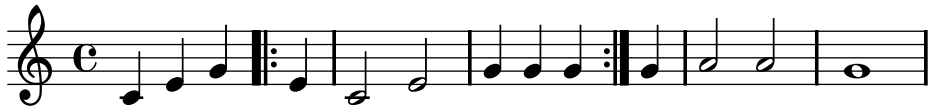


Achtung: Wenn es zwei oder mehr Alternativen gibt, darf nicht zwischen der schließenden Klammer der einen und der öffnenden Klammer der anderen Wiederholung stehen, weil sonst nicht die erwartete Anzahl von Endungen produziert wird.

Achtung: Wenn man `\relative` innerhalb von `\repeat` notiert, ohne den `Voice`-Kontext explizit zu beginnen, erscheinen zusätzliche (ungewollte) Systeme. Sie auch Abschnitt “Ein zusätzliches System erscheint” in *Anwendungsbenutzung*.
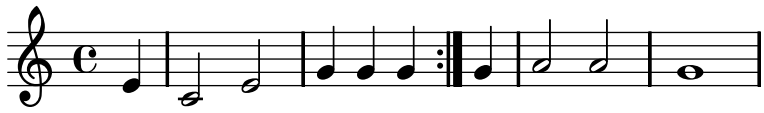
Wenn eine Wiederholung mitten in einem Takt beginnt und keine Alternativen hat, fällt normalerweise auch das Ende der Wiederholung mitten in einen Takt, sodass beide unvollständigen Takt einen vollständigen Takt ergeben. In diesem Fall bezeichnen die Wiederholungsstriche keine richtigen Taktstriche. Benutzen Sie nicht `\partial`-Befehle oder Taktüberprüfung, wo die Wiederholungslinien gesetzt werden:

```
% no \partial here
c4 e g  % no bar check here
% no \partial here
\repeat volta 4 {
  e4 |
  c2 e |
  % no \partial here
  g4 g g  % no bar check here
}
% no \partial here
g4 |
a2 a |
g1 |
```



Ähnlich ist es, wenn eine Wiederholung mit einem Auftakt beginnt und keine Alternativen hat. In diesem Fall muss man aber den `\partial`-Befehl zu Beginn der Partitur setzen:
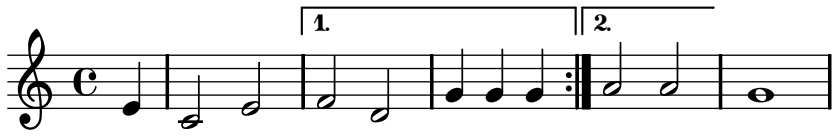
```
\partial 4  % required
\repeat volta 4 {
  e4 |
  c2 e |
  % no \partial here
  g4 g g  % no bar check here
}
% no \partial here
g4 |
a2 a |
g1 |
```



Wenn alternative Endungen zu einer Wiederholung hinzugefügt werden, die mit einem Auftakt beginnt, muss die `Timing.measureLength`-Eigenschaft manuell gesetzt werden, und an folgenden Stellen:

- am Beginn eines unvollständigen Taktes innerhalb der `\alternative`-Umgebung, die normalerweise am Ende jeder Alternative auftreten, außer (in den meisten Fällen) in der letzten.
- zu Beginn jeder Alternative außer der ersten.

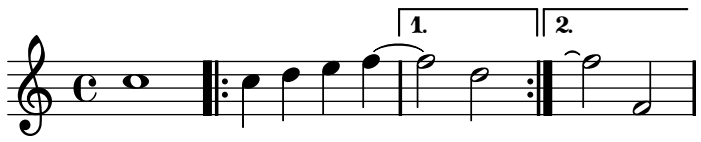
```
\partial 4
\repeat volta 2 { e4 | c2 e | }
\alternative {
  {
    f2 d |
    \set Timing.measureLength = #(ly:make-moment 3 4)
    g4 g g  % optional bar check is allowed here
  }
  {
    \set Timing.measureLength = #(ly:make-moment 4 4)
    a2 a |
  }
}
g1 |
```



Die `measureLength`-Eigenschaft ist beschrieben in [Verwaltung der Zeiteinheiten], Seite 105.

Bindebögen können auch an eine zweite Klammer angefügt werden:

```
c1
\repeat volta 2 { c4 d e f~ }
\alternative {
  { f2 d }
  { f2\repeatTie f, }
}
```
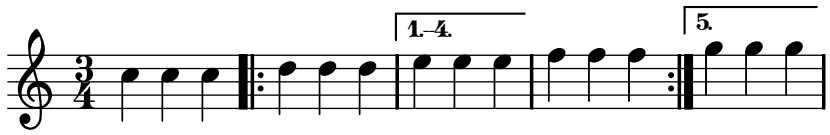


Ausgewählte Schnipsel

Volta-Klammern verkürzen

Volta-Klammern werden normalerweise über alle Noten der Klammer gezogen, aber es ist möglich sie zu verkürzen. Hierzu muss `voltaSpannerDuration` definiert werden, in dem Beispiel etwa als 3/4, sodass die Klammer nur einen Takt dauert.

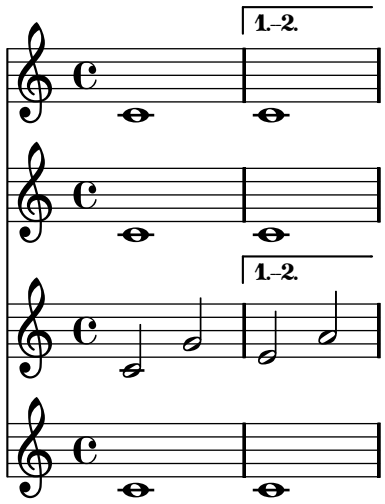
```
\relative c'' {
  \time 3/4
  c4 c c
  \set Score.voltaSpannerDuration = #(ly:make-moment 3 4)
  \repeat volta 5 { d4 d d }
  \alternative {
    {
      e4 e e
      f4 f f
    }
    { g4 g g }
  }
}
```



Volta-Klammern zu zusätzlichen Systemen hinzufügen

Der `Volta_engraver` befindet sich im `Score`-Kontext und Klammern werden deshalb nur auf dem obersten System dargestellt. Das kann umgangen werden, indem man den `Volta_engraver` zu dem `Staff`-Kontext hinzufügt, in dem die Klammern zusätzlichen vorkommen sollen. Siehe auch das "Volta multi staff"-Schnipsel.

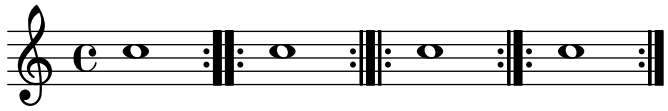
```
<<
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
>>
```



Setting the double repeat default for volte

There are three different styles of double repeats for volte, that can be set using `doubleRepeatType`.

```
\relative c'' {
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.:"
  \repeat volta 1 { c1 }
}
```



Siehe auch

Glossar: Abschnitt “repeat” in *Glossar*, Abschnitt “volta” in *Glossar*.

Notationsreferenz: [Taktstriche], Seite 87, Abschnitt 5.1.4 [Umgebungs-Plugins verändern], Seite 519, [Verwaltung der Zeiteinheiten], Seite 105.

Schnipsel: Abschnitt “Repeats” in *Schnipsel*.

Referenz der Interna: Abschnitt “VoltaBracket” in *Referenz der Interna*, Abschnitt “RepeatedMusic” in *Referenz der Interna*, Abschnitt “VoltaRepeatedMusic” in *Referenz der Interna*, Abschnitt “UnfoldedRepeatedMusic” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Bindebögen, die von einer `\repeat`-Umgebung in eine `\alternative`-Umgebung ragen, funktionieren nur in der ersten Klammer. Bindebögen können auch nicht von der Ende einer Wiederholungsklammer auf den Anfang der Wiederholung verweisen.

Wenn eine Wiederholung innerhalt eines unvollständigen Taktes beginnt und eine `\alternative`-Umgebung mit einer Veränderung von `measureLength` enghält, führt die Verwendung von `\unfoldRepeats` zu falsch gesetzten Taktstrichen und Taktüberprüfungswarnungen.

Eine ineinandergeschachtelte Wiederholung wie

```
\repeat ...
\repeat ...
```

`\alternative`
ist mehrdeutig, weil nicht klar ist, zu welchem `\repeat`-Abschnitt die `\alternative`-Endung gehört. Diese Mehrdeutigkeit wird von LilyPond aufgelöst, indem die alternative Endung immer zu der innersten Wiederholung gehört. Um Klarheit zu schaffen, bietet es sich an, in solchen Situationen Klammern zu benutzen.
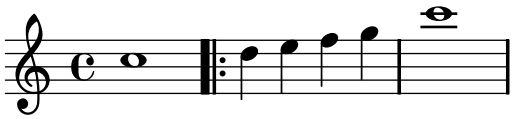
Manuelle Wiederholungszeichen

Achtung: Diese Methoden werden nur verwendet, um ungewöhnliche Wiederholungskonstruktionen darzustellen und können sich unerwünscht verhalten. In den meisten Fällen sollten Wiederholungen mit dem Befehl `\repeat` erstellt werden oder indem die entsprechenden Taktstriche eingegeben werden. Mehr Information in [Taktstriche], Seite 87.

Die Eigenschaft `repeatCommands` kann verwendet werden, um das Aussehen der Wiederholungen zu beeinflussen. Ihr Argument ist eine Scheme-Liste an Wiederholungsbefehlen.

`start-repeat`
: Setzt eine `|:` Taktlinie.
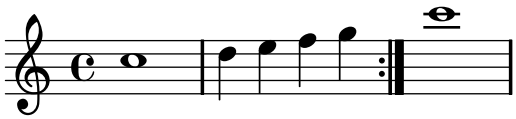
```
c1
\set Score.repeatCommands = #'(start-repeat)
d4 e f g
c1
```



Der Notensatzpraxis folgend werden Wiederholungstaktstriche nicht zu Beginn eines Stückes gesetzt.
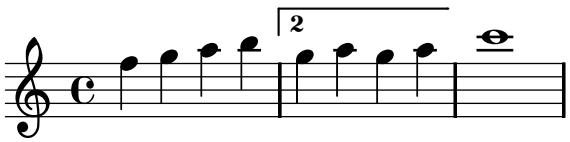
`end-repeat`
: Setzt eine `:|` Taktlinie.

```
c1
d4 e f g
\set Score.repeatCommands = #'(end-repeat)
c1
```



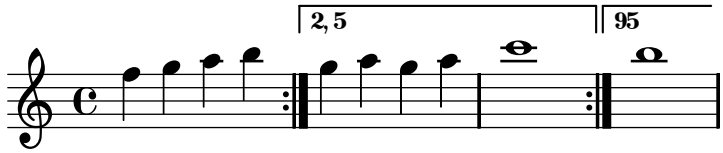
`(volta` *`Zahl`*`) ... (volta #f)`
: Setzt eine Volta-Klammer mit der Beschriftung *Nummer*. Die Volta-Klammer muss explizit beendet werden, sonst wird sie nicht ausgegeben.

```
f4 g a b
\set Score.repeatCommands = #'((volta "2"))
g4 a g a
\set Score.repeatCommands = #'((volta #f))
c1
```
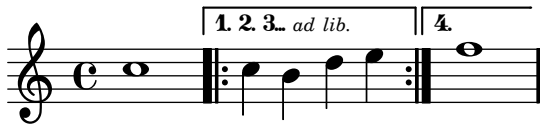


Mehrfache Wiederholungszeichen können an der selben Stelle vorkommen:

```
f4 g a b
\set Score.repeatCommands = #'((volta "2, 5") end-repeat)
g4 a g a
c1
\set Score.repeatCommands = #'((volta #f) (volta "95") end-repeat)
b1
\set Score.repeatCommands = #'((volta #f))
```



Text kann auch in der Volta-Klammer gesetzt werden. Der Text kann aus Zahlen oder einer Zahl oder einer Textbeschriftung bestehen, siehe Abschnitt 1.8.2 [Text formatieren], Seite 210. Die einfachste Art Text zu benutzen ist, die Beschriftung zuerst zu definieren und dann die Beschriftung in einer Scheme-Liste einzufügen.

```
voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
\relative c'' {
  c1
  \set Score.repeatCommands = #(list(list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}
```



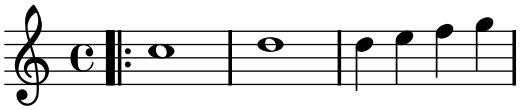
Ausgewählte Schnipsel

Ein Wiederholungszeichen zu Beginn eines Stückes ausgeben

Ein |:-Taktstrich kann auch zu Beginn eines Stückes ausgegeben werden, indem man die entsprechende Eigenschaft verändert:

```
\relative c'' {
  \once \override Score.BreakAlignment #'break-align-orders =
    #(make-vector 3 '(instrument-name
                      left-edge
                      ambitus
                      breathing-sign
                      clef
                      key-signature
                      time-signature
                      staff-bar
                      custos))
  \once \override Staff.TimeSignature #'space-alist =
    #'((first-note . (fixed-space . 2.0))
       (right-edge . (extra-space . 0.5))
```

```
      ;; free up some space between time signature
      ;; and repeat bar line
      (staff-bar . (extra-space . 1)))
  \bar "|:"
  c1
  d1
  d4 e f g
}
```



Siehe auch

Notationsreferenz: [Taktstriche], Seite 87, Abschnitt 1.8.2 [Text formatieren], Seite 210.

Schnipsel: Abschnitt “Repeats” in *Schnipsel*.

Referenz der Interna: Abschnitt “VoltaBracket” in *Referenz der Interna*, Abschnitt “RepeatedMusic” in *Referenz der Interna*, Abschnitt “VoltaRepeatedMusic” in *Referenz der Interna*.
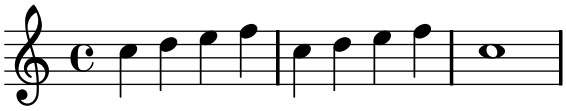
Ausgeschriebene Wiederholungen

Mit dem `unfold`-Befehl können Wiederholungen eingesetzt werden, um repetitive Musik zu notieren. Die Syntax ist

`\repeat unfold` *`Wiederholungszähler musikAusdr`*
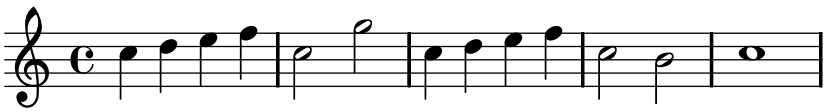
wobei *`musikAusdr`* ein musikalischer Ausdruck ist und *`Wiederholungszähler`* die Anzahl bezeichnet, mit der *`musikAusdr`* wiederholt wird.

```
\repeat unfold 2 { c4 d e f }
c1
```



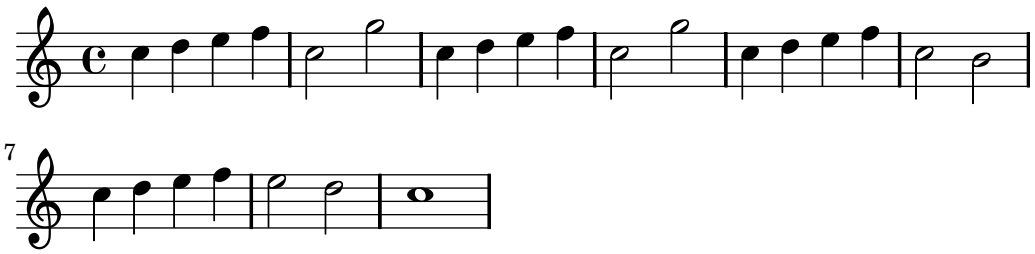
Repititive Wiederholungen können auch mit mehreren Klammern notiert werden:

```
\repeat unfold 2 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
}
c1
```
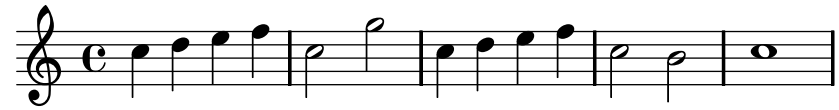


Wenn es mehr Wiederholungen als Alternativen gibt, wird die erste Alternative so oft eingesetzt, bis sich zusammen mit den restlichen Alternativen die Gesamtanzahl der Wiederholungen ergeben.

```
\repeat unfold 4 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
  { e2 d }
 }
c1
```
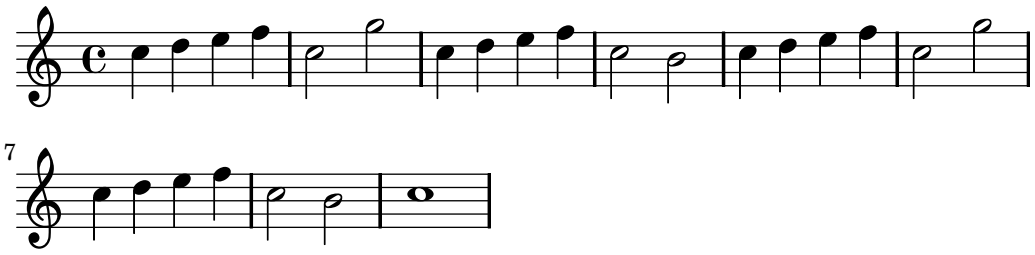


Wenn es mehr Alternativen als Wiederholungen gibt, werden die ersten Alternativen ausgegeben und die restlichen Alternativen ignoriert.

```
\repeat unfold 2 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
  { e2 d }
}
c1
```



Es ist auch möglich, mehrere `unfold`-Wiederholungen (allerdings ohne Alternativen) ineinander zu verschachteln:

```
\repeat unfold 2 {
  \repeat unfold 2 { c4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
  }
}
c1
```



Achtung: Wenn man `\relative` innerhalb von `\repeat` notiert, ohne den `Voice`-Kontext explizit zu beginnen, erscheinen zusätzliche (ungewollte) Systeme. Sie auch Abschnitt “Ein zusätzliches System erscheint” in *Anwendungsbenutzung*.

Siehe auch

Schnipsel: Abschnitt “Repeats” in *Schnipsel*.

Referenz der Interna: Abschnitt “RepeatedMusic” in *Referenz der Interna*, Abschnitt “UnfoldedRepeatedMusic” in *Referenz der Interna*.

1.4.2 Kurze Wiederholungen

Dieser Abschnitt zeigt, wie man kurze Wiederholungen notiert. Kurze Wiederholungen haben zwei Formen: Wiederholungen von einer Note bis zu zwei Takten, die mit Schrägstrichen oder Prozentzeichen dargestellt werden, und Tremolos.

Prozent-Wiederholungen

Kurze wiederholte Muster werden einmal gesetzt und das wiederholte Muster wird durch ein besonderes Zeichen ersetzt.

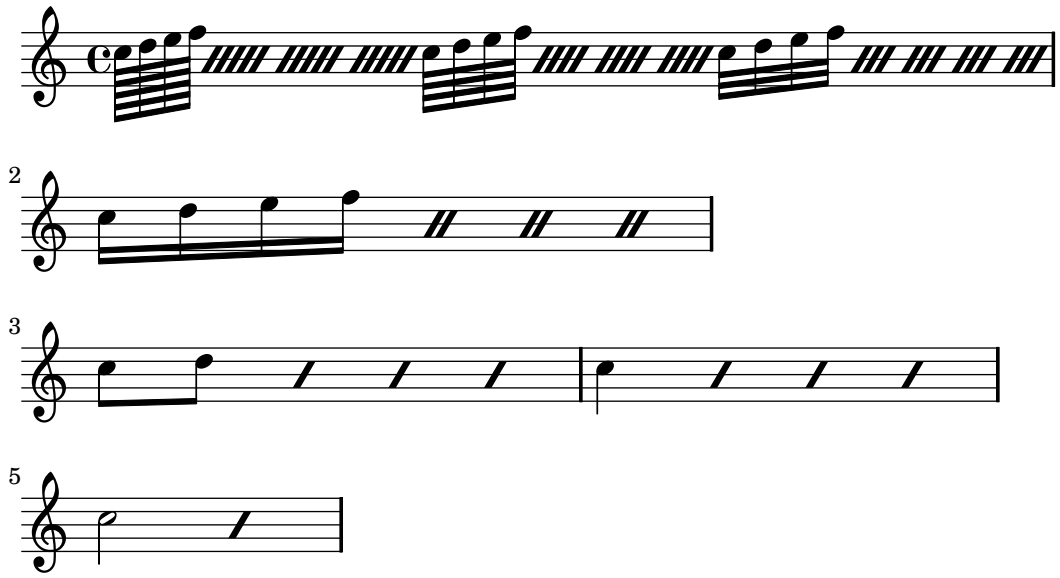
Die Syntax lautet:

`\repeat percent` *`Wiederholungszahl musikAusdr`*

wobei *`musikAusdr`* ein musikalischer Ausdruck ist.
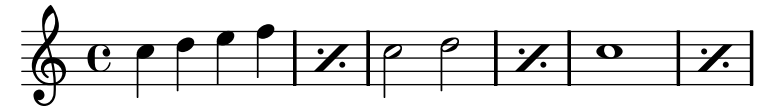
Muster, die kürzer als ein Takt sind, werden mit Schrägstrichen ersetzt:

```
\repeat percent 4 { c128 d e f }
\repeat percent 4 { c64 d e f }
\repeat percent 5 { c32 d e f }
\repeat percent 4 { c16 d e f }
\repeat percent 4 { c8 d }
\repeat percent 4 { c4 }
\repeat percent 2 { c2 }
```
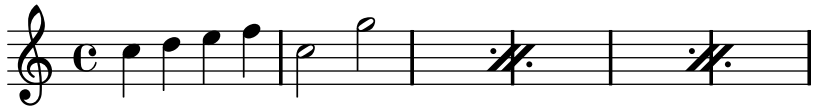


Muster von einem oder zwei Takten Dauer werden mit prozentartigen Symbolen ersetzt:

```
\repeat percent 2 { c4 d e f }
\repeat percent 2 { c2 d }
\repeat percent 2 { c1 }
```

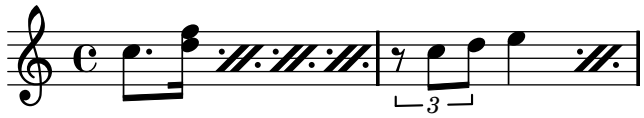


```
\repeat percent 3 { c4 d e f | c2 g' }
```



Muster, die kürzer als ein Takt sind, aber unterschiedliche Dauern beinhalten, benützten ein doppeltes Prozent-Symbol.

```
\repeat percent 4 { c8. <d f>16 }
\repeat percent 2 { \times 2/3 { r8 c d } e4 }
```
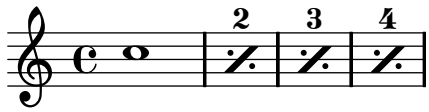


Ausgewählte Schnipsel

Prozent-Wiederholungen zählen

Ganztaktwiederholungen mit mehr als zwei Wiederholungen erhalten einen Zähler, wenn man die entsprechende Eigenschaft einsetzt:

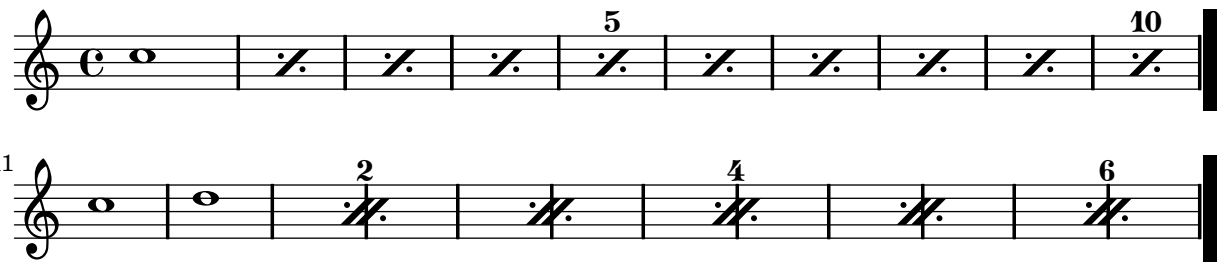
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



Sichtbarkeit von Prozent-Wiederholungen

Prozentwiederholungszähler können in regelmäßigen Intervallen angezeigt werden, indem man die Eigenschaft `repeatCountVisibility` beeinflusst.

```
\relative c'' {
  \set countPercentRepeats = ##t
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)
  \repeat percent 10 { c1 } \break
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
  \repeat percent 6 { c1 d1 }
}
```
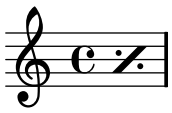


Isolierte Prozentwiederholungen

Isolierte Prozentwiederholungen können auch ausgegeben werden. Das wird erreicht, indem man eine Ganztaktpause notiert und ihre Ausgabeform ändert:

```
makePercent =
#(define-music-function (parser location note) (ly:music?)
   "Make a percent repeat the same length as NOTE."
   (make-music 'PercentEvent
               'length (ly:music-length note)))

\relative c'' {
  \makePercent s1
}
```



Siehe auch

Glossar: Abschnitt “percent repeat” in *Glossar*, Abschnitt “simile” in *Glossar*.

Schnipsel: Abschnitt “Repeats” in *Schnipsel*.
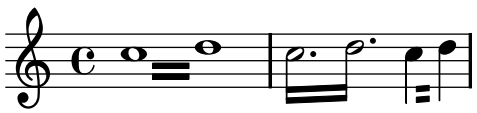
Referenz der Interna: Abschnitt “RepeatSlash” in *Referenz der Interna*, Abschnitt “RepeatSlashEvent” in *Referenz der Interna*, Abschnitt “DoubleRepeatSlash” in *Referenz der Interna*, Abschnitt “PercentRepeat” in *Referenz der Interna*, Abschnitt “PercentRepeatCounter” in *Referenz der Interna*, Abschnitt “PercentRepeatedMusic” in *Referenz der Interna*, Abschnitt “Percent_repeat_engraver” in *Referenz der Interna*, Abschnitt “DoublePercentEvent” in *Referenz der Interna*, Abschnitt “DoublePercentRepeat” in *Referenz der Interna*, Abschnitt “DoublePercentRepeatCounter” in *Referenz der Interna*, Abschnitt “Double_percent_repeat_engraver” in *Referenz der Interna*, Abschnitt “Slash_repeat_engraver” in *Referenz der Interna*.

Tremolo-Wiederholung

Tremolos können in zwei Arten notiert werden: als Wechsel zwischen zwei Noten oder Akkorden oder als schnelle Wiederholung einer einzigen Note. Tremolos, die als Wechsel realisiert werden, werden dargestellt, indem Balken zwischen die Noten gesetzt werden, Tremolos, die eine schnelle Wiederholung darstellen, haben Balken oder Schrägstriche am Hals einer einzigen Note.

Um Tremolobalken zwischen Noten zu setzen, kann der `\repeat`-Befehl mit dem Tremolo-Stil benutzt werden:

```
\repeat tremolo 8 { c16 d }
\repeat tremolo 6 { c16 d }
\repeat tremolo 2 { c16 d }
```
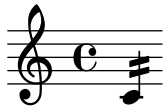


Die `\repeat tremolo`-Syntax braucht genau zwei Noten innerhalb der geschweiften Klammern, und die Anzahl der Wiederholungen muss einem Wert entsprechen, der mit einfachen oder punktierten Noten ausgedrückt werden kann. `\repeat tremolo 7` funktioniert und setzt Tremolo für die Dauer einer Doppeltpunktierten, aber `\repeat tremolo 9` funktioniert nicht.

Die Dauer des Tremolos entspricht der Dauer der Wertes in Klammern, multipliziert mit der Zahl der Wiederholungen: `\repeat tremolo 8 { c16 d16 }` ergibt ein Tremolo für eine Ganze, notiert als zwei Ganze, die zwei Tremolobalken zwischen sich haben.
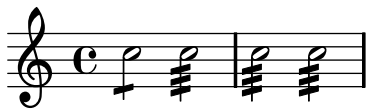
Es gibt zwei Möglichkeiten, ein Tremolozeichen zu einer einzelnen Noten hinzuzufügen. Die `\repeat tremolo`-Syntax kann hier auch benutzt werden; in diesem Fall wird die Note allerdings nicht eingeklammert:

```
\repeat tremolo 4 c'16
```



Die gleiche Darstellung wird erreicht, indem nach der Note `:Zahl` geschrieben wird. Die Zahl zeigt die Dauer der Unterteilung an, und sie muss mindestens den Wert 8 haben. Ein Wert von 8 ergibt einen Balken durch den Notenhals. Wenn die Zahl ausgelassen wird, wird der letzte benutzte Wert eingesetzt (gespeichert in `tremoloFlags`):

```
c2:8 c:32
c: c:
```

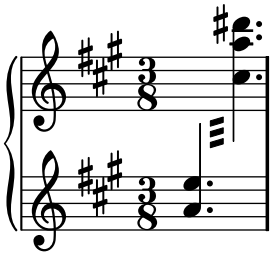


Ausgewählte Schnipsel

Cross-staff tremolos

Since `\repeat tremolo` expects exactly two musical arguments for chord tremolos, the note or chord which changes staff within a cross-staff tremolo should be placed inside curly braces together with its `\change Staff` command.
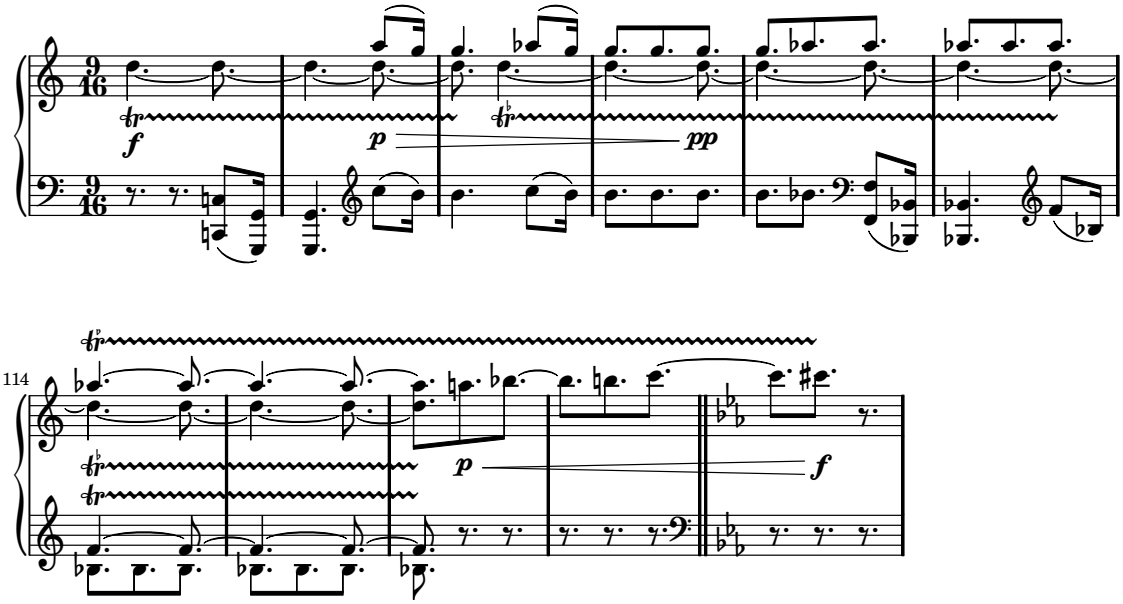
```
\new PianoStaff <<
  \new Staff = "up" \relative c'' {
    \key a \major
    \time 3/8
    s4.
  }
  \new Staff = "down" \relative c'' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
>>
```



Siehe auch

Schnipsel: Abschnitt “Repeats” in *Schnipsel*.

1.5 Gleichzeitig erscheinende Noten



Polyphonie bedeutet in der musikalischen Terminologie das Vorhandensein von mehr als einer (eigenständigen) Stimme in einem Stück. Für LilyPond bedeutet es aber das Vorhandensein von mehr als einer Stimme pro System.

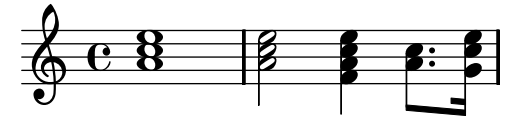
1.5.1 Eine einzelne Stimme

Dieser Abschnitt behandelt gleichzeitige Noten innerhalb derselben Stimme.

Noten mit Akkorden

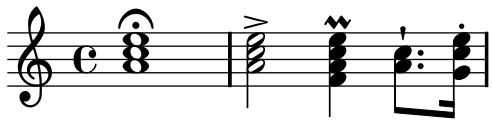
Ein Akkord wird notiert, indem die zu ihm gehörenden Tonhöhen zwischen spitze Klammern (< und >) gesetzt werden. Auf einen Akkord kann eine Dauer-Angabe folgen, genauso wie bei einfachen Noten.

```
<a c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
```



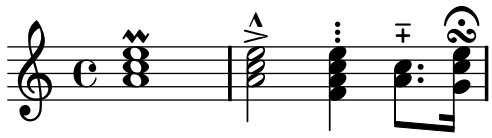
Akkorde können auch von Artikulationen gefolgt werden, genau wie auch einfache Noten.

```
<a c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^| <g c e>16-.
```



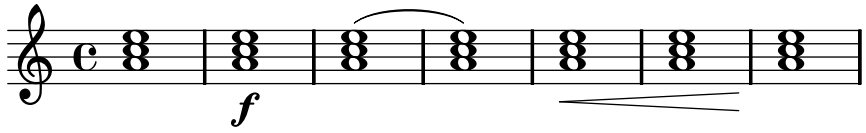
Die Noten innerhalb der Akkorde konnen auch von Artikulationen oder Ornamenten gefolgt werden.

```
<a c\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4 <a-+ c-->8. <g\fermata c e\turn>16
```



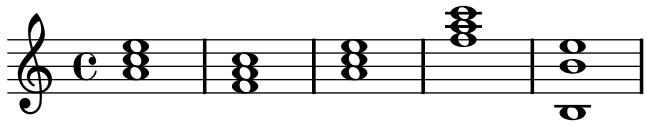
Manche Notationselemente, wie etwa Dynamik, Crescendo-Klammern und Legatobögen müssen an den gesamten Akkord gehängt werden und nicht an einzelne Noten, damit sie ausgegeben werden.

```
<a\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>) <a c e>\< <a c e> <a c e>\!
```



Der relative Modus kann auch für Tonhöhen in Akkorden eingesetzt werden. Die erste Note eines Akkordes ist immer relativ zur ersten Note des vorherigen Akkordes, oder mit der Tonhöhe der letzten Note vor dem Akkord (wenn kein Akkord vorhergeht). Alle anderen Noten innerhalb des Akkordes sind relativ zu der Note vorher innerhalb des selben Akkordes.

```
<a c e>1 <f a c> <a c e> <f' a c> <b, e b,>
```



Mehr Information über Akkorden findet sich in Abschnitt 2.7 [Notation von Akkorden], Seite 371.

Siehe auch

Musikglossar: Abschnitt “chord” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Noten zu Akkorden verbinden” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 2.7 [Notation von Akkorden], Seite 371, [Artikulationszeichen und Verzierungen], Seite 106, [Relative Oktavenbezeichnung], Seite 2, Abschnitt 1.5.2 [Mehrere Stimmen], Seite 146.

Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

Bekannte Probleme und Warnungen

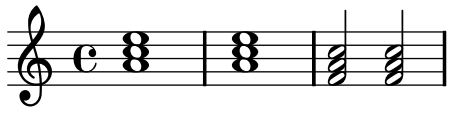
Akkorde, die mehr als zwei Tonhöhen für einen Notenlinenzwischenraum enthalten (wie etwa ‘`<e f! fis!>`’) produzieren überlappende Notenköpfe. Abhängig von der Situation kann eines der folgenden Dinge helfen, die Darstellung zu verbessern:

- Kurzzeitig mehrere Stimmen benutzen, siehe Abschnitt 1.5.2 [Mehrere Stimmen], Seite 146: ‘`<y f! \\ <e fis!> >>`’,
- enharmonische Transkription für einen oder mehrere Tonhöhen vornehmen: ‘`<e f ges>`’ oder
- Cluster, siehe [Cluster], Seite 145.

Akkord-Wiederholungen

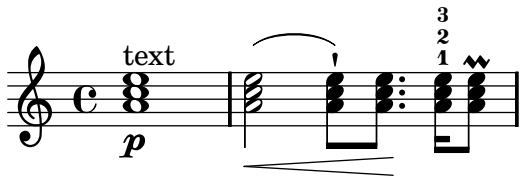
Um Schreibarbeit zu ersparen, kann ein Zeichen benutzt werden, um den vorhergehenden Akkord zu wiederholen. Das Symbol hierzu ist `q`:

```
<a c e>1 q <f a c>2 q
```



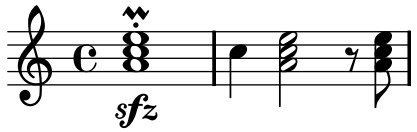
Genauso wie normale Akkorde kann auch das Akkord-Wiederholungssymbol in Verbindung mit Tondauern, Artikulationen, Beschriftungen, Legatobögen, Balken usw. benutzt werden, weil nur die Tonhöhen des vorangehenden Akkordes wiederholgt werden.

```
<a c e>1\p^"text" q2\<( q8)[-| q8.]\! q16-1-2-3 q8\prall
```



Das Akkord-Wiederholungssymbol behält keine Dynamikzeichen, Artikulationen oder Ornamente, die in oder an den vorhergehenden Akkord gehängt waren.

```
<a-. c\prall e>1\sfz c4 q2 r8 q8
```



Siehe auch

Notationsreferenz: Abschnitt 2.7 [Notation von Akkorden], Seite 371, [Artikulationszeichen und Verzierungen], Seite 106.
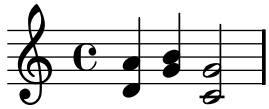
Installierte Dateien: ‘`ly/chord-repetition-init.ly`’.

Gleichzeitige Ausdrücke

Eine oder mehrere musikalische Ausdrücke, die in doppelte spitze Klammern eingeschlossen werden, werden gleichzeitig gesetzt. Wenn der erste Ausdruck mit einer einzelnen Note beginnt oder die gesamte Konstruktion explizit in einer einzelnen Stimme erstellt wird, wird auch nur ein Notensystem erstellt. In anderem Falle werden die Elemente der simultanen Konstruktion auf unterschiedlichen Systemen gesetzt.
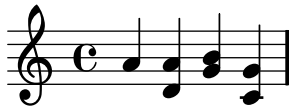
Das nächste Beispiel zeigt simultane Konstruktionen auf einem System:

```
\new Voice {  % explicit single voice
  << { a4 b g2 } { d4 g c,2 } >>
}
```



```
% single first note
```
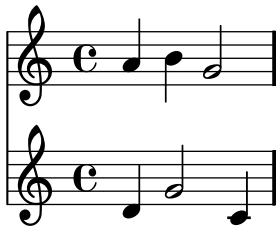
```
a << { a4 b g } { d4 g c, } >>
```



Dass kann benutzt werden, wenn die simultanen Abschnitte einen identischen Rhythmus haben, aber wenn versucht wird, Noten mit unterschiedlicher Dauer an denselben Hals zu setzen, gibt es Fehlermeldungen.

Das nächste Beispiel zeigt, wie ein simultaner Ausdruck implizit mehrere Systeme erstellt:

```
% no single first note
<< { a4 b g2 } { d4 g2 c,4 } >>
```

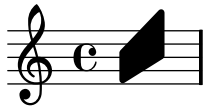


In diesem Fall stellt der unterschiedliche Rhythmus kein Problem dar.

Cluster

Ein Cluster zeigt an, dass alle Tonhöhen in einem Bereich gleichzeitig gespielt werden sollen. Cluster können gedeutet werden als eine Zusammenfassung einer ganzen Anzahl von Noten. Sie werden notiert, indem die Funktion `\makeClusters` auf eine Reihe von Akkorden angewendet wird:

```
\makeClusters { <g b>2 <c g'> }
```



Normale Noten und Cluster können zusammen im selben System notiert werden, sogar gleichzeitig. In solchen Fällen wird nicht versucht, automatisch Zusammenstöße zwischen normalen Noten und Clustern aufzulösen.

Siehe auch

Musikglossar: Abschnitt “cluster” in *Glossar*.

Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

Referenz der Interna: Abschnitt “ClusterSpanner” in *Referenz der Interna*, Abschnitt “ClusterSpannerBeacon” in *Referenz der Interna*, Abschnitt “Cluster_spanner_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Cluster sehen nur gut aus, wenn sie wenigstens über zwei Akkorde reichen – andernfalls sind sie zu schmal.

Cluster haben keine Hälse und können auch selber keine Dauern darstellen, aber die Länge des gesetzten Clusters wird erschlossen anhand der Dauern der definierten Akkorde. Voneinander getrennte Cluster brauchen eine unsichtbare Pause zwischen sich.

Cluster produzieren kein MIDI.

1.5.2 Mehrere Stimmen
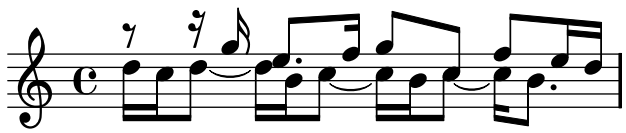
Dieser Abschnitt behandelt gleichzeitige Noten in mehreren Stimmen oder mehreren Systemen.

Mehrstimmigkeit in einem System

Stimmen explicit beginnen

Die grundlegende Struktur, die man benötigt, um mehrere unabhängige Stimmen in einem Notensystem zu setzen, ist im Beispiel unten dargestellt:

```
\new Staff <<
  \new Voice = "first"
    { \voiceOne r8 r16 g e8. f16 g8[ c,] f e16 d }
  \new Voice= "second"
    { \voiceTwo d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```



Stimmen werden hier explizit erstellt und erhalten Bezeichnungen zugewiesen. Die `\voiceOne` ... `\voiceFour`-Befehle stellen die Stimmen so ein, dass für die erste und dritte Stimme die Hälse nach oben zeigen, für die zweite und vierte Stimme hingegen nach unten. Die Noten der dritten und vierten Stimme werden horizontal verschoben, und Pausen in den entsprechenden Stimmen werden automatisch verschoben, um Zusammenstöße zu vermeiden. Der `\oneVoice`-Befehl stellt das Standardverhalten mit neutralen Halsrichtungen wieder her.

Vorübergehende polyphone Passagen
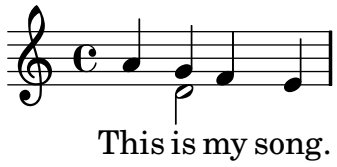
Ein vorübergehender polyphoner Abschnitt kann mit folgender Konstruktion erstellt werden:

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

Der erste Ausdruck innerhalb des polyphonen Abschnitts wird in den `Voice`-Kontext gestellt, der unmittelbar vor dem polyphonen Abschnitt aktiv war, und der gleiche `Voice`-Kontext setzt sich nach dem Abschnitt fort. Andere Ausdrücke innerhalb der eckigen Klammern werden anderen Stimmennummern zugewiesen. Damit lassen sich auch Gesangstexte einer durchgehenden Stimme vor, während und nach dem polyphonen Abschnitt zuweisen:

```
<<
  \new Voice = "melody" {
    a4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
```

```
  }
  \new Lyrics \lyricsto "melody" {
  This is my song.
  }
>>
```
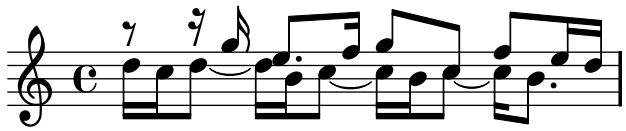


Hierbei sind die Befehle `\voiceOne` und `\voiceTwo` notwendig, um die Einstellungen für jede Stimme zu initialisieren.

Die Konstruktion mit doppeltem Backslash

Die `<< {...} \\ {...} >>`-Konstruktion, in welcher die beiden (oder mehreren) Ausdrücke durch doppelte Backslash-Zeichen (Taste AltGr+ß) getrennt werden, verhält sich anderes als die ähnliche Konstruktion ohne die doppelten Schrägstriche: *alle* Ausdrücke innerhalb der eckigen Klammern werden in diesem Fall jeweils neuen `Voice`-Kontexten zugeordnet. Diese neuen `Voice`-Kontexte werden implizit erstellt und haben die festen Bezeichnungen `"1"`, `"2"` usw.

Das erste Beispiel könnte also auch wie folgt notiert werden:
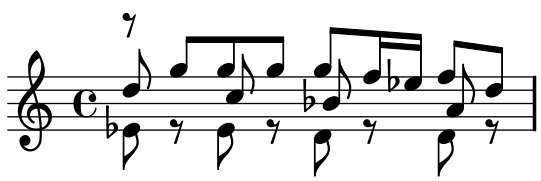
```
<<
  { r8 r16 g e8. f16 g8[ c,] f e16 d }
  \\
  { d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```



Diese Syntax kann benutzt werden, wenn es keine Rolle spielt, ob vorübergehend Stimmen erstellt werden und dann wieder verworfen werden. Diese implizit erstellten Stimmen erhalten die Einstellungen, die in den Befehlen `\voiceOne` ... `\voiceFour` enthalten sind, in der Reihenfolge, in der sie im Quelltext auftauchen.

Im nächsten Beispiel zeigen die Hälse der zeitweiligen Stimme nach oben, sie wird deshalb erst als dritte in der Konstruktion notiert, damit sie die Eigenschaften von `voiceThree` zugewiesen bekommt. Unsichtbare Pause werden eingesetzt, damit keine doppelten Pausen ausgegeben werden.

```
<<
  { r8 g g  g g f16 ees f8 d }
  \\
  { ees,8 r ees r d r d r }
  \\
  { d'8 s c s bes s a s }
>>
```



Es wird sehr empfohlen, in allen außer den allereinfachsten Stücken explizite Stimmenkontexte zu erstellen, wie erklärt inAbschnitt “Kontexte und Engraver” in *Handbuch zum Lernen* und Abschnitt “Stimmen explizit beginnen” in *Handbuch zum Lernen*.
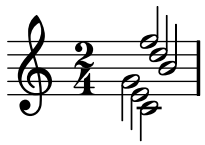
Stimmen-Anordnung

Wenn mehrere Stimmen notiert werden, sollte folgende Anordnung eingehalten werden:

```
Stimme 1: höchste
Stimme 2: tiefste
Stimme 3: zweithöchste
Stimme 4: zweittiefste
Stimme 5: dritthöchste
Stimme 6: dritttiefste
usw.
```

Auch wenn das erst nicht einleuchtend erscheint, erleichtert es den automatischen Layoutprozess doch sehr. Die ungeraden Stimmen erhalten Hälse nach oben, die graden Stimmen Hälse nach unten:
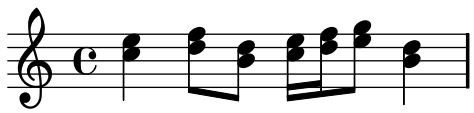
```
\new Staff <<
  \time 2/4
  { f''2 }  % 1: highest
  \\
  { c'2  }  % 2: lowest
  \\
  { d''2 }  % 3: second-highest
  \\
  { e'2  }  % 4: second-lowest
  \\
  { b'2  }  % 5: third-highest
  \\
  { g'2  }  % 6: third-lowest
>>
```



Identische Rhythmen

Wenn parallele Abschnitte gesetzt werden sollen, die identischen Rhythmus haben, kann man die Ausdrücke in einen einzigen `Voice`-Kontext parallel kombinieren, sodass sich Akkorde ergeben. Um das zu erreichen, müssen sie einfach von spitzen Klammern innerhalb einer expliziten Stimme umgeben werden:

```
\new Voice <<
  { e4 f8 d e16 f g8 d4 }
  { c4 d8 b c16 d e8 b4 }
>>
```



Mit dieser Methode können sich seltsame Balken und Warnungen ergeben, wenn die Musikausdrücke nicht den gleichen Rhythmus haben.

Vordefinierte Befehle

`\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`.

Siehe auch

Handbuch zum Lernen: Abschnitt “Voice enthält Noten” in *Handbuch zum Lernen*, Abschnitt “Stimmen explizit beginnen” in *Handbuch zum Lernen*.

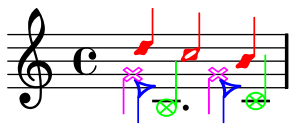
Notationsreferenz: [Schlagzeugsysteme], Seite 351, [Unsichtbare Pausen], Seite 53, [Hälse], Seite 196.

Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

Stimmenstile

Stimmen können unterschiedliche Farben erhalten, um einfach erkennbar zu sein:

```
<<
  { \voiceOneStyle d4 c2 b4 }
  \\
  { \voiceTwoStyle e,2 e }
  \\
  { \voiceThreeStyle b2. c4 }
  \\
  { \voiceFourStyle g'2 g }
>>
```



Der `\voiceNeutralStyle`-Befehl wird benutzt, um wieder die Standardausgabe einzuschalten.

Vordefinierte Befehle

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`, `\voiceNeutralStyle`.

Siehe auch

Handbuch zum Lernen: Abschnitt “Ich höre Stimmen” in *Handbuch zum Lernen*, Abschnitt “Mehr Information” in *Handbuch zum Lernen*.
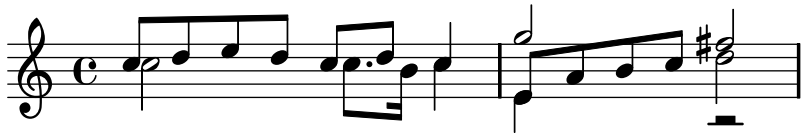
Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

Auflösung von Zusammenstößen

Die Notenköpfe von Noten in unterschiedlichen Stimmen mit derselben Tonhöhe, demselben Notenkopf und den Hälsen in entgegengesetzte Richtungen werden automatisch verschmolzen, aber Noten mit unterschiedlichen Köpfen oder den Hälsen in die selbe Richtung werden nicht verschmolzen. Pausen, die einem Hals in einer anderen Stimme gegenüberstehen, werden vertikal verschoben. Das folgende Beispiel zeigt drei unterschiedliche Situationen, auf Taktposition 1 und 3 in Takt 1 und Taktposition 1 in Takt 2, wo das automatische Verschmelzen nicht funktioniert.
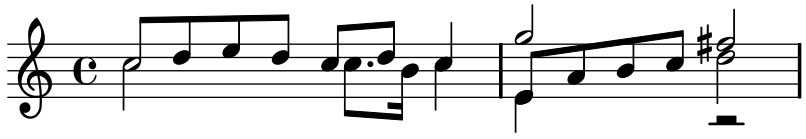
```
<<
  {
    c8 d e d c d c4
    g'2 fis
```

```
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



Noten mit unterschiedlichen Notenköpfen können verschmolzen werden, mit der Ausnahme von Halben- und Viertelnotenköpfen, wie im Beispiel unten gezeigt. Hier werden die Notenköpfe auf Taktposition 1 im ersten Takt verschmolzen:
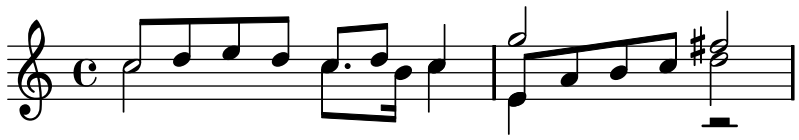
```
<<
  {
    \mergeDifferentlyHeadedOn
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



Auch Köpfe mit unterschiedlichen Punktierungen wie auf Taktposition 3 im ersten Takt können verschmolzen werden:
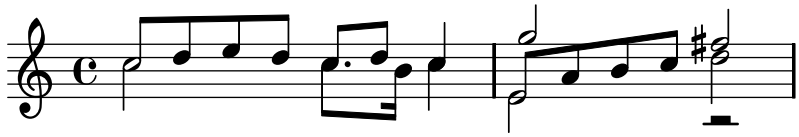
```
<<
  {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
```

```
    e8 a b c d2
  }
>>
```



Die Halbe und die Achtel am Anfang des zweiten Taktes werden fehlerhaft verschmolzen, weil die automatische Verschmelzung nicht richtig arbeiten kann, wenn drei oder mehr Noten zur gleichen Zeit auftreten – und in diesem Fall ist der verschmolzene Notenkopf nicht richtig. Um das Verschmelzen zuzulassen, muss ein `\shift` (Verschiebung) auf die Note angewendet werden, die nicht verschmolzen werden soll. In diesem Fall wurde `\shiftOn` gesetzt, um das oberste *g* aus der Kolumne zu entfernen. Jetzt funktioniert `\mergeDifferentlyHeadedOn` (verschmelze Noten mit unterschiedlichen Köpfen) so wie es soll.

```
<<
  {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c8 d e d c d c4
    \shiftOn
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```
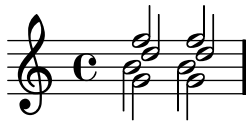


Der `shiftOn`-Befehl ermöglicht die Noten einer Stimme zu verschieben, erzwingt dieses Verhalten aber nicht. Wenn `shiftOn` auf eine Stimme angewendet wird, eine Note oder ein Akkord in der Stimme wird nur verschoben, wenn sein Hals mit dem Hals der Note einer anderen Stimme kollidieren würde, und nur, wenn der Hals der Kollisionsnote in die gleiche Richtung zeigt. Der `shiftOff`-Befehl verhindert, dass eine derartige Verschiebung stattfinden kann.

Die äußeren Stimmen (also normalerweise Stimmen 1 und 2) haben als Standard `shiftOff` eingestellt, während die inneren Stimmen (3 und mehr) `shiftOn` definiert haben. Wenn eine Verschiebung stattfindet, werden Stimmen mit den Hälsen nach oben (also ungerade Stimmen) nach rechts verschoben, während Stimmen mit den Hälsen nach unten (also gerade Stimmen) nach links verschoben werden.

Hier ein Beispiel, das verstehen hilft, wie ein verkürzter polyphonischer Abschnitt intern ausgeweitet wird.

Achtung: Wenn Sie drei oder mehr Stimmen haben, sollte die vertikale Anordnung der Stimmen in der Eingabedatei nicht die gleiche sein wie die vertikale Anordnung der Stimmen im Notensystem!

```
\new Staff \relative c'' {
  %% abbreviated entry
  <<
    { f2  }  % 1: highest
    \\
    { g,2 }  % 2: lowest
    \\
    { d'2 }  % 3: upper middle
    \\
    { b2  }  % 4: lower middle
  >>
  %% internal expansion of the above
  <<
    \new Voice = "1" { \voiceOne   \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo   \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn  d'2 } % shifts right
    \new Voice = "4" { \voiceFour  \shiftOn  b2  } % shifts left
  >>
}
```



Zwei zusätzliche Befehle, `shiftOnn` und `shiftOnnn` stellen weitere Verschiebungsebenen zu verfügung, die vorübergehend eingesetzt werden können um Zusammenstöße in komplizierten Situationen aufzulösen. Siehe auch Abschnitt "Beispiel aus dem Leben" in *Handbuch zum Lernen*.

Noten werden nur verschmolzen, wenn ihre Hälse in entgegengesetzte Richtungen zeigen (also etwa wie `Voice` 1 und 2 in den Standardeinstellungen oder wenn die Hälse explizit in unterschiedliche Richtungen gedreht sind).

Vordefinierte Befehle

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`, `\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`.

Ausgewählte Schnipsel
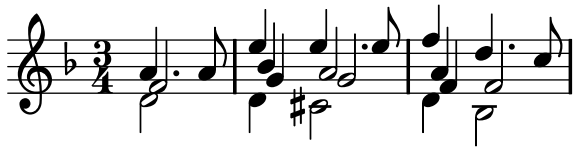
Zusätzliche Stimmen um Zusammenstöße zu vermeiden

Ein einigen Fällen von sehr komplexer polyphoner Musik sind zusätzliche Stimmen notwendig, um Zusammenstöße zwischen den Noten zu vermeiden. Wenn mehr als vier parallele Stimmen benötigt werden, können zusätzliche Stimmen definiert werden, indem eine Variable mit der Funktion `context-spec-music` definiert wird.

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)

\relative c'' {
  \time 3/4
  \key d \minor
  \partial 2
```

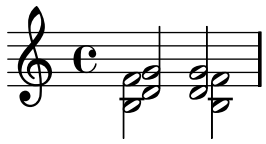
```
  <<
    {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \\
    {
      \voiceThree
      f,2
      bes4 a2
      a4 s2
    }
    \\
    {
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
    \\
    \bar "||"{
      \voiceTwo
      d2
      d4 cis2
      d4 bes2
    }
  >>
}
```



Horizontale Verschiebung von Noten erzwingen

Wenn es zu Zusammenstößen kommt, kann mit folgender Lösung eine andere Position manuell eingestellt werden. Die Einheiten hier sind Notenlinienzwischenräume.

```
\relative c' <<
  {
    <d g>2 <d g>
  }
  \\
  {
    <b f'>2
    \once \override NoteColumn #'force-hshift = #1.7
    <b f'>2
  }
>>
```



Siehe auch

Musikglossar: Abschnitt “polyphony” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Mehrere Noten auf einmal” in *Handbuch zum Lernen*, Abschnitt “Voice enthält Noten” in *Handbuch zum Lernen*, Abschnitt “Beispiele aus dem Leben” in *Handbuch zum Lernen*.

Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

Referenz der Interna: Abschnitt “NoteColumn” in *Referenz der Interna*, Abschnitt “NoteCollision” in *Referenz der Interna*, Abschnitt “RestCollision” in *Referenz der Interna*.

Automatische Kombination von Stimmen

Automatische Kombination von Stimmen wird verwendet, um zwei Stimmen auf einem Notensystem zu setzen. Es wird vor allem in Orchesterpartituren eingesetzt. Wenn beide Stimmen für einige Noten identisch sind, wird nur eine dargestellt. An den Stellen, an denen die beiden Stimmen sich unterscheiden, werden sie als unterschiedliche Stimmen gesetzt, und die Richtung der Hälse wird automatisch bestimmt. Zusätzlich werden *solo* und *a due*-Stellen erkannt und bezeichnet.

Die Syntax zur Stimmenkombination lautet:

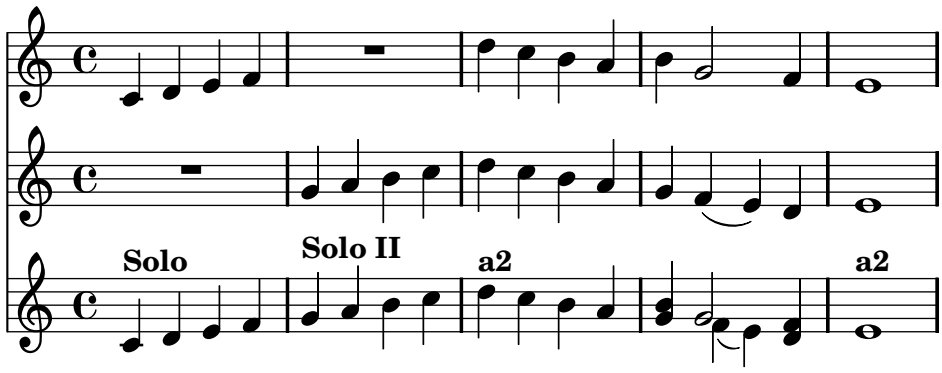
```
\partcombine musikAusdr1 musikAusdr2
```

Das nächste Beispiel zeigt, wie die Kombination funktioniert. Hier werden die Stimmen erst auf einem jeweils eigenen System und dann kombiniert gesetzt, beachten Sie, wie sich die Einstellungen für Polyphonie ändern.

```
instrumentOne = \relative c' {
  c4 d e f
  R1
  d'4 c b a
  b4 g2 f4
  e1
}

instrumentTwo = \relative g' {
  R1
  g4 a b c
  d c b a
  g f( e) d
  e1
}

<<
  \new Staff \instrumentOne
  \new Staff \instrumentTwo
  \new Staff \partcombine \instrumentOne \instrumentTwo
>>
```



Die Noten des dritten Taktes werden nur einfach ausgegeben, obwohl sie ein beiden Stimmen definiert sind. Die Richtung von Hälsen und Bögen werden automatisch gewählt, abhängig davon ob es eine Solo-Stelle oder Unisono ist. In polyphonen Situationen erhält die erste Stimme immer Hälse nach oben, die zweite Stimme Hälse nach unten. An Solo-Stellen werden die Stimmen mit „Solo“ bzw. „Solo II“ bezeichnet. Die Unisono-Stellen (*a due*) werden mit dem Text „a2“ gekennzeichnet.

Beide Argumente von `\partcombine` werden als `Voice`-Kontexte interpretiert. Wenn relative Oktaven benutzt werden, muss `\relative` für beide Stimmen benutzt werden, also:

```
\partcombine
  \relative ... musikAusdr1
  \relative ... musikAusdr2
```

Ein `\relative`-Abschnitt, der sich außerhalb von`\partcombine` befindet, hat keinen Einfluss auf die Tonhöhen von *`musikAusdr1`* oder *`musikAusdr2`*.

In professionellen Partituren werden Stimmen oft für lange Strecken auseinander gehalten, auch wenn eine oder mehrere Noten tatsälich aufeinander treffen und einfach als Unisono gesetzt werden könnten. Noten zu Akkorden zusammenzufassen oder eine Stimme als Solo anzuzeigen ist darum nicht ideal, weil die `\partcombine`-Funktion jede Note separat betrachtet. Aus diesem Grund kann die `\partcombine`-Funktion mit folgenden Befehlen verändert werden:

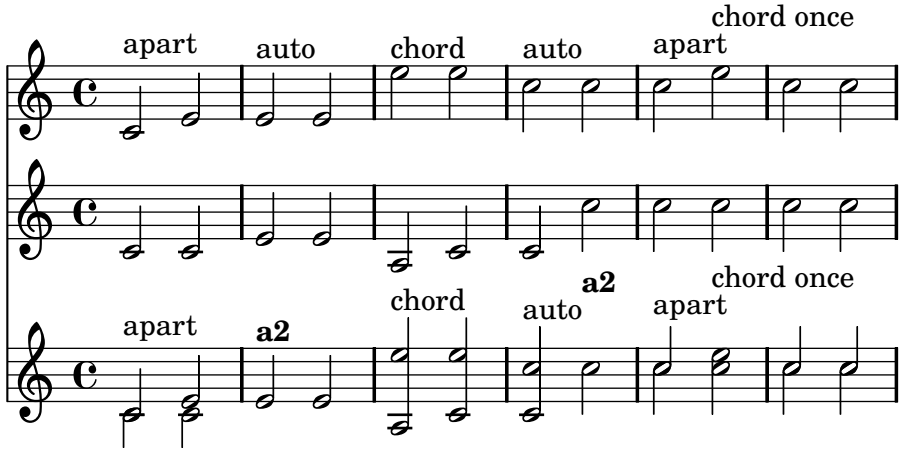
- `\partcombineApart`, `\partcombineApartOnce`: Behält die Noten als zwei unterschiedliche Stimmen, auch wenn sie als Akkord oder Unisono kombiniert werden könnten.
- `\partcombineChords`, `\partcombineChordsOnce`: Kombiniert die Noten als Akkord.
- `\partcombineUnisono`, `\partcombineUnisonoOnce`: Die zwei Stimmen sind Unisono.
- `\partcombineSoloI`, `\partcombineSoloIOnce`: Zeigt nur Stimme eins und markiert sie als Solo.
- `\partcombineSoloII`, `\partcombineSoloIIOnce`: Zeigt nur Stimme zwei und markiert sie als Solo.
- `\partcombineAutomatic`, `\partcombineAutomaticOnce`: Beendet die Wirkung der Sonderbefehle weiter oben und stellt das normale Verhalten des Kombinationsmechanismus wieder her.

Alle Befehle die auf `...Once` enden, gelten nur für die darauf folgende Note.

```
instrumentOne = \relative c' {
  \partcombineApart c2^"apart" e |
  \partcombineAutomatic e^"auto" e |
  \partcombineChords e'^"chord" e |
  \partcombineAutomatic c^"auto" c |
  \partcombineApart c^"apart" \partcombineChordsOnce e^"chord once" |
  c c |
}
instrumentTwo = \relative c' {
  c2 c |
```

```
  e2 e |
  a,2 c |
  c2 c' |
  c2 c |
  c2 c |
}

<<
  \new Staff { \instrumentOne }
  \new Staff { \instrumentTwo }
  \new Staff { \partcombine \instrumentOne \instrumentTwo }
>>
```



Ausgewählte Schnipsel

Zwei Stimmen auf einem System kombinieren
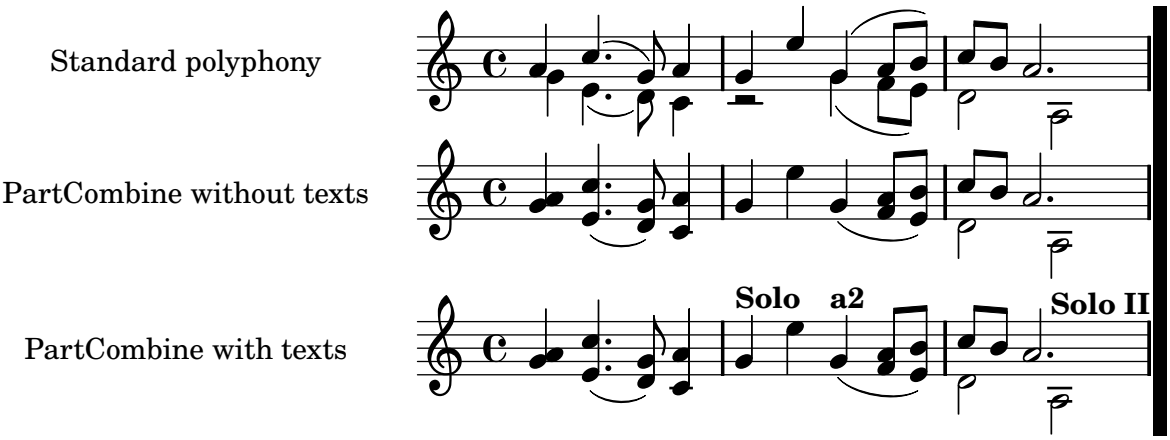
Die Funktion, die Stimmen kombiniert (also der `\partcombine`-Befehl) ermöglicht die Kombination unterschiedlicher Stimmen auf einem System. Textanweisungen wie "solo" oder "a2" werden automatisch hinzugefügt. Wenn man sie entfernen will, muss man die Eigenschaft `printPartCombineTexts` auf falsch setzen. Für Klavierauszüge muss natürlich kein "solo"/"a2" usw. hinzugefügt werden, man sollte sie also ausschalten. Wenn aber Solo-Stellen in einem Klavierauszug oder einer Chorpartitur angezeigt werden, ist es besser, normale Polyphonie zu verwenden, weil so die Solostellen angezeigt werden, auch wenn der Text des Stimmenkombinierers ausgeschaltet ist.

Der Schnipsel zeigt drei Möglichkeiten, Stimmen auf einem System zu kombinieren: Standardpolyphonie, `\partcombine` ohne Text und `\partcombine` mit Text.

```
musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
  c b a2.
}

musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}
```

```
\score {
  <<
    <<
    \new Staff {
      \set Staff.instrumentName = #"Standard polyphony"
      << \musicUp \\ \musicDown >>
    }
    \new Staff \with { printPartCombineTexts = ##f } {
      \set Staff.instrumentName = #"PartCombine without texts"
      \partcombine \musicUp \musicDown
    }
    \new Staff {
      \set Staff.instrumentName = #"PartCombine with texts"
      \partcombine \musicUp \musicDown
    }
    >>
  >>
  \layout {
    indent = 6.0\cm
    \context {
      \Score
      \override SystemStartBar #'collapse-height = #30
    }
  }
}
```
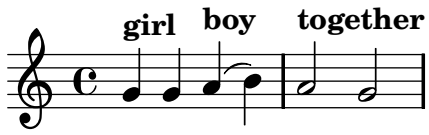


Partcombine-Text ändern

Wenn Stimmen automatisch kombiniert werden, kann der Text, der für Solo- und Unisono-Stellen ausgegeben wird, geändert werden:

```
\new Staff <<
  \set Staff.soloText = #"girl"
  \set Staff.soloIIText = #"boy"
  \set Staff.aDueText = #"together"
  \partcombine
    \relative c'' {
      g4 g r r
```

```
      a2 g
    }
    \relative c'' {
      r4 r a( b)
      a2 g
    }
>>
```



Siehe auch

Musikglossar: Abschnitt “a due” in *Glossar*, Abschnitt “part” in *Glossar*.

Notationsreferenz: Abschnitt 1.6.3 [Orchesterstimmen erstellen], Seite 178.

Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

Referenz der Interna: Abschnitt “PartCombineMusic” in *Referenz der Interna*, Abschnitt “Voice” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

`\partcombine` kann nur zwei Stimmen bearbeiten.

Wenn `printPartCombineTexts` (drucke Stimmenkombinationstext) gesetzt ist und die Stimmen die gleichen Noten wiederholt spielen, kann `a2` in einem Takt mehrmals gesetzt werden.

`\partcombine` kann nicht innerhalb von `\times` benutzt werden.

`\partcombine` kann nicht innerhalb von `\relative` benutzt werden.

Intern werden beide Argumente von `\partcombine` als Stimmen (`Voice`) interpretiert und entschieden, wann die Stimmen kombiniert werden können. Wenn sie unterschiedliche Dauern haben, können sie nicht kombiniert werden und erhalten die Bezeichnung `one` und `two` Darum werden Wechsel zu einem `Voice`-Kontext, der eine andere Bezeichnung hat, ignoriert. Genausowenig ist die Stimmenkombination dazu ausgelegt, Gesangstext zu verarbeiten: wenn eine der Stimmen eine explizite Bezeichnung erhält, damit Text damit verknüpft werden kann, hört die Stimmenkombination auf zu arbeiten.

`\partcombine` findet nur den Beginn von Noten. Es kann nicht bestimmen, ob eine vorher begonnene Noten weiterklingt, was zu verschiedenen Problemen führen kann.

Musik parallel notieren

Noten für mehrere Stimmen können verschachtelt notiert werden. Die Funktion `\parallelMusic` akzeptiert eine Liste mit den Bezeichnungen einer Reihe von Variablen und einen musikalischen Ausdruck. Der Inhalt der verschiedenen Takte in dem musikalischen Ausdruck bekommt die Bezeichnung der Variablen zugewiesen, sodass sie benutzt werden können, um die Musik dann zu setzen. Dabei entspricht jede Zeile einer Stimme.

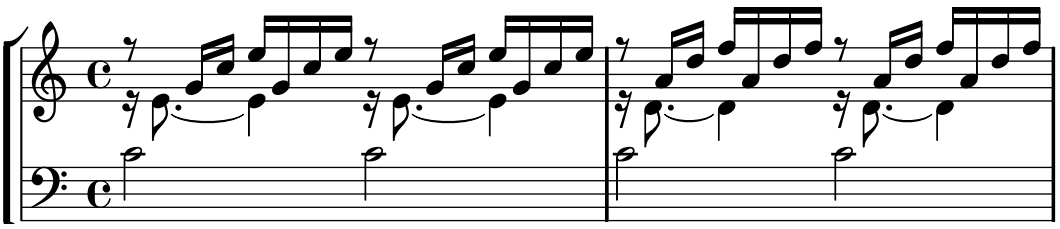
Achtung: Taktüberprüfungen `|` müssen benutzt werden, und die Takte müssen die gleiche Länge haben.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Bar 1
```

```
  r8 g'16 c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~   e'4           r16 e'8.~   e'4           |
  c'2                       c'2                       |

  % Bar 2
  r8 a'16 d'' f'' a' d'' f'' r8 a'16 d'' f'' a' d'' f'' |
  r16 d'8.~   d'4           r16 d'8.~   d'4           |
  c'2                       c'2                       |

}
\new StaffGroup <<
  \new Staff << \voiceA \\ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
```
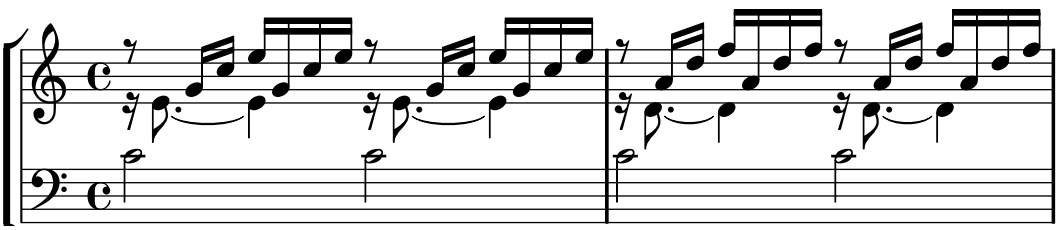


Der relative Modus kann auch benutzt werden. Beachten Sie, dass der `\relative`-Befehl nicht innerhalb von `\parallelMusic` benutzt wird. Die Noten sind parallel zu der vorherigen Note der gleichen Stimme , nicht zu der vorherigen Note in der Quelldatei. Anders gesagt ignorieren relative Noten von `voiceA` die Noten von `voiceB`.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Bar 1
  r8 g16 c e g, c e r8 g,16 c e g, c e  |
  r16 e8.~ e4       r16 e8.~  e4        |
  c2                c                   |

  % Bar 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~  d4       r16 d8.~  d4       |
  c2                 c                  |

 }
\new StaffGroup <<
  \new Staff << \relative c'' \voiceA \\ \relative c' \voiceB >>
  \new Staff \relative c' { \clef bass \voiceC }
>>
```



Das funktioniert ziemlich gut für Klaviernoten. Dieses Beispiel speichert vier konsekutive Takte in vier Variablen:
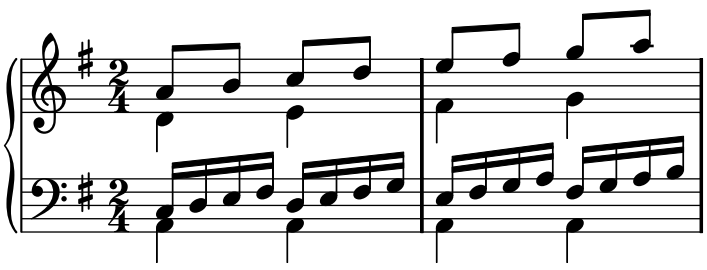
```
global = {
  \key g \major
  \time 2/4
}

\parallelMusic #'(voiceA voiceB voiceC voiceD) {
  % Bar 1
  a8    b     c   d     |
  d4          e         |
  c16 d e fis d e fis g |
  a4          a         |

  % Bar 2
  e8      fis  g     a   |
  fis4         g         |
  e16 fis g  a fis g a b |
  a4           a         |

  % Bar 3 ...
}

\score {
  \new PianoStaff <<
     \new Staff {
       \global
       <<
         \relative c'' \voiceA
         \\
         \relative c'  \voiceB
       >>
     }
     \new Staff {
       \global \clef bass
       <<
         \relative c \voiceC
         \\
         \relative c \voiceD
       >>
     }
  >>
}
```
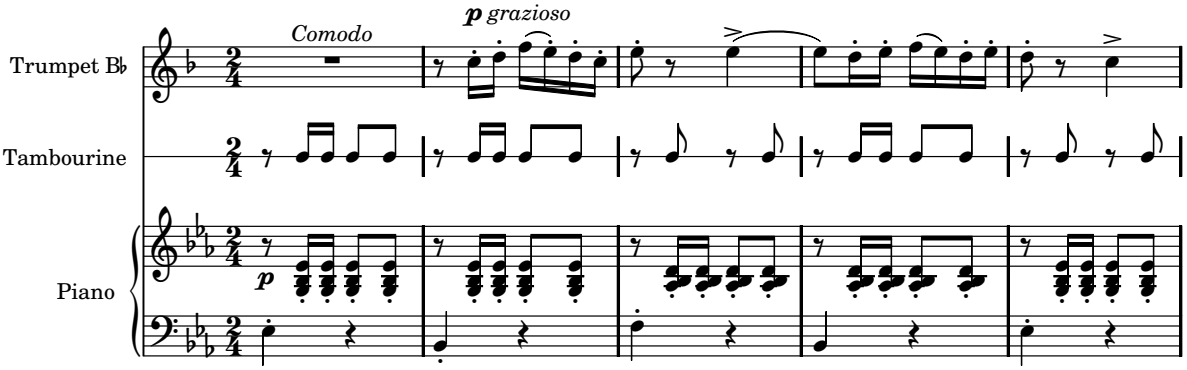


Siehe auch

Handbuch zum Lernen: Abschnitt “Stücke durch Bezeichner organisieren” in *Handbuch zum Lernen*.

Schnipsel: Abschnitt “Simultaneous notes” in *Schnipsel*.

1.6 Notation auf Systemen



Dieser Abschnitt zeigt, wie die Erscheinung von Systemen beeinflusst wird, wie Partituren mit mehr als einem System gesetzt werden und wie man Aufführungsanweisungen und Stichnoten zu einzelnen Systemen hinzufügt.

1.6.1 Systeme anzeigen lassen
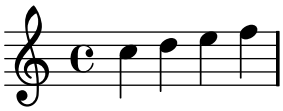
Dieser Abschnitt zeigt unterschiedliche Methoden, Notensysteme und Gruppen von Systemen zu erstellen.

Neue Notensysteme erstellen

Notensysteme (engl. *staff*, Pl. *staves*) werden mit dem `\new` oder `\context`-Befehl erstellt. Zu Einzelheiten siehe Abschnitt 5.1.2 [Kontexte erstellen], Seite 515.

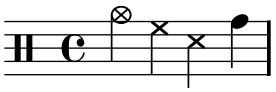
Der einfachste Notensystem-Kontext ist `Staff`:

```
\new Staff { c4 d e f }
```



`DrumStaff` (Perkussionsnotensystem) erstellt ein Notensystem mit fünf Linien, das für ein typisches Schlagzeug eingerichtet ist. Für jedes Instrument werden unterschiedliche Symbole dargestellt. Die Instrumente werden innerhalb der `drummode`-Umgebung gesetzt, wo jedes Instrument seine eigene Bezeichnung hat. Zu Einzelheiten siehe [Schlagzeugsysteme], Seite 351.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



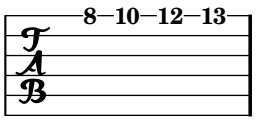
`RhythmicStaff` (Rhythmus-System) erstellt ein Notensystem mit nur einer Notenlinie, auf welcher nur die rhythmischen Werte der eingegebenen Noten dargestellt werden. Die wirklichen Längen bleiben erhalten. Zu Einzelheiten, siehe [Melodierhythmus anzeigen], Seite 72.

```
\new RhythmicStaff { c4 d e f }
```



`TabStaff` (Tabulatursystem) erstellt eine Tabulatur mit sechs Saiten in der üblichen Gitarrenstimmung. Zu Einzelheiten siehe [Standardtabulaturen], Seite 305.
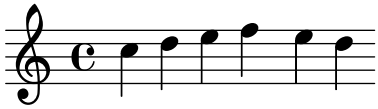
```
\new TabStaff { c4 d e f }
```



Es gibt zwei Notensysteme, die zur Notation von Alter Musik eingesetzt werden: `MensuralStaff` and `VaticanaStaff`. Sie sind erklärt in [Vordefinierte Umgebungen], Seite 393.

Das `GregorianTranscriptionStaff` (System zur Transkription des Gregorianischen Chorals) erstellt ein Notensystem, um modernen Gregorianischen Choral zu notieren. Es hat keine Notenlinien.

```
\new GregorianTranscriptionStaff { c4 d e f e d }
```



Neue Notensystem-Kontexte können selber definiert werden. Zu Einzelheiten, siehe Abschnitt 5.1.6 [Neue Kontexte definieren], Seite 522.

Siehe auch

Glossar: Abschnitt “staff” in *Glossar*, Abschnitt “staves” in *Glossar*.

Notationsreferenz: Abschnitt 5.1.2 [Kontexte erstellen], Seite 515, [Schlagzeugsysteme], Seite 351, [Melodierhythmus anzeigen], Seite 72, [Standardtabulaturen], Seite 305, [Vordefinierte Umgebungen], Seite 393, [Das Notensystem], Seite 169, [Gregorianische Gesangs-Kontexte], Seite 403, [Mensural-Kontexte], Seite 395, Abschnitt 5.1.6 [Neue Kontexte definieren], Seite 522.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

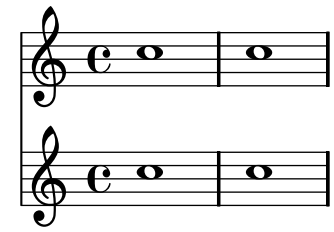
Referenz der Interna: Abschnitt “Staff” in *Referenz der Interna*, Abschnitt “DrumStaff” in *Referenz der Interna*, Abschnitt “GregorianTranscriptionStaff” in *Referenz der Interna*, Abschnitt “RhythmicStaff” in *Referenz der Interna*, Abschnitt “TabStaff” in *Referenz der Interna*, Abschnitt “MensuralStaff” in *Referenz der Interna*, Abschnitt “VaticanaStaff” in *Referenz der Interna*, Abschnitt “StaffSymbol” in *Referenz der Interna*.

Systeme gruppieren

Es gibt verschiedene Kontexte, um einzelne Notensysteme zu gruppieren und einer Partitur zu verbinden. Jeder Gruppenstil beeinflusst das Aussehen des Systemanfangs und das Verhalten der Taktlinien.

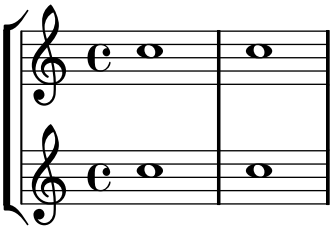
Wenn kein Kontext angegeben ist, wird die Standardeinstellung eingesetzt: die Gruppe beginnt mit einer vertikalen Linie und die Taktlinien sind nicht verbunden.

```
<<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



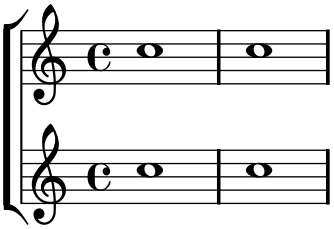
Im `StaffGroup`-Kontext die Gruppe mit einer eckigen Klammer begonnen und die Taktlinien durch alle Systeme gezogen.

```
\new StaffGroup <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



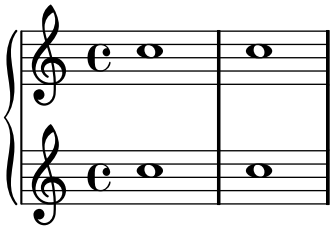
In einem `ChoirStaff` (Chorsystem) beginnt die Gruppe mit einer eckigen Klammer, aber die Taktlinien sind nicht verbunden.

```
\new ChoirStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



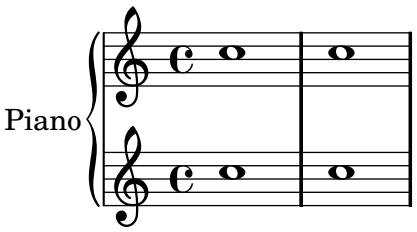
In einem `GrandStaff` (Akkolade) beginnt die Gruppe mit einer geschweiften Klammer und die Taktlinien sind durchgezogen.

```
\new GrandStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



Der `PianoStaff`-(Klaviersystem)-Kontext ist identisch mit dem `GrandStaff`-Kontext, aber es ermöglicht zusätzlich direkt die Angabe einer Instrumentbezeichnung. Zu Einzelheiten siehe [Instrumentenbezeichnungen], Seite 178.

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano"
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



Jede Systemgruppe stellt die Eigenschaft `systemStartDelimiter` (SystemBeginnBegrenzer) auf einen der folgenden Werte: `SystemStartBar`, `SystemStartBrace` oder `SystemStartBracket`. Ein vierter Begrenzer, `SystemStartSquare`, ist auch erreichbar, aber man muss ihr explizit einstellen.

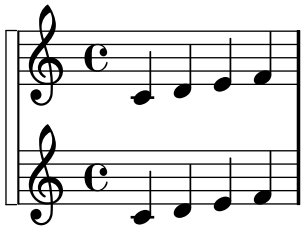
Neue Systemgruppen können definiert werden. Zu Einzelheiten siehe Abschnitt 5.1.6 [Neue Kontexte definieren], Seite 522.

Ausgewählte Schnipsel

Eine eckige Klammer zu Beginn von Systemgruppen benutzen

Die Klammer zu Beginn von Systemgruppen kann auch in eine eckige Klammer (`SystemStartSquare`) umgewandelt werden, wenn man sie explizit im `StaffGroup`- oder `ChoirStaffGroup`-Kontext setzt.

```
\score {
  \new StaffGroup { <<
  \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```



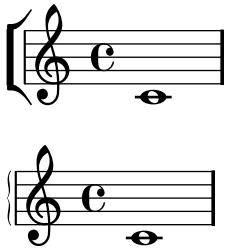
Klammer anzeigen wenn nur ein System gesetzt wird

Wenn nur ein System einer Systemgruppe vom Typ `ChoirStaff` oder `StaffGroup` angezeigt wird, wird die Klammer zu Beginn normalerweise nicht gesetzt. Das kann verändert werden, indem man die entsprechende Eigenschaft verändert.

Bei Systemen wie `PianoStaff` und `GrandStaff`, die mit einer geschweiften Klammer beginne, muss eine andere Eigenschaft verändert werden, wie das zweite Beispiel zeigt.

```
\markup \left-column {
  \score {
    \new StaffGroup <<
```

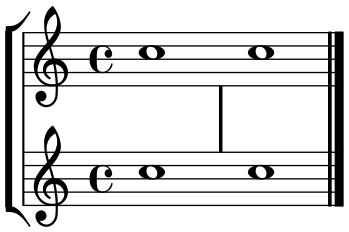
```
      % Must be lower than the actual number of staff lines
      \override StaffGroup.SystemStartBracket #'collapse-height = #1
      \override Score.SystemStartBar #'collapse-height = #1
      \new Staff {
        c'1
      }
    >>
    \layout { }
  }
  \null
  \score {
    \new PianoStaff <<
      \override PianoStaff.SystemStartBrace #'collapse-height = #1
      \override Score.SystemStartBar #'collapse-height = #1
      \new Staff {
        c'1
      }
    >>
    \layout { }
  }
}
```



Mensurstriche-Layout (Taktstriche zwischen den Systemen

Das Mensurstiche-Layout, in welchem die Taktlinien nicht auf den Systemen, sondern zwischen den Systemen gesetzt werden, kann mit einer `StaffGroup` anstelle von `ChoirStaff` erreicht werden. Die Taktlinien auf den Systemen werden mit der `transparent`-Eigenschaft ausgelöscht.

```
global = {
  \override Staff.BarLine #'transparent = ##t
  s1 s
  % the final bar line is not interrupted
  \revert Staff.BarLine #'transparent
  \bar "|."
}
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Siehe auch

Glossar: Abschnitt “brace” in *Glossar*, Abschnitt “bracket” in *Glossar*, Abschnitt “grand staff” in *Glossar*.

Notationsreferenz: [Instrumentenbezeichnungen], Seite 178, Abschnitt 5.1.6 [Neue Kontexte definieren], Seite 522.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “Staff” in *Referenz der Interna*, Abschnitt “StaffGroup” in *Referenz der Interna*, Abschnitt “ChoirStaff” in *Referenz der Interna*, Abschnitt “GrandStaff” in *Referenz der Interna*, Abschnitt “PianoStaff” in *Referenz der Interna*, Abschnitt “SystemStartBar” in *Referenz der Interna*, Abschnitt “SystemStartBrace” in *Referenz der Interna*, Abschnitt “SystemStartBracket” in *Referenz der Interna*, Abschnitt “SystemStartSquare” in *Referenz der Interna*.
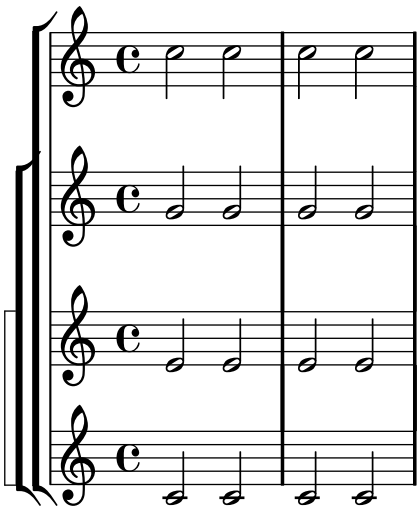
Bekannte Probleme und Warnungen

`PianoStaff` nimmt standardmäßig keine `ChordNames` (Akkordbezeichnungen) auf.

Verschachtelte Notensysteme

System-Gruppen können in beliebiger Tiefe geschachtelt werden. In diesem Fall erstellt jeder neue, innen liegende Kontext eine neue Klammer außerhalb der Klammer der Systemgruppe, in der er sich befindet.

```
\new StaffGroup <<
  \new Staff { c2 c | c2 c }
  \new StaffGroup <<
    \new Staff { g2 g | g2 g }
    \new StaffGroup \with {
      systemStartDelimiter = #'SystemStartSquare
    }
    <<
      \new Staff { e2 e | e2 e }
      \new Staff { c2 c | c2 c }
    >>
  >>
>>
```



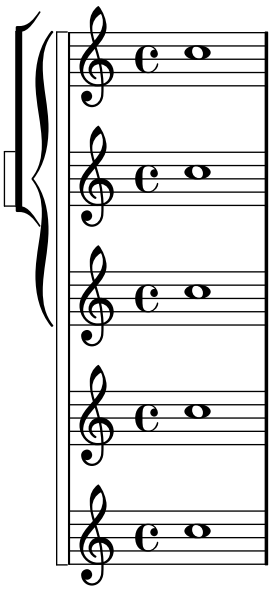
Neue geschachtelte Systemgruppen können definiert werden. Zu Einzelheiten siehe Abschnitt 5.1.6 [Neue Kontexte definieren], Seite 522.

Ausgewählte Schnipsel

Systeme schachteln

Die Eigenschaft `systemStartDelimiterHierarchy` kann eingesetzt werden, um komplizierte geschachtelte Systemklammern zu erstellen. Der Befehl `\set StaffGroup.systemStartDelimiterHierarchy` nimmt eine Liste mit der Anzahl der Systeme, die ausgegeben werden, auf. Vor jedem System kann eine Systemanfangsklammer angegeben werden. Sie muss in Klammern eingefügt werden und umfasst so viele Systeme, wie die Klammer einschließt. Elemente in der Liste können ausgelassen werden, aber die erste Klammer umfasst immer die gesamte Gruppe. Die Möglichkeiten der Anfangsklammer sind: `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace` und `SystemStartSquare`.

```
\new StaffGroup
\relative c'' <<
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
                              (SystemStartSquare b)  ) c ) d)
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```



Siehe auch

Notationsreferenz: [Systeme gruppieren], Seite 162, [Instrumentenbezeichnungen], Seite 178, Abschnitt 5.1.6 [Neue Kontexte definieren], Seite 522.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

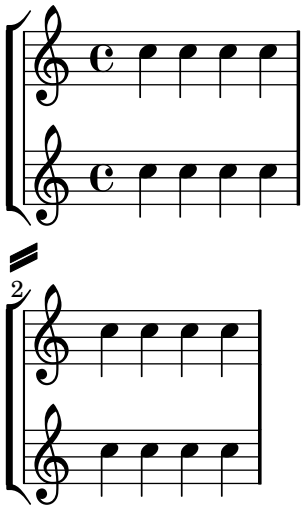
Referenz der Interna: Abschnitt “StaffGroup” in *Referenz der Interna*, Abschnitt “ChoirStaff” in *Referenz der Interna*, Abschnitt “SystemStartBar” in *Referenz der Interna*, Abschnitt “SystemStartBrace” in *Referenz der Interna*, Abschnitt “SystemStartBracket” in *Referenz der Interna*, Abschnitt “SystemStartSquare” in *Referenz der Interna*.

Systeme trennen

Wenn die Anzahl der Systeme sich von Seite zu Seite ändert, wird normalerweise ein Trennzeichen hinzugefügt, dass die Systeme voneinander trennt. Die Standardeinstellung ist, dass der Trenner nicht gesetzt wird, aber man kann ihn mit einer Option in der `\paper`-Umgebung angeschalten.

```
\book {
  \score {
    \new StaffGroup <<
      \new Staff {
        \relative c'' {
          c4 c c c
          \break
          c4 c c c
        }
      }
      \new Staff {
        \relative c'' {
          c4 c c c
          \break
          c4 c c c
        }
      }
    >>
  }
  \paper {
    system-separator-markup = \slashSeparator
```

```
    % following commands are needed only to format this documentation
    paper-width = 100\mm
    paper-height = 100\mm
    tagline = ##f
  }
}
```



Siehe auch

Notationsreferenz: Abschnitt 4.1 [Seitenlayout], Seite 461.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

1.6.2 Einzelne Systeme verändern
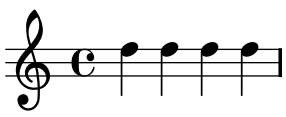
Dieser Abschnitt zeigt, wie man bestimmte Eigenschaften eines Systems ändert – etwa die Anzahl der Notenlinien oder die Größe des Systems. Es werden auch Methoden dargestellt, ein System zu beginnen und zu beenden sowie eine Methode, Ossia-Systeme zu erstellen.

Das Notensystem

Die Linien eines Notensystems gehören zu dem `StaffSymbol`-(NotensystemSymbol)-Grob. `StaffSymbol`-Eigenschaften können verändert werden, um die Erscheinung des Notensystems zu beeinflussen, aber sie müssen gesetzt werden, bevor das System erstellt wird.
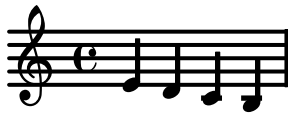
Die Anzahl der Notenlinien kann verändert werden. Die Position des Notenschlüssels und die Position von c’ können geändert werden, um dem neuen System zu entsprechen. Eine Erklärung findet sich im Schnipselabschnit in [Notenschlüssel], Seite 16.

```
\new Staff \with {
  \override StaffSymbol #'line-count = #3
}
{ d4 d d d }
```
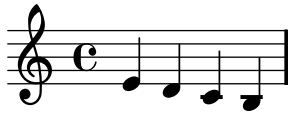


Die Liniendicke der Notenlinien kann verändert werden. Die Dicke der Hilfslinien und Notenhälse wird auch beeinflusst, weil sie von der Notenliniendicke abhängen.

```
\new Staff \with {
  \override StaffSymbol #'thickness = #3
}
{ e4 d c b }
```



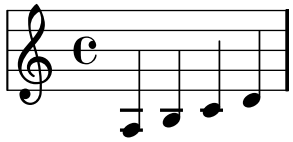
Die Dicke der Hilfslinien kann auch unabhängig von der Notenliniendicke verändert werden. Die zwei Zahlen in dem Beispiel sind Faktoren, mit denen die Notenlinien-Dicke und der Notenlinienabstand multipliziert werden. Die Addition beider Werte ergibt die Dicke der Hilfslinien.

```
\new Staff \with {
  \override StaffSymbol #'ledger-line-thickness = #'(1 . 0.2)
}
{ e4 d c b }
```



Der Abstand zwischen Notenlinien kann verändert werden. Diese Einstellung wirkt sich auch auf den Abstand der Hilfslinien aus.

```
\new Staff \with {
  \override StaffSymbol #'staff-space = #1.5
}
{ a4 b c d }
```



Weitere Einzelheiten zu den Eigenschaften von `StaffSymbol` findet sich in Abschnitt “staff-symbol-interface” in *Referenz der Interna*.

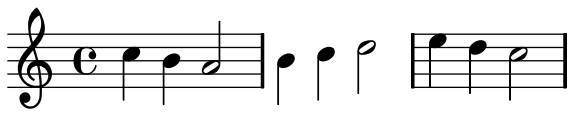
Veränderungen der Eigenschaften eines Notensystems mitten in einer Partitur können zwischen die Befehle `\stopStaff` und `\startStaff` gesetzt werden:

```
c2 c
\stopStaff
\override Staff.StaffSymbol #'line-count = #2
\startStaff
b2 b
\stopStaff
\revert Staff.StaffSymbol #'line-count
\startStaff
a2 a
```



Die Befehle `\startStaff` und `\stopStaff` können benutzt werden, um ein Notensystem irgendwo zu beenden oder zu beginnen.

```
c4 b a2
\stopStaff
b4 c d2
\startStaff
e4 d c2
```



Vordefinierte Befehle
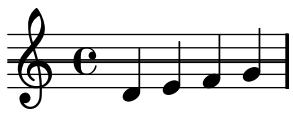
`\startStaff`, `\stopStaff`.

Ausgewählte Schnipsel

Eine Linie des Notensystems dicker als die anderen machen

Für den pädagogischen Einsatz kann eine Linie des Notensystems dicker gezeichnet werden (z. B. die Mittellinie, oder um den Schlüssel hervorzuheben). Das ist möglich, indem man zusätzliche Linien sehr nahe an der Linie, die dicker erscheinen soll, einfügt. Dazu wird die `line-positions`-Eigenschaft herangezogen.

```
{
  \override Staff.StaffSymbol #'line-positions = #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



Siehe auch

Glossar: Abschnitt “line” in *Glossar*, Abschnitt “ledger line” in *Glossar*, Abschnitt “staff” in *Glossar*.

Notationsreferenz: [Notenschlüssel], Seite 16.
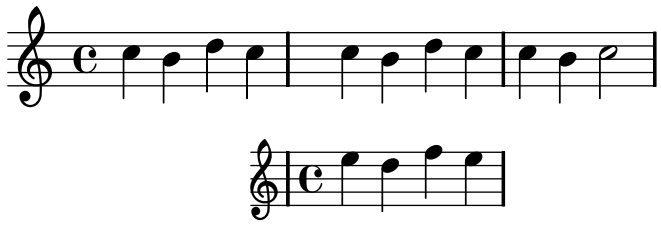
Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “StaffSymbol” in *Referenz der Interna*, Abschnitt “staff-symbol-interface” in *Referenz der Interna*.

Ossia-Systeme

Ossia-Systeme können gesetzt werden, indem zwei gleichzeitige Notensysteme an der entsprechenden Position erstellt werden:

```
\new Staff \relative c'' {
  c4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
  >>
  c4 b c2
}
```

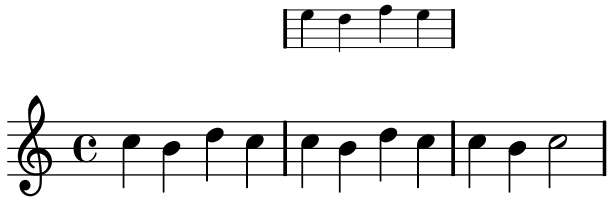


Dieses Beispiel ist aber normalerweise nicht erwünscht. Um Ossia-Systeme zu setzen, die sich über dem eigentlichen System befinden, keine Takt- und Schlüsselangaben haben und kleiner gesetzt sind, müssen einige Optimierungen angewendet werden. Im Handbuch zum Lernen wird eine Technik vorgestellt, mit der das gewünschte Ergebnis erreicht werden kann, beginnend in Abschnitt "Musikalische Ausdrücke ineinander verschachteln" in *Handbuch zum Lernen*.

Das Beispiel unten setzt die `alignAboveContext`-(oberhalbAusrichtenKontext)-Eigenschaft ein, um den Ossia-Abschnitt auszurichten. Diese Methode bietet sich an, wenn nur einige Ossia-Systeme benötigt werden.

```
\new Staff = main \relative c'' {
  c4 b d c
  <<
    { c4 b d c }
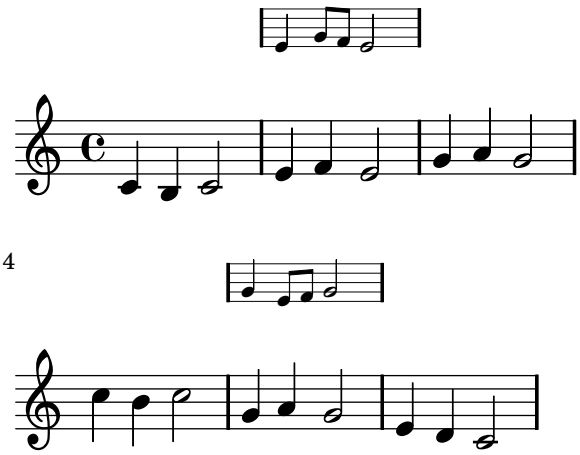
    \new Staff \with {
      \remove "Time_signature_engraver"
      alignAboveContext = #"main"
      fontSize = #-3
      \override StaffSymbol #'staff-space = #(magstep -3)
      \override StaffSymbol #'thickness = #(magstep -3)
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```



Wenn mehrere isolierte Ossia-Systeme gebraucht werden, kann es günstiger sein, einen leeren `Staff`-Kontext mit einer spezifischen *Kontextidentifikation* zu erstellen. Die Ossia-Abschnitte werden dann erstellt, indem dieser Kontext *aufgerufen* wird und mit `\startStaff` und `\stopStaff` an den richtigen Stellen sichtbar gemacht wird. Der Vorteil dieser Methode zeigt sich, wenn man längere Stücke setzt.

```
<<
  \new Staff = ossia \with {
    \remove "Time_signature_engraver"
    \override Clef #'transparent = ##t
    fontSize = #-3
    \override StaffSymbol #'staff-space = #(magstep -3)
    \override StaffSymbol #'thickness = #(magstep -3)
  }
  { \stopStaff s1*6 }
```

```
  \new Staff \relative c' {
    c4 b c2
    <<
      { e4 f e2 }
      \context Staff = ossia {
        \startStaff e4 g8 f e2 \stopStaff
      }
    >>
    g4 a g2 \break
    c4 b c2
    <<
      { g4 a g2 }
      \context Staff = ossia {
        \startStaff g4 e8 f g2 \stopStaff
      }
    >>
    e4 d c2
  }
>>
```
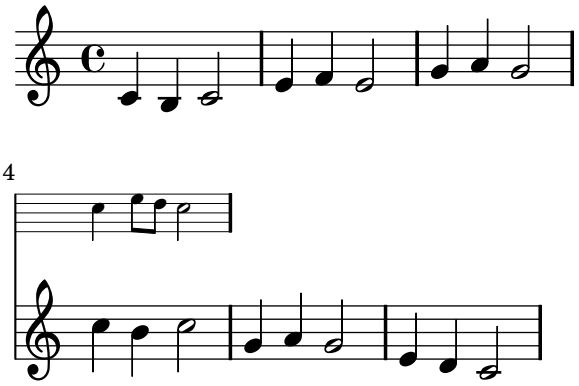


Man kann auch den `\Staff \RemoveEmptyStaves`-Befehl einsetzen, um Ossia-Systeme zu erstellen. Diese Methode eignet sich am besten, wenn nach dem Ossia sofort ein Zeilenumbruch erfolgt. Mehr Information zu `\Staff \RemoveEmptyStaves` findet sich in [Systeme verstecken], Seite 175.

```
<<
  \new Staff = ossia \with {
    \remove "Time_signature_engraver"
    \override Clef #'transparent = ##t
    fontSize = #-3
    \override StaffSymbol #'staff-space = #(magstep -3)
    \override StaffSymbol #'thickness = #(magstep -3)
  } \relative c'' {
    R1*3
    c4 e8 d c2
  }
  \new Staff \relative c' {
    c4 b c2
```

```
    e4 f e2
    g4 a g2 \break
    c4 b c2
    g4 a g2
    e4 d c2
  }
>>

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    \override VerticalAxisGroup #'remove-first = ##t
  }
}
```



Ausgewählte Schnipsel

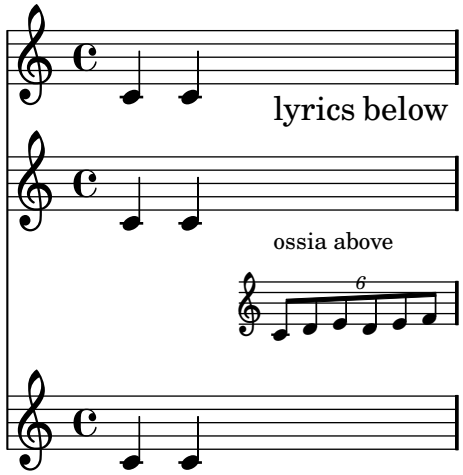
Gesangstext und Ossia vertikal ausrichten

Dieser Schnipsel zeigt, wie man die Kontexteigenschaften `alignBelowContext` und `alignAboveContext` benutzen kann, um die Positionierung von Gesangstext und Ossia-Abschnitten zu kontrollieren.

```
\paper {
  ragged-right = ##t
}

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = #"1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol #'staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
```

```
      } {
        \times 4/6 {
          \override TextScript #'padding = #3
          c8[^"ossia above" d e d e f]
        }
      }
    >>
  }
>>
```



Siehe auch

Glossar: Abschnitt “ossia” in *Glossar*, Abschnitt “staff” in *Glossar*, Abschnitt “Frenched staff” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Musikalische Ausdrücke ineinander verschachteln” in *Handbuch zum Lernen*, Abschnitt “Größe von Objekten” in *Handbuch zum Lernen*, Abschnitt “Länge und Dicke von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: [Systeme verstecken], Seite 175.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “StaffSymbol” in *Referenz der Interna*.

Systeme verstecken

Die Notenlinien können entfernt werden, indem der `Staff_symbol_engraver` aus dem `Staff`-Kontext entfernt wird. Alternativ kann auch `\stopStaff` eingesetzt werden.

```
\new Staff \with {
  \remove "Staff_symbol_engraver"
}
\relative c''' { a8 f e16 d c b a2 }
```



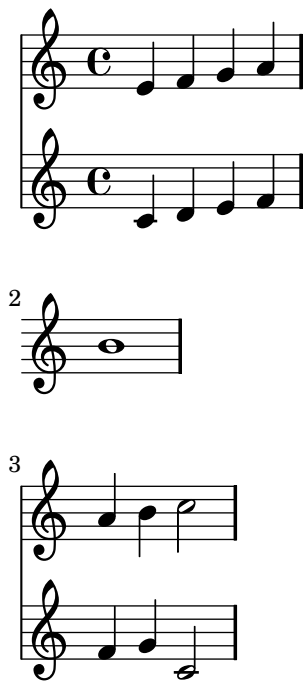
Leere Systeme können versteckt werden, wenn der `\Staff \RemoveEmptyStaves`-Befehl im `\layout`-Abschnitt benutzt wird. In großen Orchesterpartituren wird dies oft verwendet, um die

leeren Systeme von gerade nicht spielenden Instrumenten zu verstecken. In der Standardeinstellung werden alle leeren Notenzeilen außer die des ersten Systems entfernt.

Achtung: Eine Notenzeile gilt als leer, wenn sie nur Ganztaktpausen, Pausen, unsichtbare Noten, `\skip`-Befehle oder eine Kombination der drei enthält.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}

\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
    a4 b c2
  }
  \new Staff {
    c,4 d e f \break
    R1 \break
    f4 g c,2
  }
>>
```



`\Staff \RemoveEmptyStaves` kann auch eingesetzt werden, um Ossiaabschnitte zu erstellen. Zu Einzelheiten, siehe [Ossia-Systeme], Seite 171.

Der `\VaticanaStaff \RemoveEmptyStaves`-Befehl kann benutzt werden, um leere Takte in Notation der Alten Musik zu entfernen. Gleichermaßen kann `\RhythmicStaff \RemoveEmptyStaves` eingesetzt werden, um leere Takte in einem `RhythmicStaff`-Kontext zu entfernen.

Vordefinierte Befehle

`\Staff \RemoveEmptyStaves`, `\VaticanaStaff \RemoveEmptyStaves`, `\RhythmicStaff \RemoveEmptyStaves`.

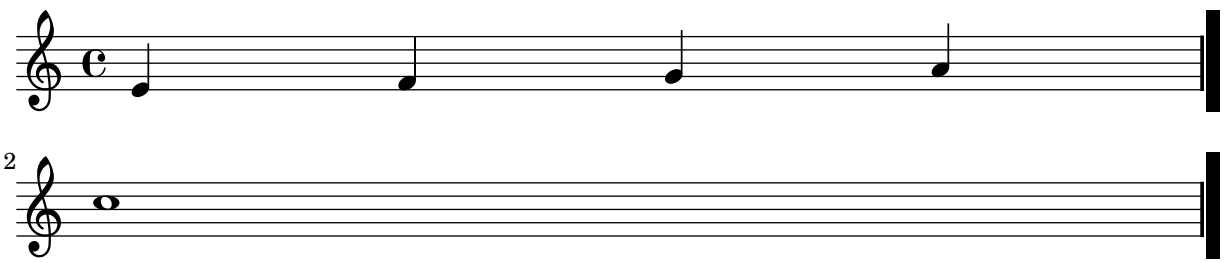
Ausgewählte Schnipsel

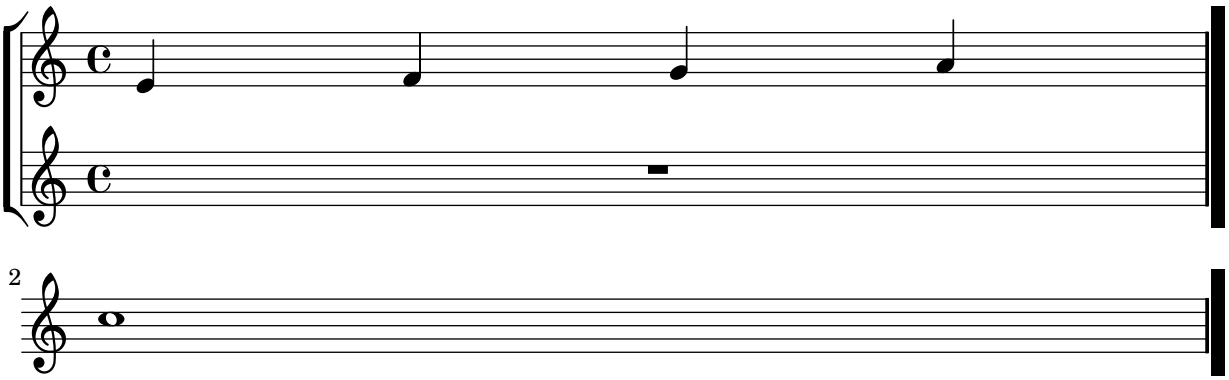
Die erste leere Notenzeile auch entfernen

Ein leeres Notensystem kann auch aus der ersten Zeile einer Partitur entfernt werden, indem die Eigenschaft `remove-first` der `VerticalAxisGroup`-Eigenschaft eingesetzt wird. Das kann man global in einer `\layout`-Umgebung oder lokal in dem bestimmten Notensystem machen, das entfernt werden soll. In letzterem Fall muss man den Kontext angeben.

Das untere Notensystem der zweiten Systemgruppe wird nicht entfernt, weil in die Einstellungen in dem Schnipsel nur für das eine Notensystem gültig sind.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
    % To use the setting globally, uncomment the following line:
    % \override VerticalAxisGroup #'remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup #'remove-first = ##t
    R1 \break
    R
  }
>>
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    R1 \break
    R
  }
>>
```





Siehe auch

Glossar: Abschnitt “Frenched staff” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Sichtbarkeit und Farbe von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 5.1.5 [Die Standardeinstellungen von Kontexten ändern], Seite 521, [Das Notensystem], Seite 169, [Ossia-Systeme], Seite 171, [Unsichtbare Noten], Seite 193, Abschnitt 5.4.7 [Sichtbarkeit von Objekten], Seite 546.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “ChordNames” in *Referenz der Interna*, Abschnitt “FiguredBass” in *Referenz der Interna*, Abschnitt “Lyrics” in *Referenz der Interna*, Abschnitt “Staff” in *Referenz der Interna*, Abschnitt “VerticalAxisGroup” in *Referenz der Interna*, Abschnitt “Staff_symbol_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Wenn man den `Staff_symbol_engraver` entfernt, werden auch die Taktlinien entfernt. Wenn eine sichtbare Taktlinie angefordert wird, kann es zu Formatierungsfehlern kommen. In diesem Fall sollten folgende Befehle eingesetzt werden, anstatt den Engraver zu entfernen:

```
\override StaffSymbol #'stencil = ##f
\override NoteHead #'no-ledgers = ##t
```

Zu den bekannten Fehlern und Warnungen, die mit `\Staff \RemoveEmptyStaves` zusammenhängen, siehe Abschnitt 5.1.5 [Die Standardeinstellungen von Kontexten ändern], Seite 521.
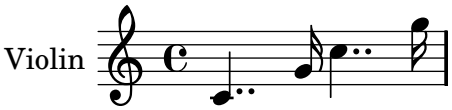
1.6.3 Orchesterstimmen erstellen
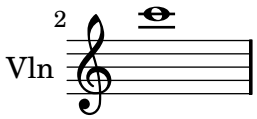
Dieser Abschnitt zeigt, wie man Tempo-Anweisungen und Instrumentenbezeichnungen einfügt. Es werden auch Möglichkeiten vorgestellt, andere Stimmen zu zitieren und Stichnoten zu formatieren.

Instrumentenbezeichnungen

Instrumentbezeichnungen können an der linken Seite von Notensystemen im `Staff`-, `PianoStaff`-, `StaffGroup`, `GrandStaff` und `ChoirStaff`-Kontext gesetzt werden. Der Wert von `instrumentName` wird für das erste System eingesetzt, der Wert von `shortInstrumentName` für alle weiteren Systeme.
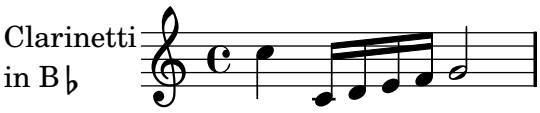
```
\set Staff.instrumentName = #"Violin "
\set Staff.shortInstrumentName = #"Vln "
c4.. g'16 c4.. g'16
\break
c1
```
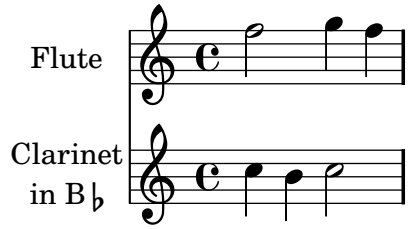




Mit dem Textbeschriftungsmodus können auch komplizierte Instrumentenbezeichnungen erstellt werden:

```
\set Staff.instrumentName = \markup {
  \column { "Clarinetti"
            \line { "in B" \smaller \flat } } }
c4 c,16 d e f g2
```



Wenn zwei oder mehr Systeme gruppiert werden, werden die Instrumentenbezeichnungen automatisch zentriert. Um auch mehrzeilige Instrumentenbezeichnungen zentriert zu setzen, muss `\center-column` benutzt werden:

```
<<
  \new Staff {
    \set Staff.instrumentName = #"Flute"
    f2 g4 f
  }
  \new Staff {
    \set Staff.instrumentName = \markup \center-column {
      Clarinet
      \line { "in B" \smaller \flat }
    }
    c4 b c2
  }
>>
```
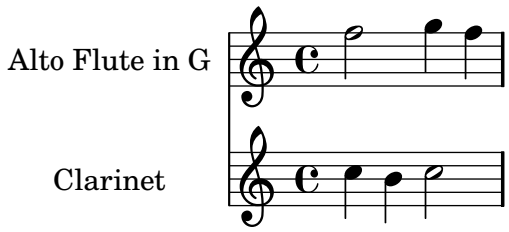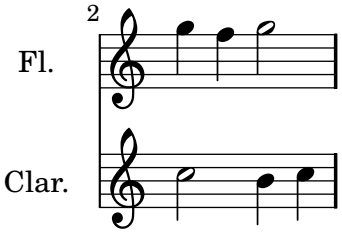


Wenn die Instrumentenbezeichnung zu lang ist, kann es vorkommen, dass die Bezeichnungen in einer Gruppe nicht zentriert werden. Um dennoch eine Zentrierung zu erhalten, müssen die Werte des Einzugs (`indent` und `short-indent`) vergrößert werden. Zu Einzelheiten siehe [\paper-Variablen für Verschiebungen und Einrückungen], Seite 468.

```
\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm
}

\relative c'' <<
  \new Staff {
    \set Staff.instrumentName = #"Alto Flute in G"
    \set Staff.shortInstrumentName = #"Fl."
    f2 g4 f \break
```

```
    g4 f g2
  }
  \new Staff {
    \set Staff.instrumentName = #"Clarinet"
    \set Staff.shortInstrumentName = #"Clar."
    c,4 b c2 \break
    c2 b4 c
  }
>>
```

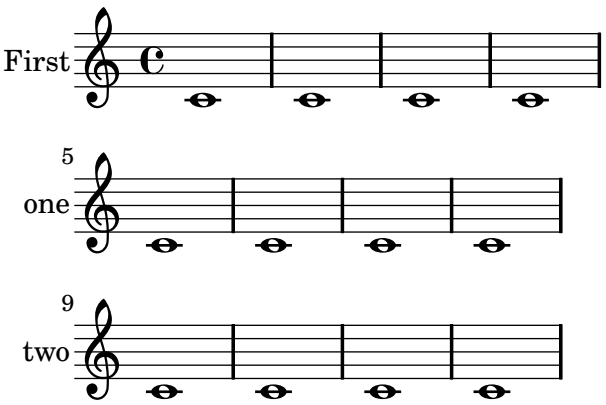


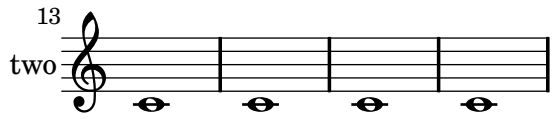


Um Instrumentenbezeichnungen zu anderen Kontexten (wie etwa `ChordNames` or `FiguredBass`) hinzuzufügen, muss der `Instrument_name_engraver` dem entsprechenden Kontext hinzugefügt werden. Zu Einzelheiten siehe Abschnitt 5.1.4 [Umgebungs-Plugins verändern], Seite 519.

Instrumentenbezeichnungen können mitten in einer Partitur geändert werden. Dabei muss jedoch beachtet werden, dass `instrumentName` nicht mitten im Stück angezeigt wird, denn es wird nur für das erste Notensystem ausgegeben:

```
\set Staff.instrumentName = #"First"
\set Staff.shortInstrumentName = #"one"
c1 c c c \break
c1 c c c \break
\set Staff.instrumentName = #"Second"
\set Staff.shortInstrumentName = #"two"
c1 c c c \break
c1 c c c \break
```

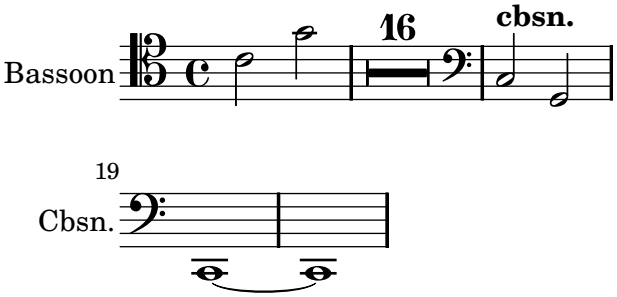




Wenn das Instrument gewechselt werden soll, kann der Befehl `\addInstrumentDefinition` in Begleitung von `\instrumentSwitch` benutzt werden, um eine detaillierte Auflistung aller notwendigen Änderungen für den Wechsel zu definieren. Der `\addInstrumentDefinition`-Befehl hat zwei Argumente: eine Identifikation und eine Assoziationsliste von Kontexteigenschaften und Werten, die für dieses Instrument benutzt werden müssen. Der Befehl muss sich auf der höchsten Ebene in der Eingabedatei befinden. `\instrumentSwitch` wird dann benutzt, um den Wechsel vorzunehmen:

```
\addInstrumentDefinition #"contrabassoon"
  #`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
     (shortInstrumentName . "Cbsn.")
     (clefGlyph . "clefs.F")
     (middleCPosition . 6)
     (clefPosition . 2)
     (instrumentCueName . ,(make-bold-markup "cbsn."))
     (midiInstrument . "bassoon"))

\new Staff \with {
  instrumentName = #"Bassoon"
}
\relative c' {
  \clef tenor
  \compressFullBarRests
  c2 g'
  R1*16
  \instrumentSwitch "contrabassoon"
  c,,2 g \break
  c,1 ~ | c1
}
```



Siehe auch

Notationsreferenz: [\paper-Variablen für Verschiebungen und Einrückungen], Seite 468, Abschnitt 5.1.4 [Umgebungs-Plugins verändern], Seite 519.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “InstrumentName” in *Referenz der Interna*, Abschnitt “PianoStaff” in *Referenz der Interna*, Abschnitt “Staff” in *Referenz der Interna*.

Stichnoten

Es kommt sehr oft vor, dass eine Orchesterstimme die gleichen Noten wie eine andere spielt. So können etwa die ersten und zweiten Geigen für eine Passage die gleichen Noten haben. In

LilyPond kann man das erreichen, indem eine Stimme von der anderen *zitiert*, sodass man die Noten nicht noch einmal eingeben muss.
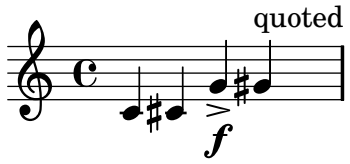
Bevor eine Stimme zitiert werden kann, muss der `\addQuote`-Befehl benutzt werden, um das zitierbare Fragment zu kennzeichnen. Dieser Befehl muss auf der höchsten Ebene der Eingabedatei benutzt werden. Das erste Argument dient zur Identifikation, das zweite ein musikalischer Ausdruck:

```
flute = \relative c'' {
  a4 gis g gis
}
\addQuote "flute" { \flute }
```

Der `\quoteDuring`-Befehl wird benutzt, um den Punkt anzuzeigen, an dem das Zitat beginnt. Er benötigt zwei Argumente: die Bezeichnung der zitierten Stimme, wie vorher mit `\addQuote` definiert, und einen musikalischen Ausdruck, der Angibt, wie lange das Zitat dauern soll; normalerweise Ganztaktpausen oder unsichtbare Noten. Die entsprechenden Noten der zitierten Stimme (inklusive aller Artikulationszeichen, Dynamik, Beschriftung usw.) wird an der Stelle in die aktuelle Stimme eingefügt:

```
flute = \relative c'' {
  a4 gis g->\f gis^\markup{quoted}
}
\addQuote "flute" { \flute }
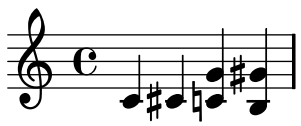
\relative c' {
  c4 cis \quoteDuring #"flute" { s2 }
}
```



Wenn der musikalische Ausdruck, der mit dem `\quoteDuring`-Befehl benutzt wird, etwas anderes als unsichtbare Noten oder Ganztaktpausen enthält, wird eine polyphone Stelle begonnen, was meistens nicht erwünscht ist:

```
flute = \relative c'' {
  a4 gis g gis
}
\addQuote "flute" { \flute }
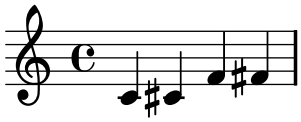
\relative c' {
  c4 cis \quoteDuring #"flute" { c4 b }
}
```



Zitate erkennen die Einstellungen von transponierten Instrumenten sowohl der Quell- als auch der Zielstimme, wenn der `\transposition`-Befehl eingesetzt wird. Zu Einzelheiten über `\transposition` siehe [Transposition von Instrumenten], Seite 22.

```
clarinet = \relative c'' {
  \transposition bes
  a4 gis g gis
}
\addQuote "clarinet" { \clarinet }

\relative c' {
  c4 cis \quoteDuring #"clarinet" { s2 }
}
```
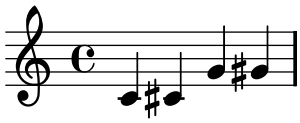


Es ist möglich, Zitate mit eindeutigen Bezeichnungen zu versehen (unter Benutzung von *tags*), um sie auf unterschiedliche Weise zu verarbeiten. Einzelheiten zu diesem Vorgehen werden vorgestellt in [Marken benutzen], Seite 444.

Es ist auch möglich, welche Objekte der originalen Stimme zitiert werden sollen, indem man die `quotedEventTypes`-Eigenschaft verändert. Standardmäßig ist ihr Wert `#'(StreamEvent)`, was bedeutet, dass alles zitiert wird. Wenn man sie beispielsweise auf den Wert `#'(note-event rest-event tie-event)` setzt, werden nur Noten, Pausen und Bindebögen zitiert, jedoch keine Artikulationszeichen, Dynamik oder Beschriftung.

```
clarinet = \relative c'' {
  a4 gis g->\f gis^\markup{quoted}
}
\addQuote "clarinet" { \clarinet }

\relative c' {
  \set Score.quotedEventTypes = #'(note-event rest-event tie-event)
  c4 cis \quoteDuring #"clarinet" { s2 }
}
```



Ausgewählte Schnipsel

Eine Stimme mit Transposition zitieren

Zitate berücksichtigen sowohl die Transposition der Quelle als auch des Zielinstruments. In diesem Beispiel spielen alle Instrumente klingendes C, das Zielinstrument ist in F. Die Noten für das Zielinstrument können mit `\transpose` transponiert werden, in diesem Fall werden alle Noten (auch die zitierten) transponiert.

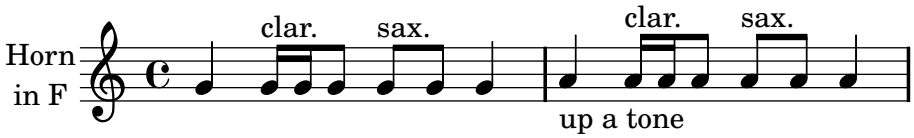
```
\addQuote clarinet {
  \transposition bes
  \repeat unfold 8 { d'16 d' d'8 }
}

\addQuote sax {
  \transposition es'
  \repeat unfold 16 { a8 }
```

```
}

quoteTest = {
  % french horn
  \transposition f
  g'4
  << \quoteDuring #"clarinet" { \skip 4 } s4^"clar." >>
  << \quoteDuring #"sax" { \skip 4 } s4^"sax." >>
  g'4
}

{
  \set Staff.instrumentName =
    \markup {
      \center-column { Horn \line { in F } }
    }
  \quoteTest
  \transpose c' d' << \quoteTest s4_"up a tone" >>
}
```



Eine andere Stimme zitieren
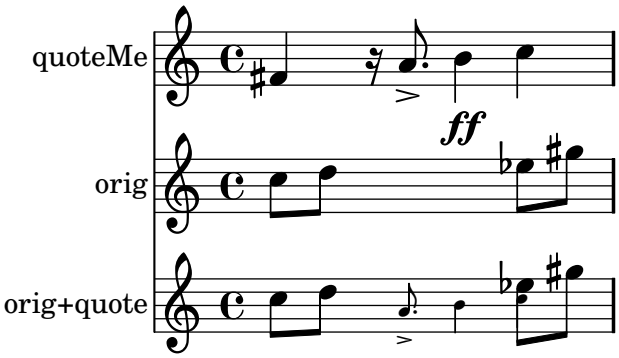
Die `quotedEventTypes`-Eigenschaft bestimmt die Typen an Musikereignissen, die zitiert werden. Die Standardeinstellung ist `(note-event rest-event)`, womit nur Noten und Pausen der zitierten Stimme für den `\quoteDuring`-Ausdruck übernommen werden. Im Beispiel hier wird die 16-Pause nicht übernommen, weil sich `rest-event` nicht in `quotedEventTypes` befindet.

```
quoteMe = \relative c' {
  fis4 r16 a8.-> b4\ff c
}
\addQuote quoteMe \quoteMe

original = \relative c'' {
  c8 d s2
  \once \override NoteColumn #'ignore-collision = ##t
  es8 gis8
}

<<
  \new Staff {
    \set Staff.instrumentName = #"quoteMe"
    \quoteMe
  }
  \new Staff {
    \set Staff.instrumentName = #"orig"
    \original
  }
  \new Staff \relative c'' <<
    \set Staff.instrumentName = #"orig+quote"
```

```
    \set Staff.quotedEventTypes =
      #'(note-event articulation-event)
    \original
    \new Voice {
      s4
      \set fontSize = #-4
      \override Stem #'length-fraction = #(magstep -4)
      \quoteDuring #"quoteMe" { \skip 2. }
    }
  >>
>>
```



Siehe auch

Notationsreferenz: [Transposition von Instrumenten], Seite 22, [Marken benutzen], Seite 444.

Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “QuoteMusic” in *Referenz der Interna*, Abschnitt “Voice” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Nur der Inhalt der ersten Stimme innerhalb eines `\addQuote`-Befehls wird für das Zitat herangezogen, die Variable *Noten* kann also keine `\new` oder `\context Voice`-Einheiten enthalten, die zu einer anderen Stimme wechseln würden.

Ziernoten und Vorschläge können nicht zitiert werden und können sogar dazu führen, dass LilyPond abstürzt.

Wenn geschachtelte Triolen zitiert werden, ist das Notenbild unter Umständen sehr schlecht.

In früheren LilyPond-Versionen (vor 2.11) wurde der Befehl `addQuote` vollständig in Kleinbuchstaben geschrieben: `\addquote`.

Stichnoten formatieren
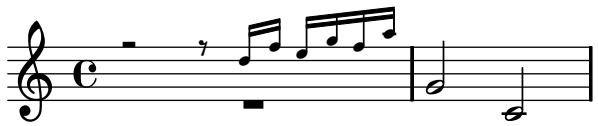
Der vorige Abschnitt zeigt, wie man Zitate erstellt. Der `\cueDuring`-Befehl (engl. cue note = Stichnote) ist eine spezialisierte Form des `\quoteDuring`-Befehls, der insbesondere dazu dient, Stichnoten zu einer Stimme hinzuzufügen. Seine Syntax lautet:

`\cueDuring #`*`Stimmenbezeichnung`* `#`*`Stimme Noten`*

Dieser Befehl kopiert nur die Noten und Pausen der entsprechenden Takte von *Stimmenbezeichnung* in einen `CueVoice`-Kontext. Eine `CueVoice` (Stichnoten-Stimme) wird implizit erstellt und erscheint simultan mit *Noten*, wobei folglich eine polyphone Situation entsteht. Das *Stimme*-Argument entscheidet, ob die Stichnoten als eine erste oder zweite Stimme eingefügt werden sollen; `UP` entspricht der ersten Stimme, `DOWN` der zweiten.

```
oboe = \relative c'' {
  r2 r8 d16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboe }

\new Voice \relative c'' {
  \cueDuring #"oboe" #UP { R1 }
  g2 c,
}
```

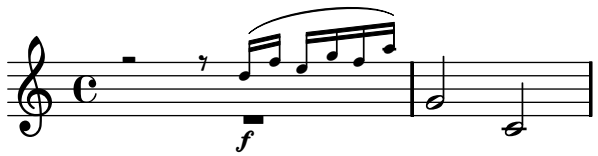


In diesem Beispiel muss der `Voice`-Kontext explizit begonnen werden, damit nicht der gesamte musikalische Ausdruck als Stichnoten-Stimme formatiert wird.

Es ist möglich anzupassen, welche Objekte der Notation von `\cueDuring` zitiert werden, indem man die `quotedCueEventTypes`-Eigenschaft verändert. Ihr Standardwert ist `#'(note-event rest-event tie-event beam-event +tuplet-span-event)`; somit werden also nur Noten, Pausen, Bindebögen, Balken und N-tolen zitiert, nicht aber Artikulationen, Dynamik, Beschriftung usw.

```
oboe = \relative c'' {
  r2 r8 d16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboe }

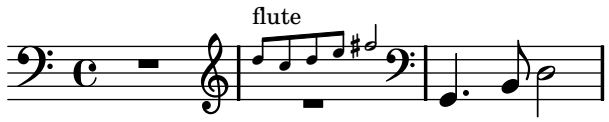
\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                      beam-event tuplet-span-event
                                      dynamic-event slur-event)
  \cueDuring #"oboe" #UP { R1 }
  g2 c,
}
```



Beschriftung kann auch benutzt werden, um die Bezeichnung des zitierten Instruments anzuzeigen. Wenn die Stichnoten einen Schlüsselwechsel erfordern, muss der originale Schlüssel am Ende der Stichnoten wieder hergestellt werden:

```
flute = \relative c'' {
  r2. c4 d8 c d e fis2 g2 d2
}
bassoon = \relative c {
  \clef bass
  R1
  \clef treble
```
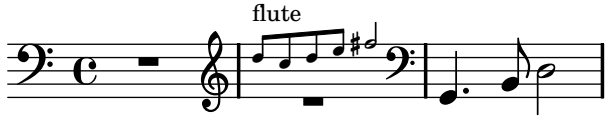
```
  s1*0^\markup { \tiny "flute" }
  \cueDuring #"flute" #UP { R1 }
  \clef bass
  g4. b8 d2
}
\addQuote "flute" { \flute }
\new Staff {
  \bassoon
}
```

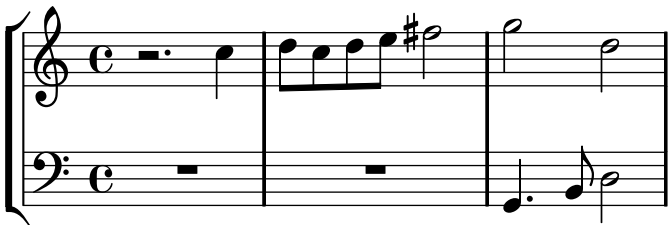


Der `\killCues`-Befehl entfernt Stichnoten aus einem musikalischen Ausdruck, sodass der selbe Ausdruck für eine Stimme mit den Stichnoten, aber auch für die Paritur eingesetzt werden kann. Der Befehl `\killCues` entfernt nur die Noten und Ereignisse, die mit `\cueDuring` zitiert worden sind. Andere mit der Beschriftungen verknüpfte Ereignisse wie Schlüsseländerungen und Bezeichnungen des Instruments können mit einem Marker versehen werden, um selektiv benutzt werden zu können. Siehe [Marken benutzen], Seite 444.

```
flute = \relative c'' {
  r2. c4 d8 c d e fis2 g2 d2
}
bassoon = \relative c {
  \clef bass
  R1
  \tag #'part {
    \clef treble
    s1*0^\markup { \tiny "flute" }
  }
  \cueDuring #"flute" #UP { R1 }
  \tag #'part \clef bass
  g4. b8 d2
}
\addQuote "flute" { \flute }

\new Staff {
  \bassoon
}
\new StaffGroup <<
  \new Staff {
    \flute
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoon } }
  }
>>
```





Alternativ können Schlüsseländerungen und Instrumentbezeichnungen auch in eine Instrument-Definition zur wiederholten Verwendung gesammelt werden, wobei `\addInstrumentDefinition` eingesetzt wird, das sich in [Instrumentenbezeichnungen], Seite 178 beschrieben findet.

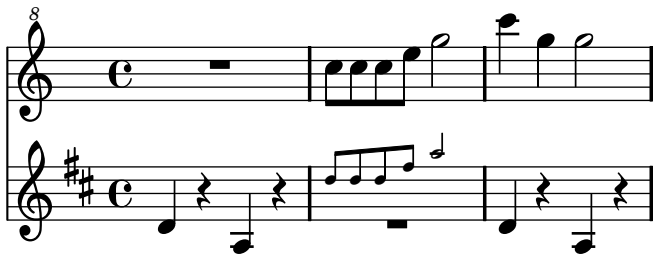
Genauso wie `\quoteDuring` berücksichtigt auch `\cueDuring` Transpositionen. Stichnoten werden auf den Tonhöhen erstellt, die für das Instrument geschrieben würden, in dessen Noten sie gesetzt werden, um die klingenden Töne des Quelleninstruments zu produzieren.

Um Stichnoten anders zu transponieren, muss `\transposedCueDuring` benutzt werden. Dieser Befehl braucht ein zusätzliches Argument, um (in absolutem Modus) die gedruckte Tonhöhe vorzugeben, mit der das das zweigestrichene C dargestellt werden soll. Das ist nützlich, wenn man Stichnoten von einem Instrument mit einem vollständig anderen Register benutzt:

```
piccolo = \relative c''' {
  \clef "treble^8"
  R1
  c8 c c e g2
  c4 g g2
}
bassClarinet = \relative c' {
  \key d \major
  \transposition bes,
  d4 r a r
  \transposedCueDuring #"piccolo" #UP d { R1 }
  d4 r a r
}

\addQuote "piccolo" { \piccolo }
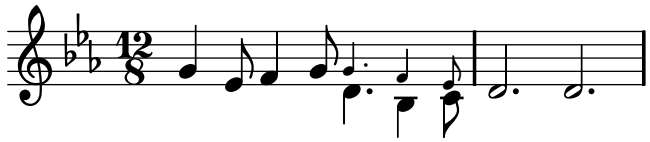
<<
  \new Staff \piccolo
  \new Staff \bassClarinet
>>
```



Ein `CueVoice`-Kontext kann auch explizit erstellt werden, wenn man kleiner Noten einsetzen will. Damit kann beispielsweise eine alternative Sequenz für hohe und tiefe Stimme gesetzt werden:

```
\time 12/8
\key ees \major
g4 ees8 f4 g8
```

```
\stemDown
<<
  { d4. bes4 c8 }
  \new CueVoice
  { g'4. f4 ees8 }
>>
\stemUp
d2. d2.
```



Siehe auch

Notationsreferenz: [Transposition von Instrumenten], Seite 22, [Instrumentenbezeichnungen], Seite 178, [Marken benutzen], Seite 444, [Musikalische Stichnoten], Seite 269.
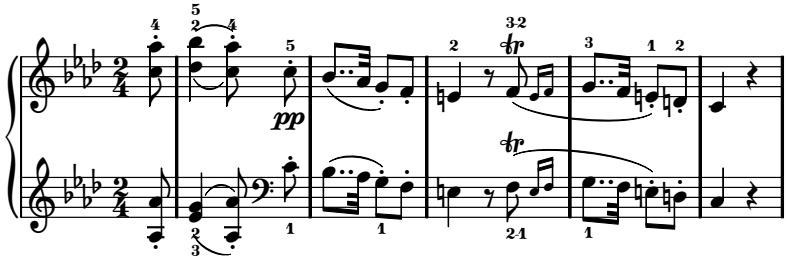
Schnipsel: Abschnitt “Staff notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “CueVoice” in *Referenz der Interna*, Abschnitt “Voice” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Zusammenstöße können zwischen Pausen der Hauptstimme und den Stichnoten des `CueVoice`-Kontexts auftreten.

1.7 Anmerkungen



Dieser Abschnitt zeigt die verschiedenen Möglichkeiten, die Erscheinung der Noten zu ändern und analytische bzw. pädagogische Anmerkungen anzubringen.

1.7.1 Innerhalb des Systems
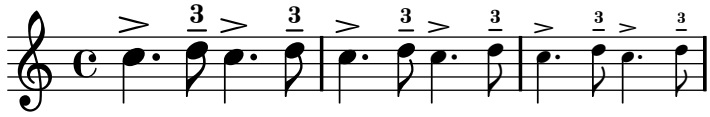
Dieser Abschnitt zeigt, wie man Elemente hervorhebt, die sich innerhalb des Notensystems befinden.

Auswahl der Notations-Schriftgröße

Die Schriftgröße von Notationselementen kann geändert werden. Damit wird allerdings nicht die Größe von veränderlichen Symbolen, wie Balken oder Bögen, geändert.
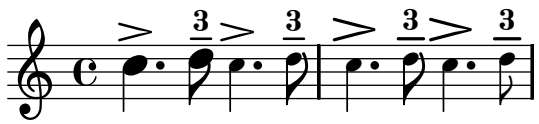
Achtung: Für Schriftgröße von Text, siehe [Überblick über die wichtigsten Textbeschriftungsbefehle], Seite 211.

```
\huge
c4.-> d8---3
\large
c4.-> d8---3
\normalsize
c4.-> d8---3
\small
c4.-> d8---3
\tiny
c4.-> d8---3
\teeny
c4.-> d8---3
```



Intern wird hiermit die `fontSize`-Eigenschaft gesetzt. Sie wird für alle Layout-Objekte definiert. Der Wert von `font-size` ist eine Zahl, die die Größe relativ zur Standardgröße für die aktuelle Systemhöhe angibt. Jeder Vergrößerungsschritt bedeutet etwa eine Vergrößerung um 12% der Schriftgröße. Mit sechs Schritten wird die Schriftgröße exakt verdoppelt. Die Scheme-Funktion `magstep` wandelt einen Wert von `font-size` in einen Skalierungsfaktor um. Die `font-size`-Eigenschaft kann auch direkt gesetzt werden, so dass sie sich nur auf bestimmte Layoutobjekte bezieht.

```
\set fontSize = #3
c4.-> d8---3
\override NoteHead #'font-size = #-4
c4.-> d8---3
\override Script #'font-size = #2
c4.-> d8---3
\override Stem #'font-size = #-5
c4.-> d8---3
```



Schriftgrößenänderungen werden erreicht, indem man die Design-Schriftgröße nimmt, die der gewünschten am nächsten kommt, und sie dann skaliert. Die Standard-Schriftgröße (für `font-size = #0`) hängt von der Standard-Systemhöhe ab. Für ein Notensystem von 20pt wird eine Schriftgröße von 10pt ausgewählt.

Die `font-size`-Eigenschaft kann nur für die Layoutobjekte gesetzt werden, die Schrift-Dateien benutzen. Das sind die, welche die `font-interface`-Layoutschnittstelle unterstützen.

Vordefinierte Befehle

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

Siehe auch
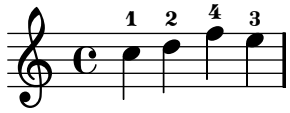
Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “font-interface” in *Referenz der Interna*.

Fingersatzanweisungen
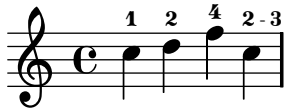
Fingersatzanweisungen können folgenderweise notiert werden: '`Note`'-*Zahl*

```
c4-1 d-2 f-4 e-3
```



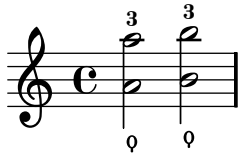
Für Fingerwechsel muss eine Textbeschriftung (markup) benutzt werden:

```
c4-1 d-2 f-4 c^\markup { \finger "2 - 3" }
```
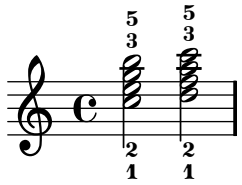


Mit dem Daumen-Befehl (`\thumb`) können die Noten bezeichnet werden, die mit dem Daumen (etwa auf dem Cello) gespielt werden sollen.

```
<a_\thumb a'-3>2 <b_\thumb b'-3>
```



Fingersätze für Akkorde können auch zu einzelnen Noten hinzugefügt werden, indem sie innerhalb der Akkord-Klammer direkt an die Noten angefügt werden.

```
<c-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
```



Fingersatzanweisungen können manuell oberhalb des Systems gesetzt werden, siehe Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

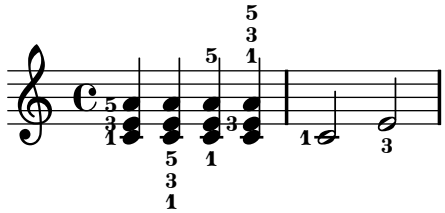
Ausgewählte Schnipsel

Position von Fingersatz in Akkorden kontrollieren

Die Position von Fingersatzzahlen kann exakt kontrolliert werden.

```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
```

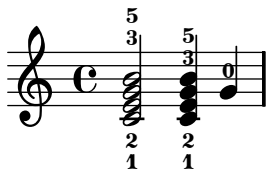
```
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Fingersatz auch innerhalb des Systems setzen

Normalerweise werden vertikal orientierte Fingersatzzahlen außerhalb des Systems gesetzt. Das kann aber verändert werden.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering #'staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 <g'-0>
}
```



Avoiding collisions with chord fingerings

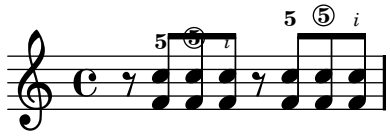
Fingerings and string numbers applied to individual notes will automatically avoid beams and stems, but this is not true by default for fingerings and string numbers applied to the individual notes of chords. The following example shows how this default behavior can be overridden.

```
\relative c' {
  \set fingeringOrientations = #'(up)
  \set stringNumberOrientations = #'(up)
  \set strokeFingerOrientations = #'(up)

  % Default behavior
  r8
  <f c'-5>8
  <f c'\5>8
  <f c'-\rightHandFinger #2 >8

  % Corrected to avoid collisions
  r8
  \override Fingering #'add-stem-support = ##t
  <f c'-5>8
  \override StringNumber #'add-stem-support = ##t
  <f c'\5>8
  \override StrokeFinger #'add-stem-support = ##t
```

```
  <f c'-\rightHandFinger #2 >8
}
```



Siehe auch

Notationsreferenz: Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “FingeringEvent” in *Referenz der Interna*, Abschnitt “fingering-event” in *Referenz der Interna*, Abschnitt “Fingering_engraver” in *Referenz der Interna*, Abschnitt “New_fingering_engraver” in *Referenz der Interna*, Abschnitt “Fingering” in *Referenz der Interna*.

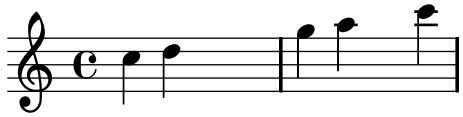
Bekannte Probleme und Warnungen

Standardmäßig ist eine Zahl größer als 9 nicht unterstützt, wenn man die Schreibweise *`Note-Zahl`* einsetzt.

Unsichtbare Noten
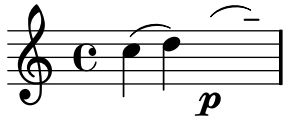
Versteckte (oder unsichtbare oder transparente) Noten können sinnvoll sein, wenn man Notation für den Theorieunterricht oder Kompositionsübungen erstellen will.

```
c4 d
\hideNotes
e4 f
\unHideNotes
g a
\hideNotes
b
\unHideNotes
c
```



Notationsobjekte, die an die unsichtbaren Noten angefügt sind, sind weiterhin sichtbar.

```
c4( d)
\hideNotes
e4(\p f)--
```



Vordefinierte Befehle

`\hideNotes`, `\unHideNotes`.

Siehe auch

Handbuch zum Lernen: Abschnitt “Sichtbarkeit und Farbe von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: [Unsichtbare Pausen], Seite 53, Abschnitt 5.4.7 [Sichtbarkeit von Objekten], Seite 546, [Systeme verstecken], Seite 175.

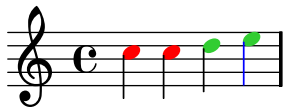
Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “Note_spacing_engraver” in *Referenz der Interna*, Abschnitt “NoteSpacing” in *Referenz der Interna*.

Farbige Objekte

Einzelnen Objekten können einfach eigene Farben zugewiesen werden. Gültige Farben-Bezeichnungen sind aufgelistet in Abschnitt A.6 [Liste der Farben], Seite 577.

```
\override NoteHead #'color = #red
c4 c
\override NoteHead #'color = #(x11-color 'LimeGreen)
d
\override Stem #'color = #blue
e
```
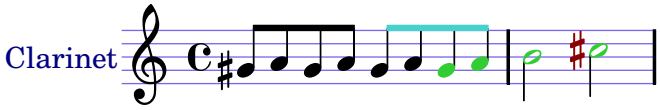


Die ganze Farbpalette, die für X11 definiert ist, kann mit der Scheme-Funktion `x11-color` benutzt werden. Diese Funktion hat ein Argument: entweder ein Symbol in der Form `'FooBar` oder eine Zeichenkette in der Form `"FooBar"`. Die erste Form ist schneller zu schreiben und effizienter. Mit der zweiten Form ist es allerdings möglich, auch Farbbezeichnungen einzusetzen, die aus mehr als einem Wort bestehen.

Wenn `x11-color` die angegebene Farbbezeichnung nicht kennt, wird Schwarz eingesetzt.

```
\override Staff.StaffSymbol #'color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}
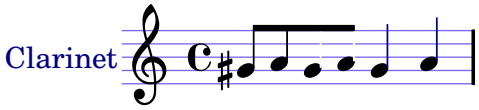
gis8 a
\override Beam #'color = #(x11-color "medium turquoise")
gis a
\override Accidental #'color = #(x11-color 'DarkRed)
gis a
\override NoteHead #'color = #(x11-color "LimeGreen")
gis a
% this is deliberate nonsense; note that the stems remain black
\override Stem #'color = #(x11-color 'Boggle)
b2 cis
```



Exakte RGB-Farben können mit Hilfe der Scheme-Funktion `rgb-color` definiert werden.

```
\override Staff.StaffSymbol #'color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}

\override Stem #'color = #(rgb-color 0 0 0)
gis8 a
\override Stem #'color = #(rgb-color 1 1 1)
gis8 a
\override Stem #'color = #(rgb-color 0 0 0.5)
gis4 a
```



Siehe auch

Notationsreferenz: Abschnitt A.6 [Liste der Farben], Seite 577, Abschnitt 5.3.4 [Der tweak-Befehl], Seite 533.

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Bekannte Probleme und Warnungen

Eine X11-Farbe hat nicht notwendigerweise exakt denselben Farbton wie eine ähnlich genannte normale Farbe.
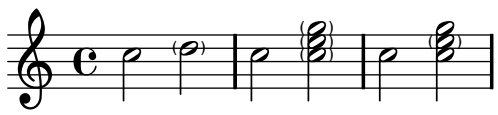
Nicht alle X11-Farben lassen sich am Webbrowser erkennen, d. h. der Unterschied etwa zwischen `'LimeGreen` und `'ForestGreen` wird eventuell nicht dargestellt. Für die Benutzung im Internet wird die Benutzung von einfachen Farben nahegelegt (z. B. `#blue`, `#green`, `#red`).

Noten in Akkorden können nicht mit `\override` eingefärbt werden, dazu muss `\tweak` benutzt werden. Siehe auch Abschnitt 5.3.4 [Der tweak-Befehl], Seite 533.

Klammern
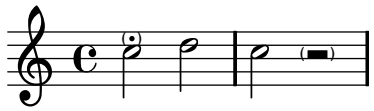
Objekte können in Klammern gesetzt werden, indem vor ihnen der Befehl `\parenthesize` geschrieben wird. Wenn ein Akkord in Klammern gesetzt wird, wirkt sich das auf jede Noten im Akkord aus. Innerhalb von einem Akkord gesetzte Befehle wirken sich auf einzelne Noten aus.

```
c2 \parenthesize d
c2 \parenthesize <c e g>
c2 <c \parenthesize e g>
```



Auch andere Objekte als Noten können in Klammern gesetzt werden. Wenn Artikulationszeichen in Klammern gesetzt werden sollen, braucht man ein Minuszeichen vor dem `\parenthesize`-Befehl.

```
c2-\parenthesize -. d
c2 \parenthesize r
```



Siehe auch

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “Parenthesis_engraver” in *Referenz der Interna*, Abschnitt “ParenthesesItem” in *Referenz der Interna*, Abschnitt “parentheses-interface” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Wenn man einen Akkord einklammert, wird um jede Note eine eigene Klammer gesetzt, anstatt den gesamten Akkord in eine große Klammer zu fassen.

Hälse

Immer, wenn das Programm eine Note findet, wird automatisch ein Notenhals (Abschnitt “Stem” in *Referenz der Interna*) -Objekt erzeugt. Auch für ganze Noten und Pausen werden sie erzeugt, aber unsichtbar gemacht.

Hälse können manuell gesetzt werden, um nach oben oder unten zu zeigen, siehe ⟨undefined⟩ [Direction and placement], Seite ⟨undefined⟩.

Vordefinierte Befehle
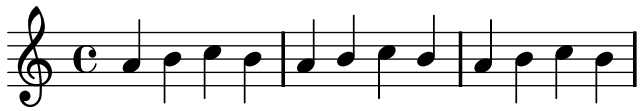
`\stemUp` (Hälse nach oben), `\stemDown` (Hälse nach unten), `\stemNeutral` (Hälse je nach Notenposition).

Ausgewählte Schnipsel

Standardrichtung für Hälse auf der Mittellinie

Die Richtung von Hälsen auf der mittleren Linie kann mit der `Stem`-Eigenschaft `neutral-direction` gesetzt werden.

```
\relative c'' {
  a4 b c b
  \override Stem #'neutral-direction = #up
  a4 b c b
  \override Stem #'neutral-direction = #down
  a4 b c b
}
```



Siehe auch

Notationsreferenz: Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “Stem_engraver” in *Referenz der Interna*, Abschnitt “Stem” in *Referenz der Interna*, Abschnitt “stem-interface” in *Referenz der Interna*.
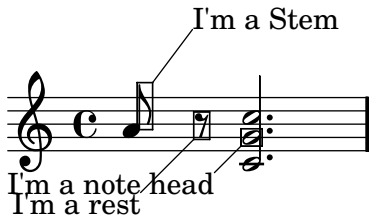
1.7.2 Außerhalb des Notensystems

Dieser Abschnitt zeigt, wie man Elemente im System von außerhalb des Systems hervorhebt.

Erklärungen in Ballonform

Notationselemente können bezeichnet und markiert werden, indem um sie eine rechteckige Blase gezeichnet wird. Dies ist vor allem dazu da, Notation zu erklären.

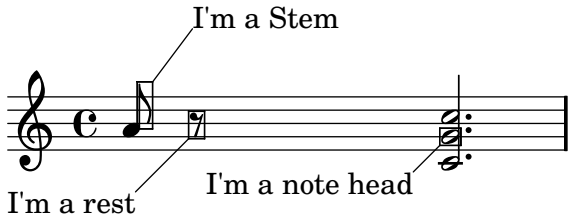
```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonGrobText #'Stem #'(3 . 4) \markup { "I'm a Stem" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "I'm a rest" }
  r
  <c, g'-\balloonText #'(-2 . -2) \markup { "I'm a note head" } c>2.
}
```



Es gibt zwei Funktionen, `balloonGrobText` und `balloonText`; die erste wird auf gleiche Art wie ein `\once \override` eingesetzt umd Text an einen Grob zu hängen, die zweite funktioniert wie ein `\tweak` und wird üblicherweise innerhalb von Akkorden eingesetzt, um Text an einzelne Noten zu hängen.

Textblasen beeinflussen normalerweise die Positionierung der Notation, aber das kann geändert werden.

```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonLengthOff
  \balloonGrobText #'Stem #'(3 . 4) \markup { "I'm a Stem" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "I'm a rest" }
  r
  \balloonLengthOn
  <c, g'-\balloonText #'(-2 . -2) \markup { "I'm a note head" } c>2.
}
```



Vordefinierte Befehle

`\balloonLengthOn`, `\balloonLengthOff`.

Siehe auch

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “Balloon_engraver” in *Referenz der Interna*, Abschnitt “BalloonTextItem” in *Referenz der Interna*, Abschnitt “balloon-interface” in *Referenz der Interna*.

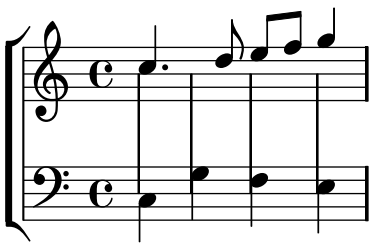
Gitternetzlinien

Vertikale Linien können zwischen Systemen gesetzt werden, die mit den Noten synchronisiert sind.

Der `Grid_point_engraver` muss benutzt werden, um die Endpunkte der Linien zu definieren, und der `Grid_line_span_engraver` wird benutzt, um dann die Linien zu setzen. Der Standard ist, dass die Gitterlinien unter den Noten und zur linken Seite des Notenkopfes gesetzt werden. Sie reichen von der Mitte eines Systems bis zur Mitte des anderen. Mit `gridInterval` wird die Dauer zwischen den Linien festgesetzt.

```
\layout {
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #(ly:make-moment 1 4)
  }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
  }
}

\score {
  \new ChoirStaff <<
    \new Staff \relative c'' {
      \stemUp
      c4. d8 e8 f g4
    }
    \new Staff \relative c {
      \clef bass
      \stemDown
      c4 g' f e
    }
  >>
}
```
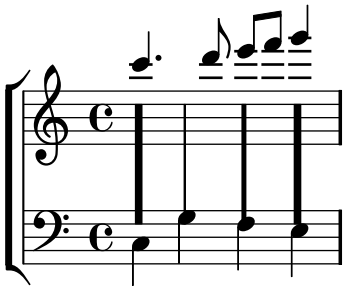


Ausgewählte Schnipsel

Gitternetzlinien: Aussehen verändern

Die Erscheinung der Gitternetzlinien kann durch einige Eigenschaften geändert werden.

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \relative c'' {
        \stemUp
```

```
        c'4. d8 e8 f g4
      }
    }
    \new Staff {
      \relative c {
        % this moves them up one staff space from the default position
        \override Score.GridLine #'extra-offset = #'(0.0 . 1.0)
        \stemDown
        \clef bass
        \once \override Score.GridLine #'thickness = #5.0
        c4
        \once \override Score.GridLine #'thickness = #1.0
        g'4
        \once \override Score.GridLine #'thickness = #3.0
        f4
        \once \override Score.GridLine #'thickness = #5.0
        e4
      }
    }
  >>
  \layout {
    \context {
      \Staff
      % set up grids
      \consists "Grid_point_engraver"
      % set the grid interval to one quarter note
      gridInterval = #(ly:make-moment 1 4)
    }
    \context {
      \Score
      \consists "Grid_line_span_engraver"
      % this moves them to the right half a staff space
      \override NoteColumn #'X-offset = #-0.5
    }
  }
}
```



Siehe auch

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.
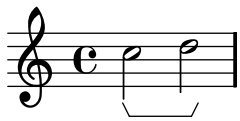
Referenz der Interna: Abschnitt “Grid_line_span_engraver” in *Referenz der Interna*, Abschnitt “Grid_point_engraver” in *Referenz der Interna*, Abschnitt “GridLine” in *Referenz der Interna*,

Abschnitt “GridPoint” in *Referenz der Interna*, Abschnitt “grid-line-interface” in *Referenz der Interna*, Abschnitt “grid-point-interface” in *Referenz der Interna*.

Analyseklammern

Klammern über dem System werden in der Musikanalyse benutzt, um strukturelle Einheiten der Musik zu markieren. Einfache horizontale Klammern werden von LilyPond unterstützt.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' {
  c2\startGroup
  d\stopGroup
}
```



Analyseklammern können verschachtelt sein.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' {
  c4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```
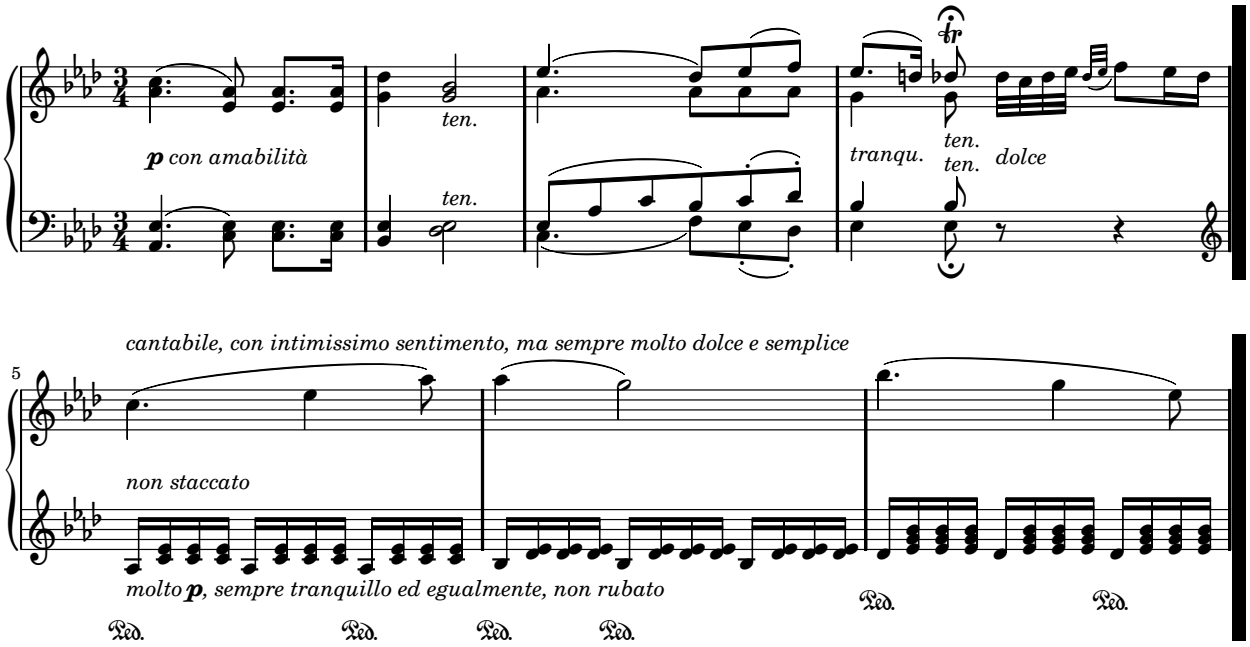


Siehe auch

Schnipsel: Abschnitt “Editorial annotations” in *Schnipsel*.

Referenz der Interna: Abschnitt “Horizontal_bracket_engraver” in *Referenz der Interna*, Abschnitt “HorizontalBracket” in *Referenz der Interna*, Abschnitt “horizontal-bracket-interface” in *Referenz der Interna*, Abschnitt “Staff” in *Referenz der Interna*.

1.8 Text



Dieser Abschnitt erklärt, wie man Text (mit vielfältiger Formatierung) in Partituren einfügt.

Einige Textelemente, die hier nicht behandelt werden, finden sich in anderen Abschnitten: Abschnitt 2.1 [Notation von Gesang], Seite 226, Abschnitt 3.2 [Titel], Seite 433.

1.8.1 Text eingeben
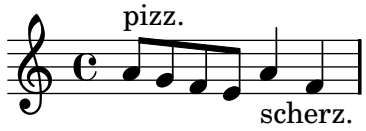
Dieser Abschnitt zeigt verschiedene Arten, wie Text in die Partitur eingefügt werden kann.

Achtung: Wenn man Zeichen mit Akzenten und Umlaute oder besondere Zeichen (wie etwa Text mit anderen Alphabeten) eingeben möchte, kann man die Zeichen einfach direkt in die Datei einfügen. Die Datei muss als UTF-8 gespeichert werden. Für mehr Information siehe Abschnitt 3.3.3 [Zeichenkodierung], Seite 447.

Textarten
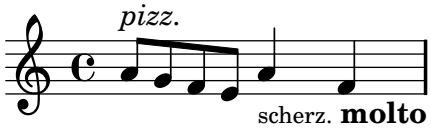
Am einfachsten kann Text mit geraden Anführungsstrichen in eine Partitur eingefügt werden, wie das folgende Beispiel zeigt. Derartiger Text kann manuell über oder unter dem Notensystem platziert werden, die Syntax hierzu ist beschrieben in Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538.

```
a8^"pizz." g f e a4-"scherz." f
```
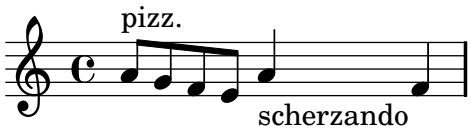


Diese Syntax ist eine Kurzform, komplexere Formatierungen können einem Text hinzugefügt werden, wenn man explizit den `\markup`-Befehl mit darauf folgenden geschweiften Klammern einsetzt, wie beschrieben in Abschnitt 1.8.2 [Text formatieren], Seite 210.

```
a8^\markup { \italic pizz. } g f e
a4_\markup { \tiny scherz. \bold molto } f
```



Standardmäßig haben Textbeschriftungen keinen Einfluss auf die Positionierung der Noten. Man kann aber auch bestimmen, dass die Breite des Textes mit berücksichtigt wird. Im nächsten Beispiel fordert der erste Text keinen Platz, während der zweite die Note nach rechts verschiebt. Das Verhalten wird mit dem Befehl `\textLengthOn` (Textlänge an) erreicht, rückgängig kann es mit dem Befehl `\textLengthOff` gemacht werden.

```
a8^"pizz." g f e
\textLengthOn
a4_"scherzando" f
```



Neben Textbeschriftungen können auch Artikulationen an Noten angehängt werden. Siehe auch [Artikulationszeichen und Verzierungen], Seite 106.

Zu weiterer Information zu der relativen Anordnung von Textbeschriftungen und Artikulationen, siehe Abschnitt “Positionierung von Objekten” in *Handbuch zum Lernen*.

Vordefinierte Befehle

`\textLengthOn`, `\textLengthOff`.

Siehe auch

Handbuch zum Lernen: Abschnitt “Positionierung von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 1.8.2 [Text formatieren], Seite 210, Abschnitt 5.4.2 [Richtung und Platzierung], Seite 538, [Artikulationszeichen und Verzierungen], Seite 106.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

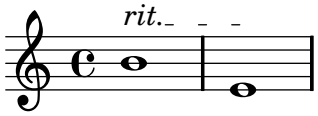
Eine Überprüfung, ob sich auch alle Textbeschriftungen und Gesangstext innerhalb der Ränder der Noten befinden, braucht verhältnismäßig viel Rechenaufwand. Sollten Sie aber etwas bessere Leistung bevorzugen, schreiben Sie in Ihre Datei:

```
\override Score.PaperColumn #'keep-inside-line = ##f
```

Text mit Verbindungslinien

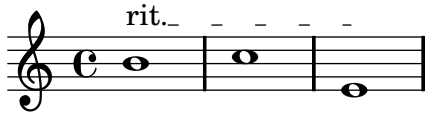
Einige Aufführungsanweisungen, etwa *rallentando* oder *accelerando*, werden als Text geschrieben, gefolgt von einer gestrichelten Linie, die anzeigt, wie weit sich die Anweisung auswirkt. Solche Objekte, „Strecker“ (engl. spanners) genannt, können von einer Note bis zu einer anderen mit folgender Anweisung erstellt werden:

```
\override TextSpanner #'(bound-details left text) = "rit."
b1\startTextSpan
e,\stopTextSpan
```



Der Text wird durch Objekteigenschaften beeinflusst. In den Standardeinstellungen wird er kursiv ausgegeben, aber eine andere Formatierung kann erreicht werden, indem man \markup-Blöcke einsetzt, wie beschrieben in Abschnitt 1.8.2 [Text formatieren], Seite 210.

```
\override TextSpanner #'(bound-details left text) =
  \markup { \upright "rit." }
b1\startTextSpan c
e,\stopTextSpan
```



Auch der Stil der Linie kann ähnlich wie der Text mit den Objekteigenschaften geändert werden. Diese Syntax ist beschrieben in Abschnitt 5.4.8 [Linienstile], Seite 551. Textstrecker sind Teil des `Dynamic`-Kontextes, siehe Abschnitt "Dynamics" in *Referenz der Interna*.

Vordefinierte Befehle

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.
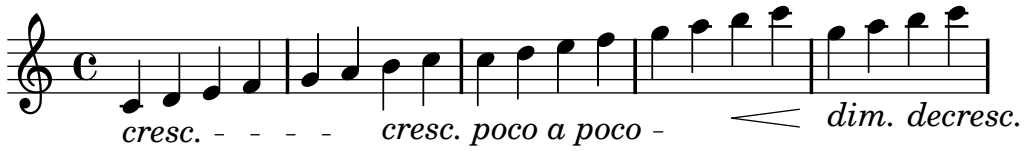
Ausgewählte Schnipsel

Dynamiktextstrecker nachgestellt

Die \cresc, \dim und \decresc Strecker können umdefiniert werden, um nachgestellt zu funktionieren und einen Textstrecker zu produzieren. Eigene Strecker können auch einfach definiert werden. Klammer- und Textcrescendi können einfach vermischt werden. \< und \> erstellen normalerweise Klammern, \cresc usw. dagegen normalerweise Textspanner.

```
% Some sample text dynamic spanners, to be used as postfix operators
crpoco =
#(make-music 'CrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "cresc. poco a poco")

\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decresc c4\!
}
```

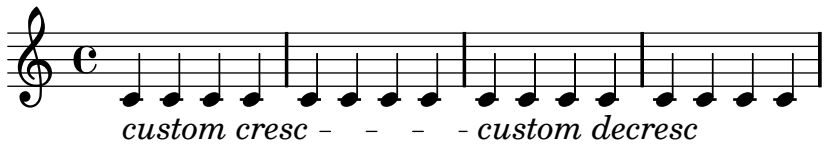


Eigene Dynamiktextspanner nachgestellt

Die Nachstellung funktioniert für eigene Crescendo-Textstrecker. Die Strecker sollten an der ersten Note eines Taktes beginnen. Man muss -\mycresc benutzen, sonst wird der Beginn des Streckers der nächsten Note zugewiesen.

```
% Two functions for (de)crescendo spanners where you can explicitly give the
% spanner text.
mycresc =
#(define-music-function (parser location mymarkup) (markup?)
   (make-music 'CrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))
mydecresc =
#(define-music-function (parser location mymarkup) (markup?)
   (make-music 'DecrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4\! c4 c4
}
```



Siehe auch

Notationsreferenz: Abschnitt 5.4.8 [Linienstile], Seite 551, [Dynamik], Seite 109, Abschnitt 1.8.2 [Text formatieren], Seite 210.
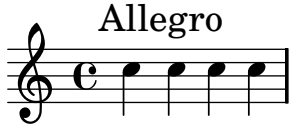
Schnipsel: Abschnitt “Text” in *Schnipsel*, Abschnitt “Expressive marks” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextSpanner” in *Referenz der Interna*

Textartige Zeichen
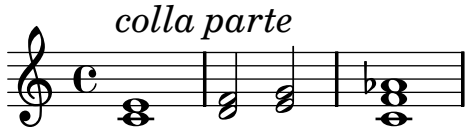
Verschiedene Textelemente können der Partitur hinzugefügt werden, indem man die Syntax für Zeichen einsetzen, wie beschrieben in [Übungszeichen], Seite 97:

```
c4
\mark "Allegro"
c c c
```
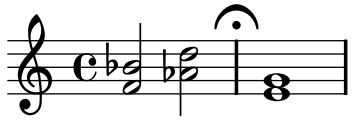


Diese Syntax ermöglicht es, beliebigen Text über eine Taktlinie zu platzieren, weitere Formatierungsmöglichkeiten sind mit dem `\markup`-Befehl gegeben, wie beschrieben in Abschnitt 1.8.2 [Text formatieren], Seite 210:

```
<c e>1
\mark \markup { \italic { colla parte } }
<d f>2 <e g>
<c f aes>1
```



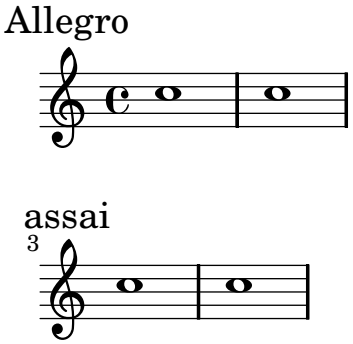
Diese Syntax ermöglicht es auch, besondere Zeichen einzufügen, wie etwa Coda-, Segno- oder Fermatenzeichen, indem das entsprechende Symbol mit dem Befehl `\musicglyph` angegeben wird, wie beschrieben in [Musikalische Notation innerhalb einer Textbeschriftung], Seite 219:

```
<bes f>2 <aes d>
\mark \markup { \musicglyph #"scripts.ufermata" }
<e g>1
```



Derartige Objekte werden über dem höchsten System einer Partitur gesetzt – abhängig davon, ob sie mitten im Takt oder an seinem Ende notiert werden, werden sie zwischen Noten oder über der Taktlinie gesetzt. Wenn sie an einem Zeilenumbruch angegeben werden, wird das Zeichen zu Beginn der nächsten Zeile ausgegeben.

```
\mark "Allegro"
c1 c
\mark "assai" \break
c  c
```
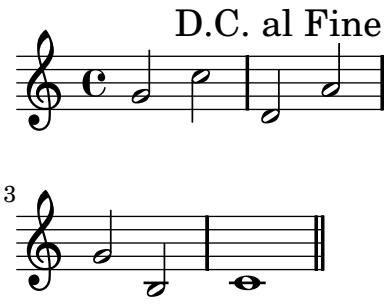


Ausgewählte Schnipsel

Printing marks at the end of a line

Marks can be printed at the end of the current line, instead of the beginning of the following line. In such cases, it might be preferable to align the right end of the mark with the bar line.

```
\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark #'break-visibility = #end-of-line-visible
  \once \override Score.RehearsalMark #'self-alignment-X = #RIGHT
  \mark "D.C. al Fine"
```

```
  \break
  g2 b,
  c1 \bar "||"
}
```



Zeichen an verschiedenen Notationsobjekten ausrichten

Wenn angegeben, können Textzeichen auch an anderen Objekten als Taktstrichen ausgerichtet werden. Zu diesen Objekten gehören `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature` und `time-signature`.

In diesem Fall werden die Zeichen horizontal über dem Objekt zentriert. Diese Ausrichtung kann auch geändert werden, wie die zweite Zeile des Beispiels zeigt. In einer Partitur mit vielen Systemen sollte diese Einstellung für alle Systeme gemacht werden.

```
\relative c' {
  e1

  % the RehearsalMark will be centered above the Clef
  \override Score.RehearsalMark #'break-align-symbols = #'(clef)
  \key a \major
  \clef treble
  \mark ""
  e1

  % the RehearsalMark will be centered above the TimeSignature
  \override Score.RehearsalMark #'break-align-symbols = #'(time-signature)
  \key a \major
  \clef treble
  \time 3/4
  \mark \markup { \char ##x2193 }
  e2.

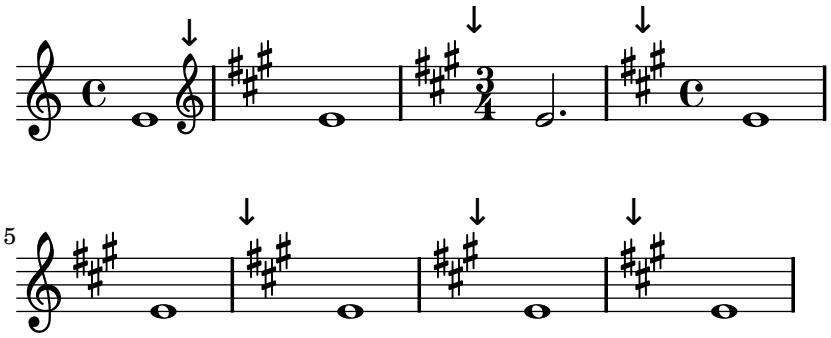
  % the RehearsalMark will be centered above the KeySignature
  \override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
  \key a \major
  \clef treble
  \time 4/4
  \mark \markup { \char ##x2193 }
  e1

  \break
  e1
```

```
  % the RehearsalMark will be aligned with the left edge of the KeySignature
  \once \override Score.KeySignature #'break-align-anchor-alignment = #LEFT
  \mark \markup { \char ##x2193 }
  \key a \major
  e1

  % the RehearsalMark will be aligned with the right edge of the KeySignature
  \once \override Score.KeySignature #'break-align-anchor-alignment = #RIGHT
  \key a \major
  \mark \markup { \char ##x2193 }
  e1

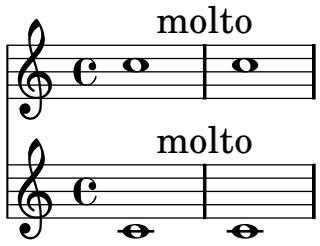
  % the RehearsalMark will be aligned with the left edge of the KeySignature
  % and then shifted right by one unit.
  \once \override Score.KeySignature #'break-align-anchor = #1
  \key a \major
  \mark \markup { \char ##x2193 }
  e1
}
```



Zeichen über jedem System ausgeben

Normalerweise werden Textzeichen nur über dem obersten Notensystem gesetzt. Sie können aber auch über jedem System ausgegeben werden.

```
\score {
  <<
    \new Staff { c''1 \mark "molto" c'' }
    \new Staff { c'1 \mark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove "Mark_engraver"
      \remove "Staff_collecting_engraver"
    }
    \context {
      \Staff
      \consists "Mark_engraver"
      \consists "Staff_collecting_engraver"
    }
  }
}
```



Siehe auch

Notationsreferenz: [Übungszeichen], Seite 97, Abschnitt 1.8.2 [Text formatieren], Seite 210, [Musikalische Notation innerhalb einer Textbeschriftung], Seite 219, Abschnitt A.7 [Die Feta-Schriftart], Seite 578.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “MarkEvent” in *Referenz der Interna*, Abschnitt “Mark_engraver” in *Referenz der Interna*, Abschnitt “RehearsalMark” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Wenn ein Zeichen am Ende des letzten Taktes einer Partitur gesetzt wird (wenn also keine nächste Zeile mehr kommt), wird das Zeichen nicht ausgegeben.

Separater Text

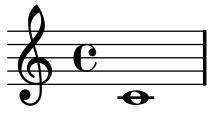
Eine `\markup`-Umgebung kann auch für sich alleine existieren, außerhalb einer `\score`-Umgebung, als ein Ausdruck auf der höchsten Ebene. Diese Syntax ist beschrieben in ⟨undefined⟩ [Dateistruktur], Seite ⟨undefined⟩.

```
\markup {
  Morgen, morgen, und morgen...
}
```

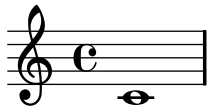
Morgen, morgen, und morgen...

Damit kann Text unabhängig von den Noten gesetzt werden. Das bietet sich vor allem in Situationen an, in denen mehrere Stücke in einer Datei vorkommen, wie beschrieben in Abschnitt 3.1.2 [Mehrere Partituren in einem Buch], Seite 428.

```
\score {
  c'1
}
\markup {
  Morgen, übermorgen, und überübermorgen...
}
\score {
  c'1
}
```



Morgen, übermorgen, und überübermorgen...



Unabhängige Textabschnitte können über mehrere Seiten reichen, so dass man Textdokumente oder Bücher ausschließlich mit LilyPond setzen kann. Einzelheiten zu den vielfältigen Möglichkeiten finden sich in [Textbeschriftung über mehrere Seiten], Seite 222.

Vordefinierte Befehle

`\markup`, `\markuplines`.

Ausgewählte Schnipsel

Isolierter Text in zwei Spalten

Isolierter Text kann in mehreren Spalten mit `\markup`-Befehlen angeordnet werden:

```
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { O sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column {
      \line { \italic { O sacred feast } }
      \line { \italic { in which Christ is received, } }
      \line { \italic { the memory of His Passion is renewed, } }
      \line { \italic { the mind is filled with grace, } }
      \line { \italic { and a pledge of future glory is given to us. } }
      \line { \italic { Amen. } }
    }
    \hspace #1
  }
}
```

O sacrum convivium
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

O sacred feast
in which Christ is received,
the memory of His Passion is renewed,
the mind is filled with grace,
and a pledge of future glory is given to us.
Amen.

Siehe auch

Notationsreferenz: Abschnitt 1.8.2 [Text formatieren], Seite 210, Abschnitt 3.1.5 [Die Dateistruktur], Seite 431, Abschnitt 3.1.2 [Mehrere Partituren in einem Buch], Seite 428, [Textbeschriftung über mehrere Seiten], Seite 222.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

1.8.2 Text formatieren

Dieser Abschnitt zeigt grundlegende und fortgeschrittene Formatierung von Text, wobei der Textbeschriftungsmodus (`\markup` benutzt wird.
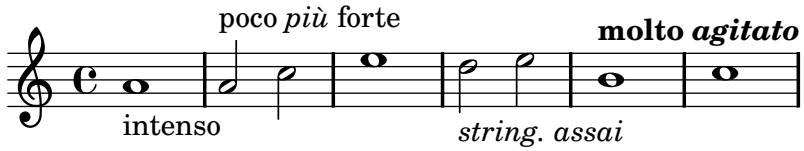
Textbeschriftung (Einleitung)

Eine `\markup`-Umgebung wird benutzt, um Text mit einer großen Anzahl von Formatierungsmöglichkeiten (im „markup-Modus“) zu setzen.

Die Syntax für Textbeschriftungen ähnelt der normalen Syntax von LilyPond: ein `\markup`-Ausdruck wird in geschweifte Klammern eingeschlossen (`{... }`). Ein einzelnes Wort wird als ein Minimalausdruck erachtet und muss deshalb nicht notwendigerweise eingeklammert werden.
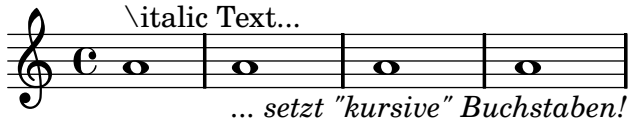
Anders als Text in Anführungsstrichen können sich in einer Textbeschriftungsumgebung (`\markup`) geschachtelte Ausdrücke oder weitere Textbefehle befinden, eingeführt mit einem Backslash (\). Derartige Befehle beziehen sich nur auf den ersten der folgenden Ausdrücke.

```
a1-\markup intenso
a2^\markup { poco \italic più forte  }
c e1
d2_\markup { \italic "string. assai" }
e
b1^\markup { \bold { molto \italic  agitato } }
c
```



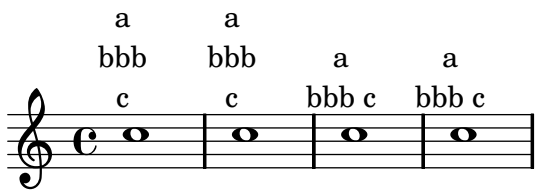
Eine `\markup`-Umgebung kann auch Text in Anführungszeichen beinhalten. Derartige Zeichenketten werden als ein Textausdruck angesehen, und darum werden innerhalb von ihnen Befehle oder Sonderzeichen (wie `\` oder `#`) so ausgegeben, wie sie eingeben werden. Doppelte Anführungsstriche können gesetzt werden, indem man ihnen einen Backslash voranstellt.

```
a1^"\italic Text..."
a_\markup { \italic "... setzt \"kursive\" Buchstaben!" }
a a
```



Damit eine Anzahl von Wörtern als ein einziger Ausdruck behandelt wird, müssen alle Wörter zwischen geraden Anführungszeichen (Shift+2) stehen oder ihnen muss ein Befehl vorangestellt werden. Die Art, wie die Ausdrücke definiert sind, wirkt sich darauf aus, wie sie übereinander gestapelt, mittig und aneinander ausgerichtet werden. Im folgenden Beispiel verhält sich der zweite `\markup`-Ausdruck genauso wie der erste:
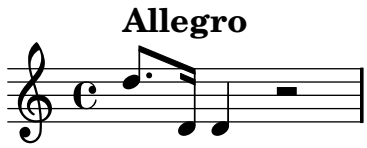
```
c1^\markup { \center-column { a bbb c } }
c1^\markup { \center-column { a { bbb c } } }
c1^\markup { \center-column { a \line { bbb c } } }
c1^\markup { \center-column { a "bbb c" } }
```



Textbeschriftung kann auch durch Variablen definiert werden. Diese Variablen können dann direkt an Noten angefügt werden:

```
allegro = \markup { \bold \large Allegro }

{
  d''8.^\allegro
  d'16 d'4 r2
}
```



Eine ausführliche Liste der `\markup`-Befehle findet sich in Abschnitt A.9 [Textbeschriftungsbefehle], Seite 597.

Siehe auch

Notationsreferenz: Abschnitt A.9 [Textbeschriftungsbefehle], Seite 597.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Installierte Dateien: ‘`scm/markup.scm`’.
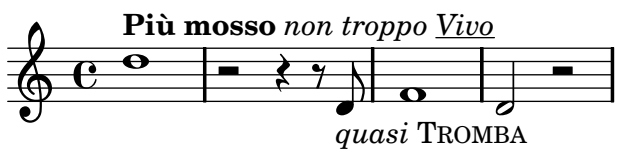
Bekannte Probleme und Warnungen

Syntaxfehler im Textbeschriftungsmodus können sehr verwirrend sein.

Überblick über die wichtigsten Textbeschriftungsbefehle

Einfache Änderungen des Schriftartschnitts können im Textbeschriftungsmodus vorgenommen werden:

```
d1^\markup {
  \bold { Più mosso }
  \italic { non troppo \underline Vivo }
}
r2 r4 r8
d,_\markup { \italic quasi \smallCaps Tromba }
f1 d2 r
```
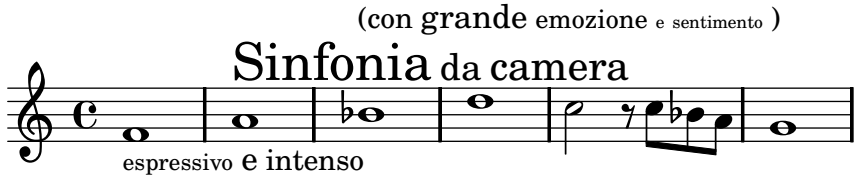


Die Größe von Buchstaben kann auf verschiedene Arten verändert werden:

- die Schriftgröße kann auf bestimmte definierte Standardgrößen gesetzt werden,
- die Schriftgröße kann mit einem absoluten Wert gesetzt werden,
- die Schriftgröße kann relativ zur vorhergehenden Größe geändert werden.

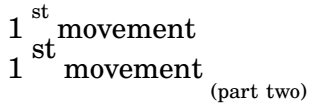
Das Beispiel unten zeigt alle drei Möglichkeiten:

```
f1_\markup {
  \tiny espressivo
  \large e
  \normalsize intenso
}
a^\markup {
  \fontsize #5 Sinfonia
  \fontsize #2 da
  \fontsize #3 camera
}
bes^\markup { (con
  \larger grande
  \smaller emozione
  \magnify #0.6 { e sentimento } )
}
d c2 r8 c bes a g1
```



Text kann auch hoch- bzw. tiefgestellt gesetzt werden. Die so markierten Buchstaben werden automatisch in einer kleineren Schriftgröße gesetzt, aber die normale Schriftgröße kann auch eingesetzt werden:

```
\markup {
  \column {
    \line { 1 \super st movement }
    \line { 1 \normal-size-super st movement
      \sub { (part two) }  }
  }
}
```



Der Textbeschriftungsmodus stellt eine einfache Möglichkeit zur Verfügung unterschiedliche Schriftschnitte anzuwählen. Ohne besondere Einstellungen wird automatisch eine Schriftart mit Serifen ausgewählt. Das Beispiel unten zeigt die Verwendung der eigenen Zahlenschriftart von LilyPond, den Einsatz von serifenloser Schriftart und von Schreibmaschinenschriftart. Die letzte Zeile zeigt, dass sich die Standardeinstellung mit dem Befehl `\roman` wieder herstellen lässt.

```
\markup {
  \column {
    \line { Act \number 1 }
    \line { \sans { Scene I. } }
    \line { \typewriter { Verona. An open place. } }
    \line { Enter \roman Valentine and Proteus. }
  }
}
```

Act I
Scene I.
Verona. An open place.
Enter Valentine and Proteus.

Einige dieser Schriftarten, etwa die Zahlenschriftart oder die Schriftart für Dynamikzeichen, stellen nicht alle Zeichen zur Verfügung, wie beschrieben in [Neue Lautstärkezeichen], Seite 114 und [Manuelle Wiederholungszeichen], Seite 134.

Einige Schriftartbefehle können ungewollte Leerzeichen innerhalb von Wörtern hervorrufen. Das kann vermieden werden, indem die einzelnen Elemente mit dem Befehl `\concat` zu einem Element verschmolzen werden:

```
\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}
```

1st movement
***p**, con dolce espressione*

Eine ausführliche Liste der unterschiedlichen Befehl zur Beeinflussung der Schriftarten findet sich in Abschnitt A.9.1 [Font], Seite 597.

Es ist auch möglich, eigene Schriftfamilien zu definieren, wie erklärt in Abschnitt 1.8.3 [Schriftarten], Seite 223.

Vordefinierte Befehle

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

Siehe auch

Notationsreferenz: Abschnitt A.9.1 [Font], Seite 597, [Neue Lautstärkezeichen], Seite 114, [Manuelle Wiederholungszeichen], Seite 134, Abschnitt 1.8.3 [Schriftarten], Seite 223.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

Installierte Dateien: ‘`scm/define-markup-commands.scm`’.

Bekannte Probleme und Warnungen

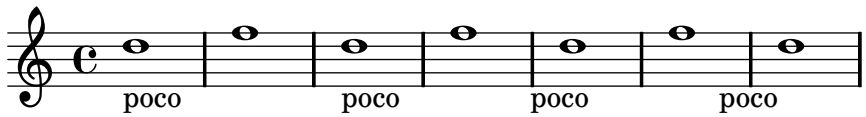
Wenn die Befehle `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large` und `\huge` eingesetzt werden, erhält man schlechte Zeilenabstände verglichen mit `\fontsize`.

Textausrichtung

Dieser Abschnitt zeigt, wie man Text im Textbeschriftungsmodus eingibt. Textobjekte können auch als eine Einheit verschoben werden, wie beschrieben in Abschnitt “Verschieben von Objekten” in *Handbuch zum Lernen*.

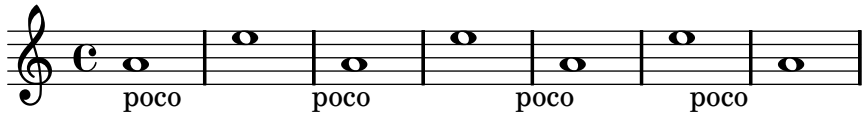
Textbeschriftungsobjekte können auf verschiedene Weise ausgerichtet werden. Standardmäßig wird ein Textobjekt an seiner linken Ecke ausgerichtet, darum wird das erste und zweite Objekt gleichermaßen an der linken Ecke ausgerichtet.

```
d1-\markup { poco }
f
d-\markup { \left-align poco }
f
d-\markup { \center-align { poco } }
f
d-\markup { \right-align poco }
```



Die horizontale Ausrichtung kann mit einer Zahl auf einen exakten Wert festgelegt werden:

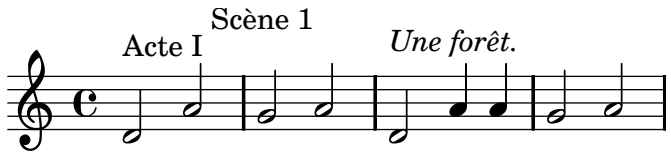
```
a1-\markup { \halign #-1 poco }
e'
a,-\markup { \halign #0 poco }
e'
a,-\markup { \halign #0.5 poco }
e'
a,-\markup { \halign #2 poco }
```



Manche Objekte haben eigene Ausrichtungsvorgänge und werden deshalb nicht von diesen Befehlen beeinflusst. Es ist möglich, solche Objekte als eine Einheit anzusprechen und zu bewegen, wie gezeigt in [Textartige Zeichen], Seite 204.

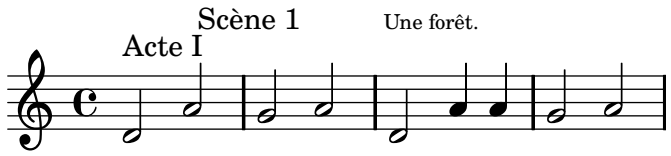
Die vertikale Ausrichtung ist etwas schwieriger. Textelemente können komplett verschoben werden, es ist aber auch möglich, nur einen Teil innerhalb der Textbeschriftung zu bewegen. In diesem Fall muss dem zu verschiebenden Objekt ein Ankerpunkt zugewiesen werden, welcher entweder ein anderes Textelement oder ein unsichtbares Objekt sein kann (im Beispiel mit `\null` erstellt). Der letzte Text im Beispiel hat keinen Anker und wird deshalb auch nicht verschoben.

```
d2^\markup {
  Acte I
  \raise #2 { Scène 1 }
}
a'
g_\markup {
  \null
  \lower #4 \bold { Très modéré }
}
a
d,^\markup {
  \raise #4 \italic { Une forêt. }
}
a'4 a g2 a
```



Einige Befehle können sowohl die horizontale als auch die vertikale Ausrichtung von Textobjekten beeinflussen. Jedes Objekt, das auf diese Weise verschoben wird, benötigt einen Anker:

```
d2^\markup {
  Acte I
  \translate #'(-1 . 2) "Scène 1"
}
a'
g_\markup {
  \null
  \general-align #Y #3.2 \bold "Très modéré"
}
a
d,^\markup {
  \null
  \translate-scaled #'(-1 . 2) \teeny "Une forêt."
}
a'4 a g2 a
```



Ein Textbeschriftungsobjekt kann mehrere Zeilen beinhalten. Im folgenden Beispiel wird jeder Ausdruck innerhalb von `\markup` auf einer eigenen Zeile gesetzt, entweder linksbündig oder zentriert:

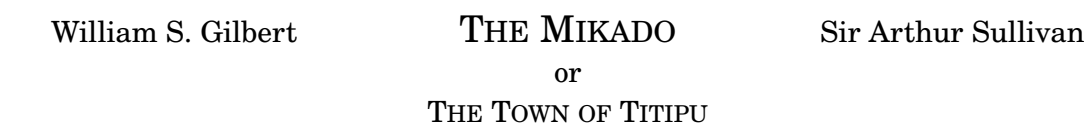
```
\markup {
  \column {
    a
    "b c"
    \line { d e f }
  }
  \hspace #10
  \center-column {
    a
    "b c"
    \line { d e f }
  }
}
```



Eine Anzahl an Ausdrücken innerhalb von `\markup` kann auch gestreckt werden, so dass die gesamte Seitenbreite benutzt wird. Wenn nur ein Objekt vorhanden ist, wird es zentriert gesetzt.

Die Ausdrücke selber können wiederum mehrzeilig sein und andere Textbeschriftungsbefehle beinhalten.

```
\markup {
  \fill-line {
    \line { William S. Gilbert }
    \center-column {
      \huge \smallCaps "The Mikado"
      or
      \smallCaps "The Town of Titipu"
    }
    \line { Sir Arthur Sullivan }
  }
}
\markup {
  \fill-line { 1885 }
}
```



Längere Texte können auch automatisch umgebrochen werden, wobei es möglich ist, die Zeilenbreite zu bestimmen. Der Text ist entweder linksbündig oder im Blocksatz, wie das nächste Beispiel illustriert:

```
\markup {
  \column {
    \line  \smallCaps { La vida breve }
    \line \bold { Acto I }
    \wordwrap \italic {
      (La escena representa el corral de una casa de
      gitanos en el Albaicín de Granada.  Al fondo una
      puerta por la que se ve el negro interior de
      una Fragua, iluminado por los rojos resplandores
      del fuego.)
    }
    \hspace #0

    \line \bold { Acto II }
    \override #'(line-width . 50)
    \justify \italic {
      (Calle de Granada.  Fachada de la casa de Carmela
      y su hermano Manuel con grandes ventanas abiertas
      a través de las que se ve el patio
      donde se celebra una alegre fiesta)
    }
  }
}
```

La vida breve

Acto I

(La escena representa el corral de una casa de gitanos en el Albaicín de Granada. Al fondo una puerta por la que se ve el negro interior de una Fragua, iluminado por los rojos resplandores del fuego.)

Acto II

(Calle de Granada. Fachada de la casa de Carmela y su hermano Manuel con grandes ventanas abiertas a través de las que se ve el patio donde se celebra una alegre fiesta)

Eine vollständige Liste der Textausrichtungsbefehle findet sich in Abschnitt A.9.2 [Align], Seite 606.

Siehe auch

Handbuch zum Lernen: Abschnitt “Verschieben von Objekten” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt A.9.2 [Align], Seite 606, [Textartige Zeichen], Seite 204.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

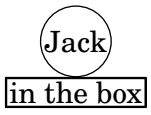
Installierte Dateien: ‘`scm/define-markup-commands.scm`’.

Graphische Notation innerhalb einer Textbeschriftung

Verschiedene graphische Objekte können im Textbeschriftungsmodus eingefügt werden.

Mit bestimmten Textbeschriftungsbefehlen kann man Textelementen Graphik hinzufügen, wie das nächste Beispiel zeigt:

```
\markup \fill-line {
  \center-column {
    \circle Jack
    \box "in the box"
    \null
    \line {
      Erik Satie
      \hspace #3
      \bracket "1866 - 1925"
    }
    \null
    \rounded-box \bold Prelude
  }
}
```

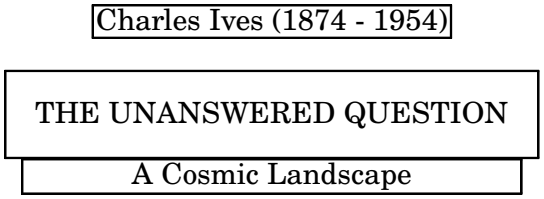


Erik Satie [1866 - 1925]

Prelude

Es kann nötig sein, einem Text mehr Platz einzuräumen. Das geschieht mit verschiedenen Befehlen, wie das folgende Beispiel zeigt. Eine ausführliche Übersicht findet sich in Abschnitt A.9.2 [Align], Seite 606.

```
\markup \fill-line {
  \center-column {
    \box "Charles Ives (1874 - 1954)"
    \null
    \box \pad-markup #2 "THE UNANSWERED QUESTION"
    \box \pad-x #8 "A Cosmic Landscape"
    \null
  }
}
\markup \column {
  \line {
    \hspace #10
    \box \pad-to-box #'(-5 . 20) #'(0 . 5)
      \bold "Largo to Presto"
  }
  \pad-around #3
      "String quartet keeps very even time,
Flute quartet keeps very uneven time."
}
```



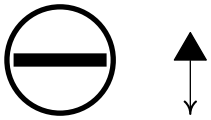


String quartet keeps very even time, Flute quartet keeps very uneven time.

Andere graphische Elemente oder Symbole können gesetzt werden, ohne dass man Text benötigt. Wie mit allen Textbeschriftungen können Objekte innerhalb von `\markup` kombiniert werden.

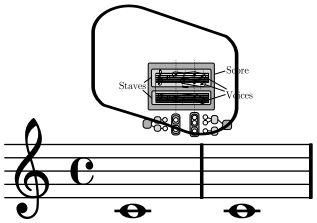
```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Fortgeschrittene graphische Möglichkeiten bietet unter Anderem eine Funktion, mit der man externe Graphiken im Encapsulated PostScript (*eps*) -Format einbinden kann oder aber Graphiken direkt in den Quelltext unter Verwendung von PostScript-Code notiert. In diesem Fall kann es nötig sein, die Größe der Zeichnung explizit anzugeben, wie im Beispiel unten gezeigt:

```
c1^\markup {
  \combine
    \epsfile #X #10 #"./context-example.eps"
    \with-dimensions #'(0 . 6) #'(0 . 10)
    \postscript #"
      -2 3 translate
      2.7 2 scale
      newpath
      2 -1 moveto
      4 -2 4 1 1 arct
      4 2 3 3 1 arct
      0 4 0 3 1 arct
      0 0 1 -1 1 arct
      closepath
      stroke"
  }
c
```



Eine ausführliche Liste der Graphik-Befehle findet sich in Abschnitt A.9.3 [Graphic], Seite 620.

Siehe auch

Notationsreferenz: Abschnitt A.9.3 [Graphic], Seite 620, Abschnitt 1.7 [Anmerkungen], Seite 189, Abschnitt A.9.2 [Align], Seite 606.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

Installierte Dateien: ‘`scm/define-markup-commands.scm`’, ‘`scm/stencil.scm`’.
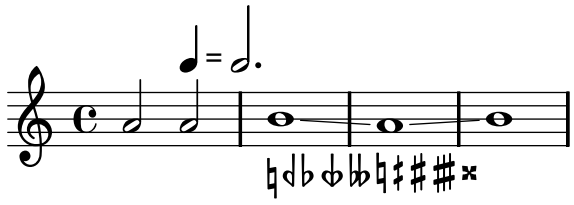
Musikalische Notation innerhalb einer Textbeschriftung

Auch Musikobjekte können innerhalb der Textbeschriftungsumgebung gesetzt werden.

Noten und Versetzungszeichen lassen sich mit `\markup` einfügen:
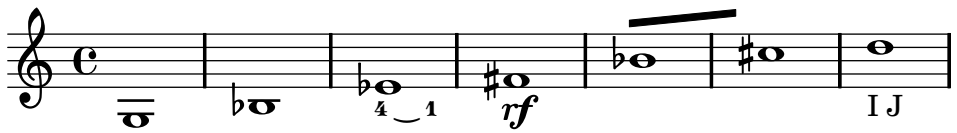
```
a2 a^\markup {
  \note #"4" #1
  =
  \note-by-number #1 #1 #1.5
}
b1_\markup {
```

```
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a1_\markup {
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b
```



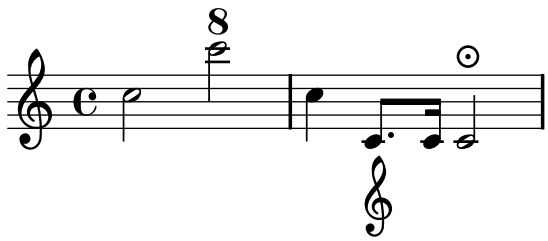
Andere Notationsobjekte können auch eingefügt werden:

```
g1 bes
ees-\markup {
  \finger 4
  \tied-lyric #"~"
  \finger 1
}
fis_\markup { \dynamic rf }
bes^\markup {
  \beam #8 #0.1 #0.5
}
cis
d-\markup {
  \markalphabet #8
  \markletter #8
}
```



Allgemeiner gesagt kann jedes verfügbare Notationssymbol unabhängig von der Notation als ein Textbeschriftungsobjekt eingefügt werden, wie unten gezeigt. Eine vollständige Liste der verfügbaren Symbole findet sich in Abschnitt A.7 [Die Feta-Schriftart], Seite 578.

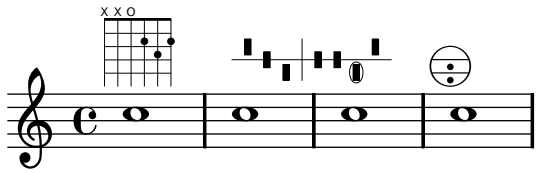
```
c2
c'^\markup { \musicglyph #"eight" }
c,4
c,8._\markup { \musicglyph #"clefs.G_change" }
c16
c2^\markup { \musicglyph #"timesig.neomensural94" }
```



Eine andere Möglichkeit, andere als Textsymbole zu schreiben, findet sich in [Was sind Schriftarten], Seite 223. Diese Methode bietet sich an, um Klammern unterschiedlicher Größe zu setzen.

Der Textbeschriftungsmodus unterstützt auch Diagramme für bestimmte Instrumente:
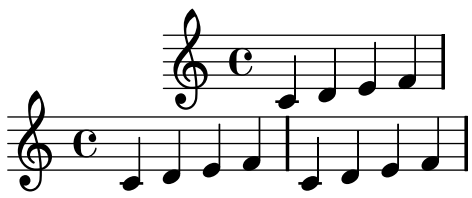
```
c1^\markup {
  \fret-diagram-terse #"x;x;o;2;3;2;"
}
c^\markup {
  \harp-pedal #"^-v|--ov^"
}
c
c^\markup {
  \combine
    \musicglyph #"accordion.discant"
    \combine
      \raise #0.5 \musicglyph #"accordion.dot"
      \raise #1.5 \musicglyph #"accordion.dot"
}
```



Derartige Digramme sind dokumentiert in Abschnitt A.9.5 [Instrument Specific Markup], Seite 630.

Sogar eine ganze Partitur kann in ein Textbeschriftungsobjekt eingefügt werden. In diesem Fall muss die eingefügte `\score`-Umgebung eine `\layout`-Umgebung haben, wie in diesem Beispiel:

```
c4 d^\markup {
  \score {
    \relative c' { c4 d e f }
    \layout { }
  }
}
e f |
c d e f
```



Eine vollständige Liste der Musiksymbol-Befehle findet sich in Abschnitt A.9.4 [Music], Seite 626.

Siehe auch

Notationsreferenz: Abschnitt A.9.4 [Music], Seite 626, Abschnitt A.7 [Die Feta-Schriftart], Seite 578, [Was sind Schriftarten], Seite 223.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

Installierte Dateien: ‘`scm/define-markup-commands.scm`’, ‘`scm/fret-diagrams.scm`’, ‘`scm/harp-pedals.scm`’.

Textbeschriftung über mehrere Seiten

Normale Textbeschriftungsobjekte können nicht getrennt werden, aber mit einer spezifischen Umgebung ist es möglich, Text auch über mehrere Seiten fließen zu lassen:

```
\markuplines {
  \justified-lines {
    A very long text of justified lines.
    ...
  }
  \wordwrap-lines {
    Another very long paragraph.
    ...
  }
  ...
}
```

A very long text of justified lines. ...

Another very long paragraph. ...

...

Die Syntax braucht eine Liste von Textbeschriftungen folgender Art:

- das Resultat eines Beschriftungslistenbefehls,
- eine Textbeschriftungsliste,
- eine Liste von Beschriftungslisten.

Eine vollständige Liste der Beschriftungslistenbefehle findet sich in Abschnitt A.10 [Textbeschriftungslistenbefehle], Seite 639.

Siehe auch

Notationsreferenz: Abschnitt A.10 [Textbeschriftungslistenbefehle], Seite 639.

Schnipsel: Abschnitt “Text” in *Schnipsel*.

Erweitern: Abschnitt “Neue Definitionen von Beschriftungslistenbefehlen” in *Extending*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*.

Installierte Dateien: ‘`scm/define-markup-commands.scm`’.

Vordefinierte Befehle

`\markuplines`.

1.8.3 Schriftarten

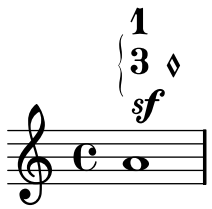
Dieser Abschnitt zeigt, wie Schriftarten eingesetzt werden können und wie man sie in Partituren ändern kann.

Was sind Schriftarten

Schriftarten werden von mehreren Bibliotheken verwaltet. FontConfig wird benützt, um die vorhandenen Schriftarten des Systems zu erkennen, die gewählte Schriftart wird dann mit Pango verarbeitet.

Notationsschriftarten können als eine Ansammlung von besonderen Zeichen erklärt werden, wobei die Sonderzeichen in verschiedene Familien klassifiziert werden. Die Syntax des folgenden Beispiels ermöglicht es, direkt auf verschiedene nicht textuelle Sonderzeichen der `feta`-Schriftart zuzugreifen. Das ist die Standardschriftart für Notationselemente in LilyPond.

```
a1^\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup #"brace120"
    \override #'(font-encoding . fetaText)
    \column { 1 3 sf }
    \override #'(font-encoding . fetaMusic)
    \lookup #"noteheads.s0petrucci"
  }
}
```



Außer den verschiedenen Klammern, die in `fetaBraces` in verschiedenen Größen enthalten sind, lassen sich alle dieses Symbole auch mit einer einfacheren Syntax notieren. Sie ist beschrieben in [Musikalische Notation innerhalb einer Textbeschriftung], Seite 219.
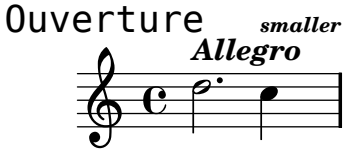
Wenn man die Klammern von `fetaBraces` benutzt, wird die Größe der Klammer durch einen numeralen Part in der Bezeichnung des Glyphs bestimmt. Als Wert kann eine Ganzzahl von `0` bis `575` benutzt werden, wobei `0` die kleinste Klammern ergibt. Der optimale Wert muss durch Ausprobieren herausgefunden werden. Diese Glyphen sind alle linke Klammern, rechte Klammern lassen sich durch eine Drehung herstellen, siehe [Drehen von Objekten], Seite 552.

Drei Textschriftarten sind verfügbar (auf Englisch `family` genannt): mit `roman` eine Schriftart mit Serifen (Standard ist New Century Schoolbook), mit `sans` eine serifenlose (gerade) Schriftart und mit `typewriter` eine Schreibmaschinenschrift, in welcher die Buchstaben alle die gleiche Weite haben. Die aktuelle Schriftart von `sans` und `typewriter` wird durch Pango entsprechend den Systemvorgaben gewählt.

Jede Familie kann verschiedene Schriftschnitte besitzen. Im Englischen wird unterschieden zwischen `shape` für kursive Schnitte und `series` für fette Schnitte. Im folgenden Beispiel wird demonstriert, wie man die verschiedenen Eigenschaften auswählen kann. Der Wert, der `font-size` übergeben wird, entspricht der geforderten Änderung in Bezug auf die Standardschriftgröße.

```
\override Score.RehearsalMark #'font-family = #'typewriter
\mark \markup "Ouverture"
```

```
\override Voice.TextScript #'font-shape = #'italic
\override Voice.TextScript #'font-series = #'bold
d2.^\markup "Allegro"
\override Voice.TextScript #'font-size = #-3
c4^smaller
```



Eine ähnliche Syntax kann im Textbeschriftungsmodus eingesetzt werden, hier bietet es sich aber an, die einfacheren Befehle zu verwenden, die erklärt wurden in [Überblick über die wichtigsten Textbeschriftungsbefehle], Seite 211:

```
\markup {
  \column {
    \line {
      \override #'(font-shape . italic)
      \override #'(font-size . 4)
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter)
      {
        \override #'(font-series . bold)
        re
        di
      }
      \override #'(font-family . sans)
      Creta
    }
  }
}
```



Auch wenn es einfach ist, zwischen den vordefinierten Schriftarten umzuschalten, kann man auch eigene Schriftarten verwenden, wie erklärt in folgenden Abschnitten: [Schriftarten für einen Eintrag], Seite 224 und [Schriftart des gesamten Dokuments], Seite 225.

Siehe auch

Notationsreferenz: Abschnitt A.7 [Die Feta-Schriftart], Seite 578, [Drehen von Objekten], Seite 552, [Musikalische Notation innerhalb einer Textbeschriftung], Seite 219, [Überblick über die wichtigsten Textbeschriftungsbefehle], Seite 211, Abschnitt A.9.1 [Font], Seite 597.
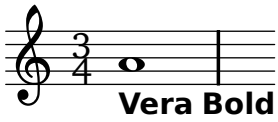
Schriftarten für einen Eintrag

Jede Schriftart, die über das Betriebssystem installiert ist und von FontConfig erkannt wird, kann in einer Partitur eingefügt werden. Dazu verwendet man folgende Syntax:

```
\override Staff.TimeSignature #'font-name = #"Bitstream Charter"
\override Staff.TimeSignature #'font-size = #2
```

```
\time 3/4

a1_\markup {
  \override #'(font-name . "Vera Bold")
    { Vera Bold }
}
```



Mit folgendem Befehl erhält man eine Liste aller verfügbaren Schriftarten des Betriebssystems:

```
lilypond -dshow-available-fonts x
```

Siehe auch

Notationsreferenz: [Was sind Schriftarten], Seite 223, [Schriftart des gesamten Dokuments], Seite 225.

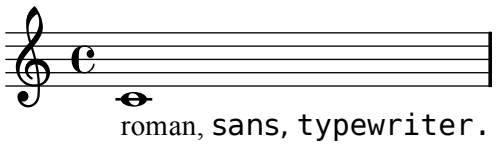
Schnipsel: Abschnitt “Text” in *Schnipsel*.

Schriftart des gesamten Dokuments

Es ist auch möglich, die Schriftarten für die gesamte Partitur zu ändern. In diesem Fall müssen die Familien `roman`, `sans` und `typewriter` in genaus dieser Reihenfolge entsprechend der Syntax unten definiert werden. Einzelheiten zu Schriftarten in [Was sind Schriftarten], Seite 223.

```
\paper  {
  myStaffSize = #20
  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans"
                          "Luxi Mono"
                           (/ myStaffSize 20)))
}

\relative c'{
  c1-\markup {
    roman,
    \sans sans,
    \typewriter typewriter. }
}
```
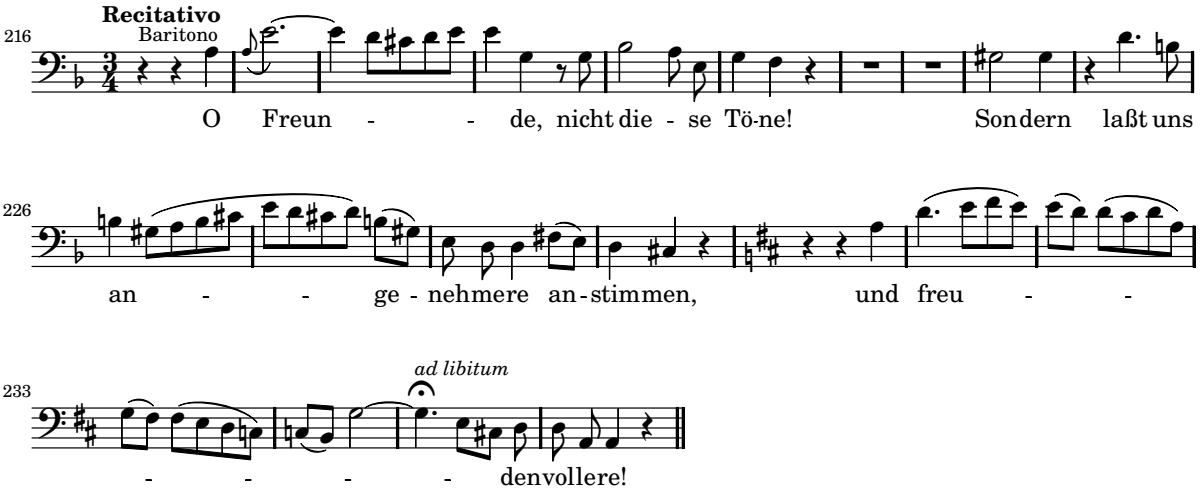


Siehe auch

Notationsreferenz: [Was sind Schriftarten], Seite 223, [Schriftarten für einen Eintrag], Seite 224, [Überblick über die wichtigsten Textbeschriftungsbefehle], Seite 211, Abschnitt A.9.1 [Font], Seite 597.

2 Spezielle Notation

Dieser Abschnitt erklärt, wie Notation erstellt wird, die nur für ein bestimmtes Instrument oder einen Stil eingesetzt wird.

2.1 Notation von Gesang



Dieser Abschnitt erklärt, wie Vokalmusik gesetzt werden kann und die Silben von Gesangstext an den Noten ausgerichtet werden.

2.1.1 Übliche Notation für Vokalmusik

Dieser Abschnitt erklärt Eigenheiten und Probleme, die die meisten Arten an Vokalmusik gemeinsam haben.

Referenz für Vokalmusik

Dieser Abschnitt, wo man Lösungen zu den Problemen finden kann, die bei der Notation von Gesang mit Text auftreten können.

- Die meisten Vokalmusikstile benutzen Text für den Gesangstext. Eine Einleitung hierzu findet sich in Abschnitt “Einfache Lieder setzen” in *Handbuch zum Lernen*.
- Vokalmusik braucht oft die Benutzung von Textbeschriftung (dem `markup`-Modus) für den Gesangstext oder andere Textelemente (Namen von Figuren usw.). Die entsprechende Syntax ist beschrieben in [Textbeschriftung (Einleitung)], Seite 210.
- ‚Ambitus‘ können zu Beginn der Stimmen hinzugefügt werden, dies findet sich erklärt in [Tonumfang], Seite 31.
- Dynamikbezeichnung werden normalerweise unter das Notensystem platziert, aber in Chormusik werden sie normalerweise über das Notensystem notiert, um Platz für den Text zu schaffen. Siehe [Partiturbeispiele für Chormusik], Seite 264.

Siehe auch

Glossar: Abschnitt “ambitus” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Einfache Lieder setzen” in *Handbuch zum Lernen*.

Notationsreferenz: [Textbeschriftung (Einleitung)], Seite 210, [Tonumfang], Seite 31, [Partiturbeispiele für Chormusik], Seite 264.

Snippets: Abschnitt “Vocal music” in *Schnipsel*.

Eingabe von Text

Gesangstext muss in einem speziellen Modus notiert werden. Der Gesangstextmodus kann mit der Umgebung `\lyricmode` angegeben werden, oder indem `\addlyrics` bzw. `\lyricsto` eingesetzt wird. In diesem Modus kann Text mit Akzenten und Satzzeichen notiert werden, und das Programm liest `d` nicht als die Tonhöhe *D*, sondern als eine Silbe Text. Anders gesagt: Silben werden wie Noten notiert, aber die Tonhöhen werden durch Text ersetzt.

Beispielsweise:

```
\lyricmode { Gern4 hätt'4 ich4 dich4 lieb!2 }
```

Es gibt zwei generelle Methoden, die horizontale Orientierung der Textsilben anzugeben, entweder indem ihre Dauer angegeben wird, wie oben in dem Beispiel, oder indem die Silben automatisch an den Noten ausgerichtet werden. Dazu muss entweder `\addlyrics` oder `\lyricsto` eingesetzt werden. Die erste Methode ist beschrieben in ⟨undefined⟩ [Manuelle Silbendauern], Seite ⟨undefined⟩, die zweite in [Automatische Silbendauern], Seite 230.

Ein Wort oder eine Silbe beginnt mit einem alphabetischen Zeichen (inklusive einige andere Zeichen, siehe unten) und endet mit einem Leerzeichen oder einer Zahl. Die folgenden Zeichen in der Silbe können beliebig sein, außer Leerzeichen und Zahlen.

Jedes Zeichen, das nicht Leerzeichen noch Zahl ist, wird als Bestandteil der Silbe angesehen. Eine Silbe kann also auch mit `}` enden, was oft zu dem Fehler

```
\lyricmode { lah- lah}
```

führen kann. Hier wird `}` als Teil der letzten Silbe gerechnet, so dass die öffnende Klammer keine schließende Klammer hat und die Eingabedatei nicht funktioniert. Klammern sollten deshalb immer von Leerzeichen umgeben sein.

```
\lyricmode { lah lah lah }
```

Auch ein Punkt, der auf eine Silbe folgt, wird in die Silbe inkorporiert. Infolgedessen müssen auch um Eigenschaftsbezeichnungen Leerzeichen gesetzt werden. Ein Befehl heißt also *nicht*:
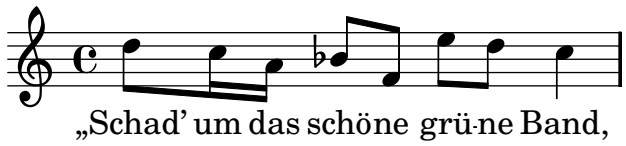
```
\override Score.LyricText #'font-shape = #'italic
```

sondern

```
\override Score . LyricText #'font-shape = #'italic
```
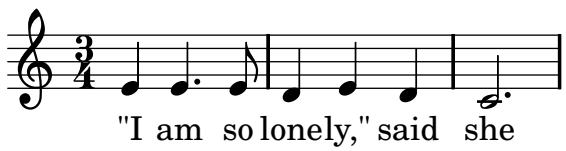
Punkte, Gesangstext mit Akzenten, verschiedene lateinische und nicht-lateinische Zeichen sowie auch etwa Sonderzeichen (wie ein Herz-Symbol) könn direkt in die Notationsdatei geschrieben werden. Es muss dabei sichergestell werden, dass die Datei in der UTF-8-Kodierung gespeichert wird. Zu mehr Information siehe ⟨undefined⟩ [Zeichnkodierung], Seite ⟨undefined⟩.

```
\relative c'' { d8 c16 a bes8 f e' d c4 }
\addlyrics { „Schad' um das schö -- ne grü -- ne Band, }
```



Normale Anführungszeichen können im Gesangstext auch benutzt werden, aber sie müssen mit einem Backslash und weiteren Anführungszeichen begleitet werden:

```
\relative c' { \time 3/4 e4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone -- "ly,\"" said she }
```



Die vollständige Definition des Anfangs eines Wortes in LilyPond ist etwas komplizierter. Ein Wort im Gesangstextmodus beginnt mit einem alphabetischen Zeichen, `_`, `?`, `!`, `:`, `'`, den Kontrollzeichen `^A` bis `^F`, `^Q` bis `^W`, `^Y`, `^^`, einem beliebigen 8-bit-Zeichen mit einem ASCII-Code größer als 127 oder einer zwei-Zeichen-Kombination aus einem Backslash, gefolgt von `` ` ``, `'`, `"` oder `^`.

Man hat sehr gut Möglichkeiten, die Erscheinung des Gesangstextes zu beeinflussen, wenn man dafür Textbeschriftungsbefehle einsetzt. Siehe hierzu Abschnitt 1.8.2 [Text formatieren], Seite 210.
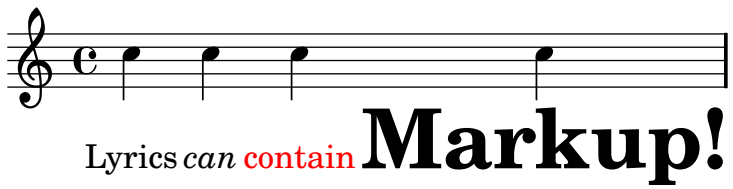
Ausgewählte Schnipsel

Formatting lyrics syllables

Markup mode may be used to format individual syllables in lyrics.

```
mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>
```



Siehe auch

Handbuch zum Lernen: Abschnitt “Lieder” in *Handbuch zum Lernen*.

Notationsreferenz: [Automatische Silbendauern], Seite 230, Abschnitt 1.8.3 [Schriftarten], Seite 223, Abschnitt 5.4.1 [Eingabe-Modi], Seite 536, ⟨undefined⟩ [Manuelle Silbendauern], Seite ⟨undefined⟩, Abschnitt 1.8.2 [Text formatieren], Seite 210, Abschnitt 3.3.3 [Zeichenkodierung], Seite 447.

Referenz der Interna: Abschnitt “LyricText” in *Referenz der Interna*.

Schnipsel: Abschnitt “Text” in *Schnipsel*

Text an einer Melodie ausrichten

Gesangstext wird gesetzt, wenn er in einem `Lyrics`-Kontext ausgewertet wird, siehe Abschnitt 5.1.1 [Was sind Kontexte?], Seite 513.

```
\new Lyrics \lyricmode { ... }
```

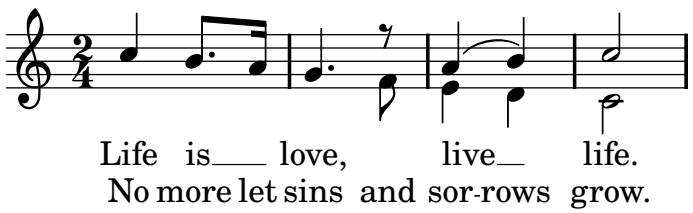
Gesangstext kann an einer Melodie auf zwei Arten ausgerichtet werden:

- Gesangstext kann automatisch ausgerichtet werden, wobei die Dauern von einer Notenstimme oder (in besonderen Umständen) von einer assoziierten Melodie übernommen werden. Das geschieht mit `\addlyrics`, `\lyricsto` oder indem man die `associatedVoice`-Eigenschaft einsetzt. Mehr Informationen in [Automatische Silbendauern], Seite 230.

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative c'' {
      \voiceOne
      c4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative c' {
      \voiceTwo
       s2 s4. f8 e4 d c2
    }
  >>

% takes durations and alignment from notes in "one"
  \new Lyrics \lyricsto "one" {
    Life is __ _ love, live __ life.
  }

% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



Die erste Strophe zeigt, wie normalerweise Text gesetzt wird.

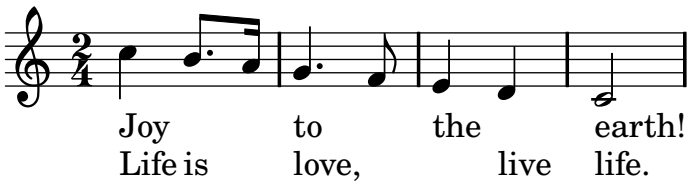
Die zweite Strophe zeigt, wie die Stimme (`Voice`), von der die Dauern entnommen werden, geändert werden kann. Das ist nützlich, wenn der Text unterschiedlicher Strophen die Noten auf unterschiedliche Weise füllt und alle Dauern als `Voice`-Kontexte vorhanden sind. Zu mehr Details siehe Abschnitt 2.1.3 [Strophen], Seite 255.

- Gesangstext kann unabhängig von den Notendauern platziert werden, indem man die Dauern der Silben explizit vorgibt und den Text innerhalb von `\lyricmode` notiert:

```
<<
  \new Voice = "one" \relative c'' {
    \time 2/4
    c4 b8. a16 g4. f8 e4 d c2
  }

% uses previous explicit duration of 2;
  \new Lyrics \lyricmode {
    Joy to the earth!
  }
```

```
% explicit durations, set to a different rhythm
  \new Lyrics \lyricmode {
    Life4 is love,2. live4 life.2
  }
>>
```



Die erste Strophe wird nicht an den Noten ausgerichtet, weil die Dauern nicht angegeben sind, und der erste Wert 2 für alle Silben benutzt wird.

Die zweite Strophe zeigt, dass die Silben sehr unabhängig von den Noten notiert werden können. Das ist nützlich, wenn der Text von verschiedenen Strophen die Noten auf unterschiedliche Weise füllt, aber die Dauer nicht in einem Noten-Kontext vorhanden ist. Mehr Details finden sich in ⟨undefined⟩ [Manuelle Silbendauern], Seite ⟨undefined⟩. Diese Technik ist auch nützlich, wenn man Dialog zu einer Musik setzt; ein Beispiel hierzu findet sich in [Dialog zur Musik], Seite 273.

Wenn Text auf diese Weise eingegeben wird, werden die Silben links an den Noten ausgerichtet, können aber auch zentriert werden, indem man eine assoziierte Stimme angibt, wenn eine existiert. Zu Einzelheiten siehe ⟨undefined⟩ [Manuelle Silbendauern], Seite ⟨undefined⟩.

Siehe auch

Handbuch zum Lernen: Abschnitt “Text an einer Melodie ausrichten” in *Handbuch zum Lernen*.

Notation Reference: Abschnitt 5.1.1 [Was sind Kontexte?], Seite 513, [Automatische Silbendauern], Seite 230. Abschnitt 2.1.3 [Strophen], Seite 255, ⟨undefined⟩ [Manuelle Silbendauern], Seite ⟨undefined⟩, [Dialog zur Musik], Seite 273.

Referenz der Interna: Abschnitt “Lyrics” in *Referenz der Interna*.

Automatische Silbendauern

Die Silben des Gesangstextes können automatisch an einer Melodie ausgerichtet werden. Das ist auf drei Arten möglich:

- indem man einen benannten `Voice`-Kontext mit der Melodie durch `\lyricsto` zuweist,
- indem man den Gesangstext mit `\addlyrics` beginnt und direkt nach dem `Voice`-Kontext mit der Melodie notiert,
- indem man die `associatedVoice`-Eigenschaft definiert, sodass man die Ausrichtung des Gesangstextes zwischen verschiedenen benannten `Voice`-Kontexten gewechselt werden.

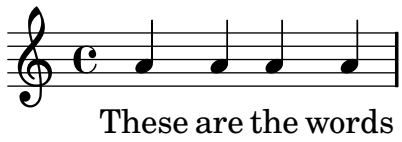
In allen drei Methoden können Bindestriche zwischen den Silben oder Fülllinien hinter einem Wortende gezogen werden. Zu Einzelheiten siehe [Fülllinien und Trennstriche], Seite 238.

Der `Voice`-Kontext, der die Melodie enthält, an der der Text ausgerichtet werden soll, darf noch nicht „gestorben“ sein, weil sonst aller Text danach verloren geht. Das kann passieren, wenn es Strecken gibt, in denen die Stimme pausiert. Zu Methoden, wie man Kontexte am Leben erhält, siehe Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516.

`\lyricsto` Benutzen

Gesangstext kann an einer Melodie automatisch ausgerichtet werdne, indem man den beannten `Voice`-Kontext mit der Melodie durch den Befehl `\lyricsto` angibt:

```
<<
  \new Voice = "melody" {
    a4 a a a
  }
  \new Lyrics \lyricsto "melody" {
    These are the words
  }
>>
```



Damit wird der Text an den Noten des benannten `Voice`-Kontextes ausgerichtet, der schon vorher existieren muss. Aus diesem Grund wird der `Voice`-Kontext normalerweise zuerst definiert, gefolgt vom `Lyrics`-Kontext. Der Gesangstext selber folgt dem `\lyricsto`-Befehl. Der `\lyricsto`-Befehl ruft den Gesangstextmodus automatisch auf, sodass man `\lyricmode` in diesem Fall auslassen kann. Standardmäßig werden die Silben unter den Noten angeordnet. Für andere Optionen siehe [Gesangstext vertikal verschieben], Seite 240.

`\addlyrics` benutzen

Der `\addlyrics`-Befehl ist eigentlich nur eine Abkürzung für eine etwas kompliziertere LilyPond-Struktur, den man manchmal aus Bequemlichkeit einsetzen kann.
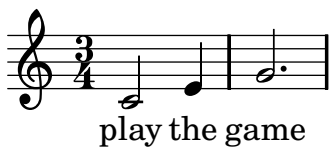
```
{ Noten }
\addlyrics { Gesangstext }
```

bedeutet das Gleiche wie

```
\new Voice = "bla" { Noten }
\new Lyrics \lyricsto "bla" { Gesangstext }
```

Hier ein Beispiel:
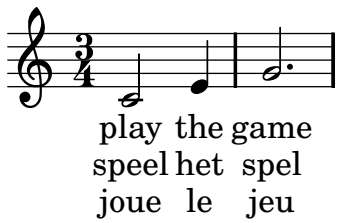
```
{
  \time 3/4
  \relative c' { c2 e4 g2. }
  \addlyrics { play the game }
}
```



Weitere Strophen können mit weiteren `\addlyrics`-Abschnitten hinzugefügt werden:

```
{
  \time 3/4
  \relative c' { c2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
```

```
  \addlyrics { joue le jeu }
}
```



Der Befehl `\addlyrics` kann keine polyphonen Situationen bewältigen. In diesen Fällen sollen man `\lyricsto` benutzen.

`associatedVoice` benutzen
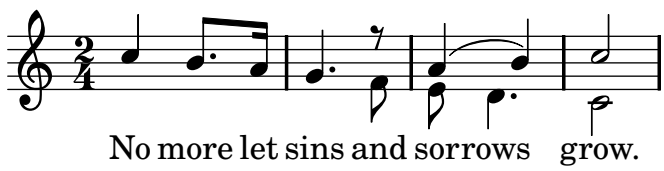
Die Melodie, an die der Gesangstext ausgerichtet wird, kann durch Setzen der `associatedVoice`-Eigenschaft geändert werden:

```
\set associatedVoice = #"lala"
```

Der Wert der Eigenschaft (hier `"lala"`) ist die Bezeichnung eines `Voice`-Kontextes. Aus technischen Gründen muss der `\set`-Befehl eine Silbe vor der Silbe gesetzt werden, auf die er wirken soll.

Ein Beispiel demonstiert das:

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative c'' {
      \voiceOne
      c4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative c' {
      \voiceTwo
       s2 s4. f8 e8 d4. c2
    }
  >>
% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



Siehe auch

Notationsreferenz: [Fülllinien und Trennstriche], Seite 238, Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516, [Gesangstext vertikal verschieben], Seite 240.

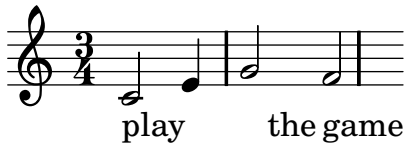
Manuelle Silbendauer

In komplexer Vokalmusik kann es nötig sein, den Gesangstext vollkommen unabhängig von den Noten zu positionieren. In diesem Fall sollte man nicht `\addlyrics` bzw. `\lyricsto` benutzen, und auch keine `associatedVoice` definieren. Die Silben werden wie Noten notiert – indem die Tonhöhen durch den Text der Silbe ersetzt werden – und die Dauer jeder Silbe muss angegeben werden.

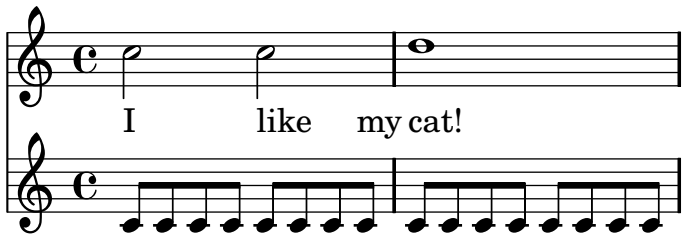
Standardmäßig werden die Silben links am entsprechenden musikalischen Moment ausgerichtet. Bindestriche zwischen den Silben können wie üblich gezogen werden, aber Fülllinien hinter dem Wortende können nicht gezogen werden, wenn es keine mit dem Text verknüpfte Melodie gibt.

Hier zwei Beispiele:

```
<<
  \new Voice = "melody" {
    \time 3/4
    c2 e4 g2 f
  }
  \new Lyrics \lyricmode {
    play1 the4 game4
  }
>>
```
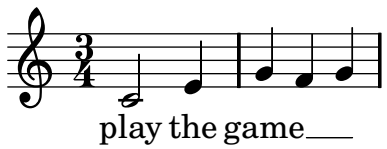



```
<<
  \new Staff {
    \relative c'' {
      c2 c2
      d1
    }
  }
  \new Lyrics {
    \lyricmode {
      I2 like4. my8 cat!1
    }
  }
  \new Staff {
    \relative c' {
      c8 c c c c c c c
      c8 c c c c c c c
    }
  }
>>
```



Diese Technik ist nützlich, wenn man Dialog zur Musik schreiben will, siehe [Dialog zur Musik], Seite 273.

Um Silben an den Noten des entsprechenden musikalischen Moments zu zentrieren, muss `associatedVoice` auf die Bezeichnung des Stimmen-Kontext eingestellt werden, in dem sich die Noten befinden. Wenn `associatedVoice` definiert ist, können doppelte Bindestriche zwischen Silben und doppelte Unterstriche hinter Wörtern für Fülllinien benutzt werden:

```
<<
  \new Voice = "melody" {
    \time 3/4
    c2 e4 g f g
  }
  \new Lyrics \lyricmode {
    \set associatedVoice = #"melody"
    play2 the4 game2. __
  }
>>
```



Siehe auch

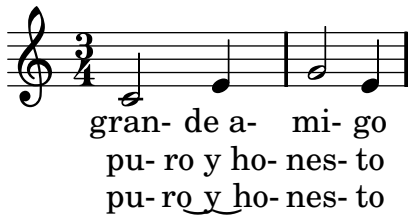
Notationsreferenz: [Dialog zur Musik], Seite 273, Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516.

Referenz der Interna: Abschnitt “Lyrics” in *Referenz der Interna*, Abschnitt “Voice” in *Referenz der Interna*.

Mehrere Silben zu einer Note

Um mehr als eine Silbe zu einer Note zuzuordnen, können die Silben mit geraden Anführungszeichen (") umgeben werden oder ein Unterstrich (_) benutzt werden, um ein Leerzeichen zwischen Silben zu setzen. Mit der Tilde (~) kann ein Bindebogen gesetzt werden. Dies erfordert, dass eine Schriftart vorhanden ist, die das entsprechende Symbol (U+203F) beinhaltet. Frei erhältliche Schriftarten, die den Bindebogen enthalten, sind „FreeSerif“ (ein Times-Klon), „DejaVuSans“ (aber nicht DejaVuSerif) oder TeXGyreSchole (ein Century Schoolbook-Klon).

```
{
  \time 3/4
  \relative c' { c2 e4 g2 e4 }
  \addlyrics { gran- de_a- mi- go }
  \addlyrics { pu- "ro y ho-" nes- to }
  \addlyrics { pu- ro~y~ho- nes- to }
}
```



Siehe auch

Referenz der Interna: Abschnitt “LyricCombineMusic” in *Referenz der Interna*.

Mehrere Noten zu einer Silbe

Öfters, insbesondere in Alter Musik, wird eine einzige Silbe zu mehreren Noten gesungen, was als Melisma bezeichnet wird. Die Silbe eines Melismas wird normalerweise links an der ersten Note des Melismas ausgerichtet.

Melismen können direkt im Gesangstext definiert werden, indem ein Unterstrich (_) für jede Note notiert wird, die übersprungen werden soll.
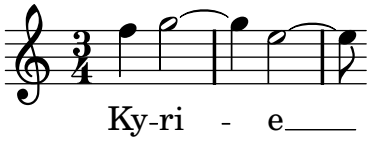
Wenn ein Melisma an einer Silbe auftritt, die nicht die letzte eines Wortes ist, wird diese Silbe mit der folgenden durch wiederholte Trennstriche verbunden. Dies wird notiert, indem man zwei Trennstriche (--) nach der Silbe notiert.

Wenn ein Melisma an der letzten Silbe eines Wortes auftritt, wird eine Fülllinie vom Ende der Silbe bis zur letzten Note des Melismas gezeichnet. Das wird durch zwei Unterstriche (__) nach der Silbe notiert.

Es gibt fünf Arten, auf die ein Melisma angezeigt werden kann:

- Melismen werden automatisch zu Noten erstellt, die übergebunden sind:
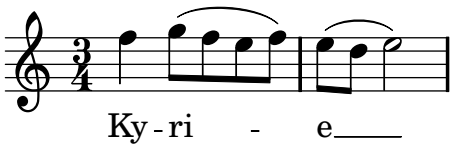
```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g2 ~ |
    g4 e2 ~ |
    e8
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```



- Melismen können automatisch aus den Noten erstellt werden, indem man Legatobögen über den Noten eines Melismas notiert. Auf diese Weise wird Gesangstext üblicherweise notiert:
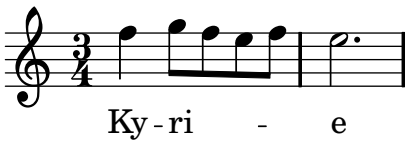
```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g8 ( f e f )
    e8 ( d e2 )
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
```

```
>>
```



- Noten werden als ein Melisma betrachtet, wenn sie manuell mit einem Balken versehen werden, vorausgesetzt, dass die automatische Bebalkung ausgeschaltet ist. Siehe [Einstellung von automatischen Balken], Seite 77.
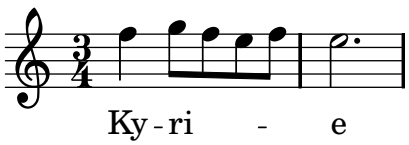
```
<<
  \new Voice = "melody" {
    \time 3/4
    \autoBeamOff
    f4 g8[ f e f]
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```



Diese Methode eignet sich natürlich nicht für Noten, die länger als Achtel sind.

- Eine Gruppe von Noten ohne Legatobogen werden als Melisma betrachtet, wenn sie zwischen `\melisma` und `\melismaEnd` eingeschlossen sind:
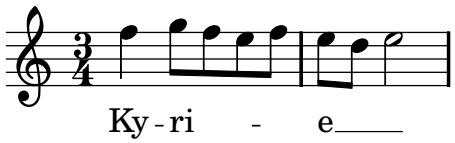
```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g8
    \melisma
    f e f
    \melismaEnd
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```



Mit dieser Methode können allerdings nicht zwei Melismen notiert werden, wenn sie direkt aufeinander folgen.
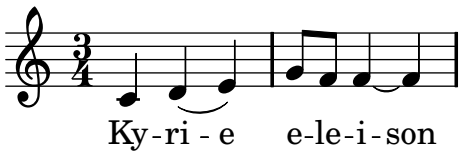
- Ein Melisma kann auch ausschließlich im Gesangstext notiert werden, indem man einzlene Unterstriche (_) für jede Note eingibt, die zum Melisma hinzugefügt werden soll.

```
<<
  \new Voice = "melody" {
    \time 3/4
    f4 g8 f e f
    e8 d e2
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ _ _ e __ _ _
  }
>>
```



Man kann durchaus auch Binde- und Legatobögen sowie manuelle Balken benutzen, ohne dass sie Melismen bezeichnen, wenn `melismaBusyProperties` aufgerufen wird:

```
<<
  \new Voice = "melody" {
    \time 3/4
    \set melismaBusyProperties = #'()
    c4 d ( e )
    g8 [ f ] f4 ~ f
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e e -- le -- i -- son
  }
>>
```
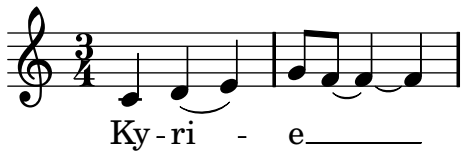


Andere Einstellungen für `melismaBusyProperties` können bneutzt werden, um nur eine Auswahl von Binde-, Legatobögen und Balken zur automatischen Melismenerkennung gelten zu lassen. Siehe `melismaBusyProperties` in Abschnitt “Tunable context properties” in *Referenz der Interna*.

Alternativ kann auch `ignoreMelismata` auf wahr gesetzt werden, wenn alle Melisma-Bezeichnungen ignoriert werden sollen, siehe [Strophen mit unterschiedlichem Rhythmus], Seite 256.

Wenn ein Melisma während einer Passage benötigt wird, in der `melismaBusyProperties` aktiviert ist, kann es durch einen einzelnen Unterstrich im Gesangstext für jede Note des Melismas angegeben werden:

```
<<
  \new Voice = "melody" {
    \time 3/4
    \set melismaBusyProperties = #'()
    c4 d ( e )
    g8 [ f ] ~ f4 ~ f
```

```
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ e __ _ _ _
  }
>>
```



Vordefinierte Befehle

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

Siehe auch

Glossar: Abschnitt “melisma” in *Glossar*.

Handbuch zum Lernen: Abschnitt “Text an einer Melodie ausrichten” in *Handbuch zum Lernen*.

Notation Reference: Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516, [Automatische Silbendauern], Seite 230, [Einstellung von automatischen Balken], Seite 77, [Strophen mit unterschiedlichem Rhythmus], Seite 256.

Internals Reference: Abschnitt “Tunable context properties” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Fülllinien zu Melismen werden nicht automatisch erkannt, sondern müssen manuell gesetzt werden.

Fülllinien und Trennstriche

Wenn die letzte Silbe eines Wortes auf ein Melisma fällt, wird das Melisma oft mit einer langen horizontalen Linie angezeigt, die nach dem Wort beginnt und mit der letzten Note des Melismas endet. Derartige Fülllinien werden mit einem doppelten Unterstrich (`__`) eingegeben, wobei beachtet werden muss, dass er von Leerzeichen umgeben ist.

Achtung: Melismen werden mit Fülllinien angezeigt, die als doppelter Unterstrich notiert sind. Kurze Melismen können auch notiert werden, indem eine Note übersprungen wird. Hierzu wird ein einfacher Unterstrich notiert und keine Fülllinie gezogen.

Zentrierte Bindestriche zwischen den einzelnen Silben werden mit einem doppelten Bindestrich (`--`) eingegeben, wobei beachtet werden muss, dass er von Leerzeichen umgeben ist. Der Bindestrich wird zwischen den Silben zentriert und seine Länge dem Notenabstand angepasst.

In sehr eng notierter Musik können die Bindestriche ganz wegfallen. Dieses Verhalten kann aber auch unterbunden werden, wenn den Eigenschaften `minimum-distance` (minimaler Abstand zwischen Silben) und `minimum-length` (Wert, unterhalb von dem Bindestriche wegfallen) andere Werte erhalten. Beide sind Eigenschaften von `LyricHyphen`.

Siehe auch
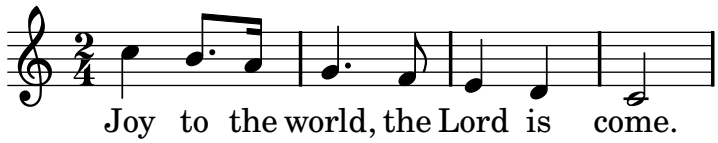
Referenz der Interna: Abschnitt “LyricExtender” in *Referenz der Interna*, Abschnitt “LyricHyphen” in *Referenz der Interna*.

2.1.2 Techniken für die Gesangstextnotation

Mit Gesangstexten und Bezeichnern arbeiten

Um Variablen zu definieren, die Gesangstext beinhalten, muss die `\lyricmode`-Umgebung benutzt werden.

```
musicOne = \relative c'' {
  c4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  <<
    \new Voice = "one" {
      \time 2/4
      \musicOne
    }
    \new Lyrics \lyricsto "one" {
      \verseOne
    }
  >>
}
```
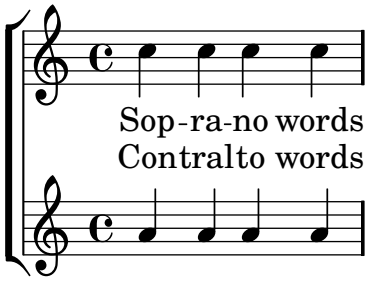


Dauern müssen nicht angegeben werden, wenn die Variable im Zusammenhang mit `\addlyrics` oder `\lyricsto` aufgerufen wird.

Für eine andere Anordnung oder kompliziertere Situationen bietet es sich an, zuerst Systeme und Gesangstextumgebungen zu definieren, dann die Hierarchie von Systemen und Gesangstextzeilen aufzustellen ohne den Gesangstext selber aufzurufen, und dann den Gesangstext mit `\context` darunter aufzurufen. Das stellt sicher, dass die Stimmen, die durch `\lyricsto` angefordert werden, auch immer schon definiert sind. Beispielsweise:

```
sopranoMusic = \relative c'' { c4 c c c }
contraltoMusic = \relative c'' { a4 a a a }
sopranoWords = \lyricmode { Sop -- ra -- no words }
contraltoWords = \lyricmode { Con -- tral -- to words }

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \sopranoMusic
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos"
    \new Staff {
      \new Voice = "contraltos" {
        \contraltoMusic
      }
```

```
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        \sopranoWords
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        \contraltoWords
      }
    }
  >>
}
```



Siehe auch

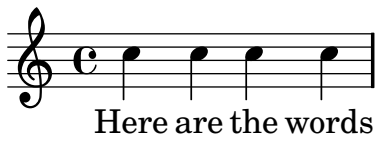
Notationsreferenz: [Gesangstext vertikal verschieben], Seite 240.

Referenz der Interna: Abschnitt “LyricCombineMusic” in *Referenz der Interna*, Abschnitt “Lyrics” in *Referenz der Interna*.

Gesangstext vertikal verschieben

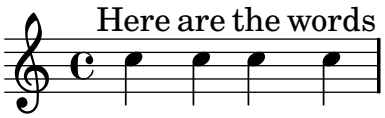
Abhängig von der Art der Musik kann der Gesangstext über oder unter einem Notensystem oder zwischen zwei Systemen positioniert werden. Es ist am einfachsten, den Text unter das verknüpfte System zu positionieren, was man erreicht, indem man den `Lyrics`-Kontext direkt unter dem System definiert:

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' { c4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



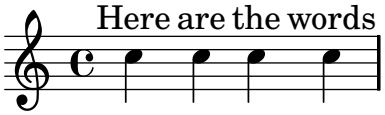
Gesangstext kann auch über dem System positioniert werden, indem man eine der beiden folgenden Methoden benutzt. Die einfachste (und bevorzugte) Methode ist es, die gleiche Syntax wie oben gezeigt einzusetzen und die Position des Gesangstextes explizit anzugeben:

```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative c'' { c4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```
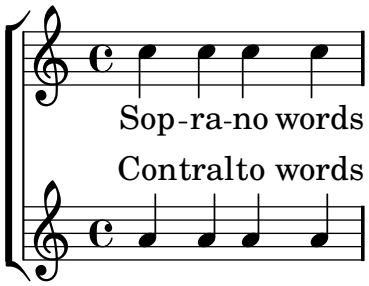


Alternativ kann auch ein zweistufiger Prozess benutzt werden. Zuerst wird der Gesangstextkontext definiert (ohne jeglichen Inhalt), bevor Stimm- und Systemkontexte definiert wurden. Dann wird der `\lyricsto`-Befehl nach der Definition der Stimme, auf die er verweist, mit `\context` angegeben:

```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup #'staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "melody" {
        \relative c'' { c4 c c c }
      }
    }
    \context Lyrics = "lyrics" {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Wenn zwei Stimmen sich auf unterschiedlichen Systemen befinden, kann der Text zwischen den Systemen platziert werden, wobei beide der Methoden einesetzt werden können. Hier ein Beispiel für die zweite Methode:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative c'' { c4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup #'staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "contraltos" {
        \relative c'' { a4 a a a }
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        Sop -- ra -- no words
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        Con -- tral -- to words
      }
    }
  >>
}
```



Andere Kombinationen von Gesangstext und System können erstellt werdne, indem man die gegebenen Beispiele modifiziert oder auch die Abschnitt “Vokalensemble” in *Handbuch zum Lernen*-Vorlagen im Handbuch zum Lernen heranzieht.

Ausgewählte Schnipsel

Textverteilung wie in Version 2.12 für neuere Versionen

Die vertikale Maschine für die Platzverteilung wurde für die Version 2.14 verändert. Das kann dazu führen, dass Gesangstext unterschiedlich verteilt wird. Es ist möglich, die Eigenschaften der `Lyric`- und `Staff`-Kontexte so einzustellen, dass die Platzverteilung wie in der Version 2.12 arbeitet.

```
global = {
  \key d \major
```

```
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

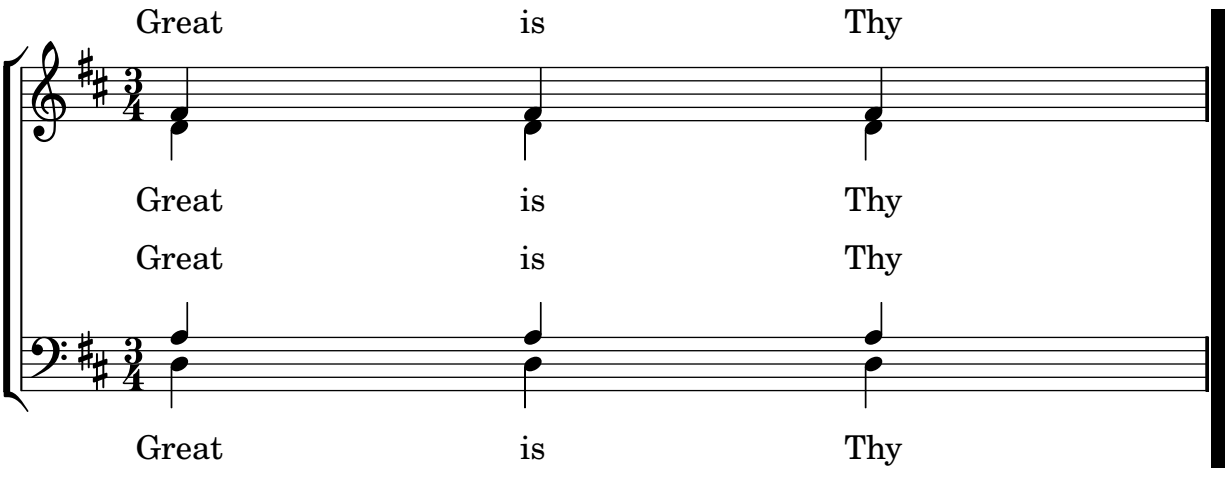
tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
}

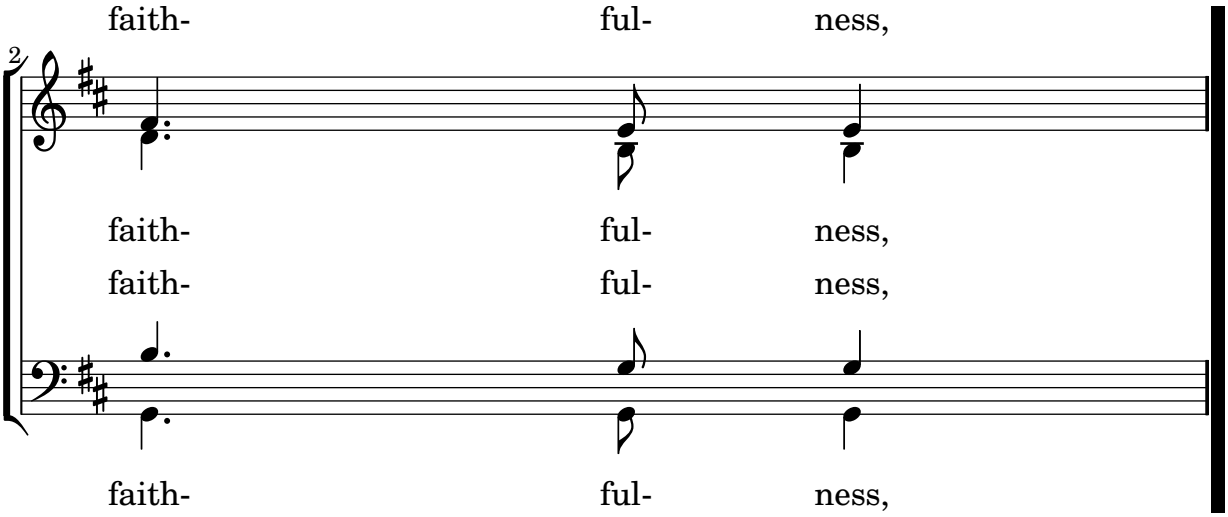
bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}

words = \lyricmode {
  Great is Thy faith- ful- ness,
}

\score {
  \new ChoirStaff <<
    \new Lyrics = sopranos
    \new Staff = women <<
      \new Voice = "sopranos" {
        \voiceOne
        \global \sopMusic
      }
      \new Voice = "altos" {
        \voiceTwo
        \global \altoMusic
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors"
    \new Staff = men <<
      \clef bass
      \new Voice = "tenors" {
        \voiceOne
        \global \tenorMusic
      }
      \new Voice = "basses" {
        \voiceTwo  \global \bassMusic
```

```
      }
    >>
    \new Lyrics = basses
    \context Lyrics = sopranos \lyricsto sopranos \words
    \context Lyrics = altos \lyricsto altos \words
    \context Lyrics = tenors \lyricsto tenors \words
    \context Lyrics = basses \lyricsto basses \words
  >>
  \layout {
    \context {
      \Lyrics
      \override VerticalAxisGroup #'staff-affinity = ##f
      \override VerticalAxisGroup #'staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
    \context {
      \Staff
      \override VerticalAxisGroup #'staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
  }
}
```





Siehe auch

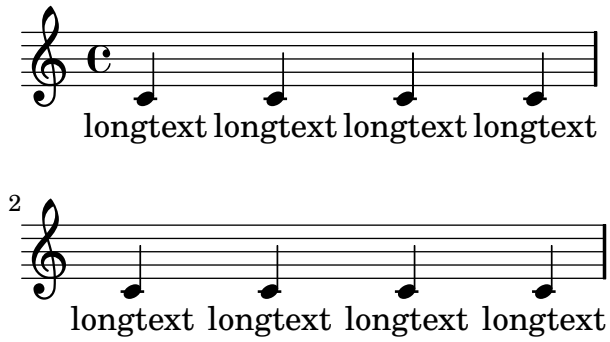
Handbuch zum Lernen: Abschnitt “Vokalensemble” in *Handbuch zum Lernen*.

Notation Reference: Abschnitt 5.1.7 [Kontexte aneinander ausrichten], Seite 524, Abschnitt 5.1.2 [Kontexte erstellen], Seite 515.

Silben horizontal verschieben

Um den Abstand zwischen Silben zu vergrößern, kann die `minimum-distance`-Eigenschaft des `LyricSpace`-Objekts gesetzt werden:

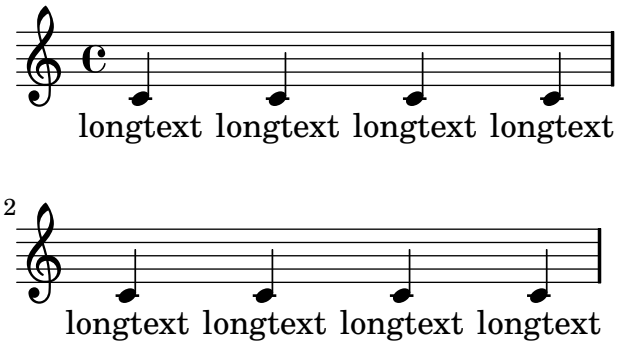
```
{
  c c c c
  \override Lyrics.LyricSpace #'minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```



Damit diese Einstellung für alle Gesangtextzeilen in einer Partitur wirkt, muss sie im `layout`-Block vorgenommen werden.

```
\score {
  \relative c' {
  c c c c
  c c c c
  }
  \addlyrics {
  longtext longtext longtext longtext
```

```
  longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace #'minimum-distance = #1.0
    }
  }
}
```

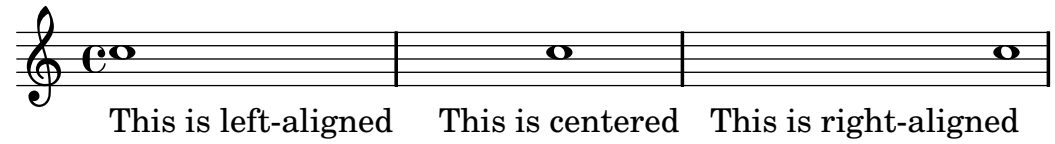


Ausgewählte Schnipsel

Ausrichtung von Gesangstext

Die horizontale Ausrichtung von Gesangstext kann eingestellt werden, indem man die `self-alignment-X`-Eigenschaft des `LyricText`-Objekts verändert. `#-1` bedeutet links, `#0` bedeutet mittig und `#1` bedeutet rechts, man kann aber genauso gut auch `#LEFT`, `#CENTER` und `#RIGHT` benutzen.

```
\layout { ragged-right = ##f }
\relative c'' {
  c1
  c1
  c1
}
\addlyrics {
  \once \override LyricText #'self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText #'self-alignment-X = #CENTER
  "This is centered"
  \once \override LyricText #'self-alignment-X = #1
  "This is right-aligned"
}
```



Eine Überprüfung, mit der sichergestellt wird, dass kein Text in die Seitenränder ragt, verlangt zusätzliche Computerzeit. Um den Notensatz etwas zu beschleunigen, kann die Überprüfung abgestellt werden:

```
\override Score.PaperColumn #'keep-inside-line = ##f
```

Damit Gesangstext auch nicht mit Taktlinien zusammenstößt, kann folgende Einstellung gesetzt werden:

```
\layout {
  \context {
    \Lyrics
      \consists "Bar_engraver"
      \consists "Separating_line_group_engraver"
      \override BarLine #'transparent = ##t
  }
}
```
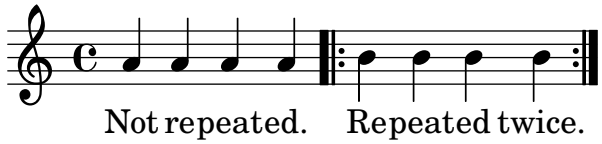
Gesangstext und Wiederholungen

Einfache Wiederholungen

Wiederholungen von *Musik* ist vollständig an anderer Stelle behandelt, siehe Abschnitt 1.4 [Wiederholungszeichen], Seite 128. Dieser Abschnitt erklärt, wie man Gesangstext zu wiederholten Noten hinzufügt.

Gesangstext zu einem Abschnitt, der wiederholt wird, muss in der gleichen Wiederholungskonstruktion wie die Noten enthalten sein, wenn der Text sich nicht ändert:
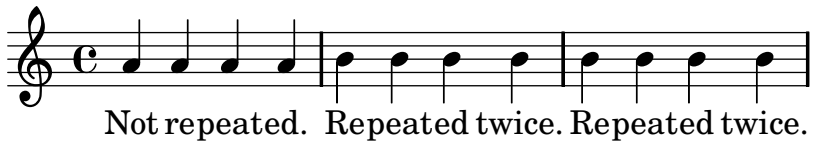
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- ed twice. }
      }
    }
  >>
}
```



Der Text wird dann richtig erweitert, wenn die Wiederholung mit `\unfoldRepeats` ausgeklappt wird:
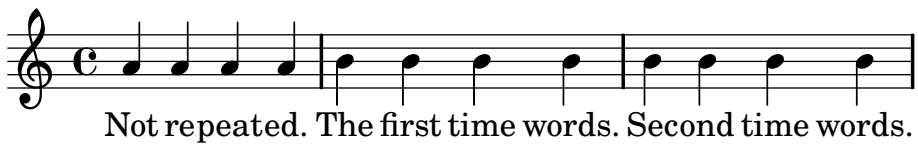
```
\score {
  \unfoldRepeats {
    <<
      \new Staff {
        \new Voice = "melody" {
```

```
          \relative c'' {
            a4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Not re -- peat -- ed.
          \repeat volta 2 { Re -- peat -- ed twice. }
        }
      }
    >>
  }
}
```
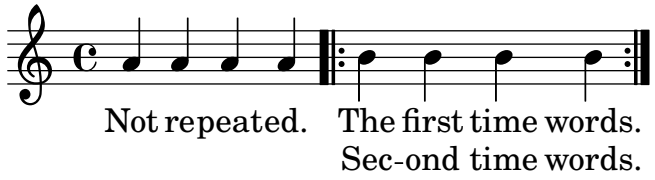


Wenn der wiederholte Abschnitt unterschiedlichen Text hat und ausgeklappt werden soll, müssen alle Wörter notiert werden:

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        The first time words.
        Sec -- ond time words.
      }
    }
  >>
}
```



Wenn der Text in einer Wiederholung mit `volta` (also mit punktiertem Doppelstrich) unterschiedlich ist, muss der Text jeder Wiederholung in einem eigenen `Lyrics`-Kontext notiert werden, der korrekt in parrallelen Abschnitten geschachtelt wird:

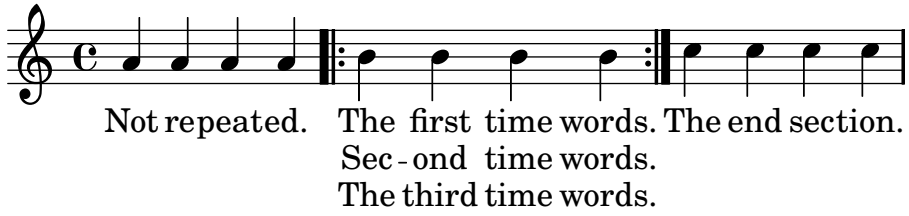
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics \lyricsto "melody" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics {
          \set associatedVoice = "melody"
          Sec -- ond time words.
        }
      >>
    }
  >>
}
```



Neue Strophen können auf die gleiche Art hinzugefügt werden:

```
\score {
  <<
    \new Staff {
      \new Voice = "singleVoice" {
        \relative c'' {
          a4 a a a
          \repeat volta 3 { b4 b b b }
          c4 c c c
        }
      }
    }
    \new Lyrics \lyricsto "singleVoice" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics {
          \set associatedVoice = "singleVoice"
          Sec -- ond time words.
        }
        \new Lyrics {
          \set associatedVoice = "singleVoice"
          The third time words.
```
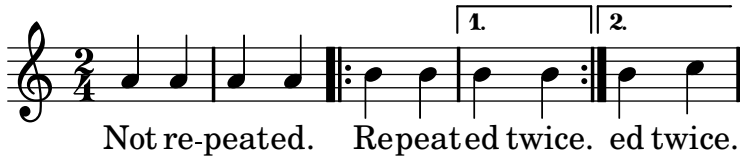
```
        }
      >>
      The end sec -- tion.
    }
  >>
}
```



Wiederholungen mit alternativen Endungen

Wenn der Text des wiederholten Abschnitts der gleiche ist, kann die gleiche Struktur für Gesangstext und Noten eingesetzt werden.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat volta 2 { b4 b }
          \alternative { { b b } { b c } }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- }
        \alternative { { ed twice. } { ed twice. } }
      }
    }
  >>
}
```
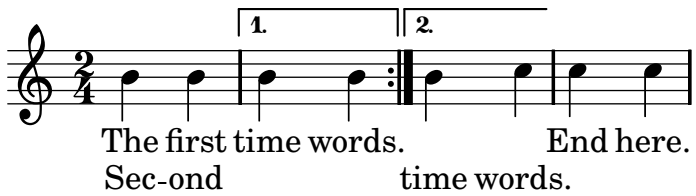


Aber wenn der wiederholte Abschnitt unterschiedlichen Text hat, kann keine `repeat`-Konstruktion um den Text eingesetzt werden und man muss manuell mit `\skip`-Befehlen die Noten überspringen, auf die sich der alternative Text nicht bezieht.

Achtung: hier darf kein Unterstrich (`_`) benutzt werden, um Noten zu überspringen, weil das ein Melisma anzeigen würde und die vorhergehende Silbe dazu veranlasst, links ausgerichtet zu werden.

Achtung: Der `\skip`-Befehl muss von einer Zahl gefolgt werden, aber diese Zahl wird ignoriert, wenn der Gesangstext seine Dauern von einer assoziierten Melodie ableitet, die mit `\addlyrics` oder `\lyricsto` angefügt wird. Jeder `\skip`-Befehl überspringt eine einzelne Note beliebiger Dauer, unabhängig vom Wert der auf den Befehl folgenden Zahl.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \repeat volta 2 { b4 b }
          \alternative { { b b } { b c } }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        The first time words.
        \repeat unfold 2 { \skip 1 }
        End here.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Sec -- ond
        \repeat unfold 2 { \skip 1 }
        time words.
      }
    }
  >>
}
```

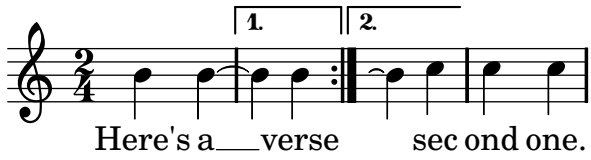


Wenn eine Note zu zwei oder mehr alternativen Endungen übergebunden wird, wird ein Bindebogen benutzt, um die Note in die erste alternative Endung überzubinden, und ein `\repeatTie` benutzt, um in die zweite und alle weiteren Klammern zu überbinden. Diese Struktur erzeugt problematische Ausrichtungen, wenn ein Gesangstext hinzu kommt und verlängert die alternativen Klammern, sodass es besser sein kann, die übergebundenen Noten vollständig in die Klammern aufzunehmen, um ein besseres Resultat zu erhalten.

Der Bindebogen erstellt ein Melisma zur ersten Klammer, aber nicht zur zweiten und allen weiteren Klammern, sodass es nötig ist, die automatische Erstellung von Melismen für die Klammer-Abschnitte zu deaktivieren und manuell die Noten mit `\skip` zu überspringen, um eine richtige Ausrichtung des Textes zu erreichen.

```
\score {
  <<
    \new Staff {
```

```
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \set melismaBusyProperties = #'()
          \repeat volta 2 { b4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          \unset melismaBusyProperties
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        \repeat volta 2 { Here's a __ }
        \alternative {
          { \skip 1 verse }
          { \skip 1 sec }
        }
        ond one.
      }
    }
  >>
}
```
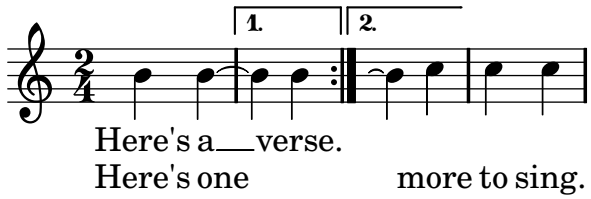


Wenn `\unfoldRepeats` in einem Abschnitt eingesetzt wird, der den `\repeatTie`-Befehl enthält, sollte der `\repeatTie` entfernt werden, damit nicht beide Bindestriche ausgegeben werden.

Wenn der wiederholte Abschnitt unterschiedlichen Text hat, kann `\repeat` nicht um den Gesangstext benutzt werden, und `\skip`-Befehle müssen manuell eingegeben werden:

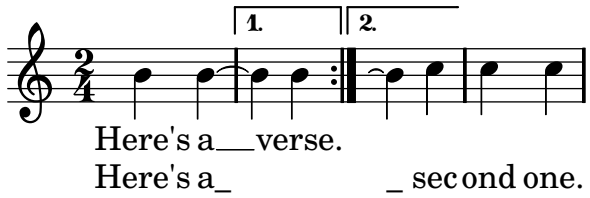
```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \repeat volta 2 { b4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \repeat unfold 2 { \skip 1 }
      }
```

```
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's one
        \repeat unfold 2 { \skip 1 }
        more to sing.
      }
    }
  >>
}
```



Wenn Sie Bindestriche und Fülllinien zwischen Wiederholung und Klammer benutzen wollen, müssen sie manuell notiert werden:

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative c'' {
          \repeat volta 2 { b4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's "a_"
        \skip 1
        "_" sec -- ond one.
      }
    }
  >>
}
```
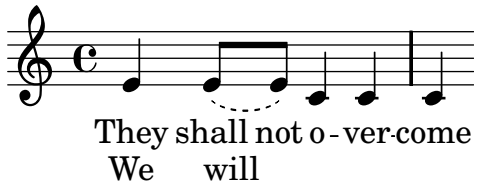


Siehe auch

Notationsreferenz: Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516, Abschnitt 1.4 [Wiederholungszeichen], Seite 128.

Getrennte Texte

Wenn nur Text und Rhythmus von zwei Texten unterschiedlich sind, während die Noten gleich bleiben, kann man die automatische Melisma-Erkennung kurzzeitig ausschalten und das Melisma im Text anzeigen:
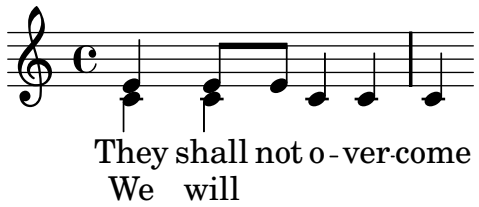
```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        \set melismaBusyProperties = #'()
        \slurDown
        \slurDashed
        e4 e8 ( e ) c4 c |
        \unset melismaBusyProperties
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "melody" {
      We will _
    }
  >>
}
```



Wenn sich sowohl Noten als auch Worte unterscheiden, kann es besser sein, die unterschiedlichen Noten und den Text zu notieren, indem man `Voice`-Kontexte benennt und den Text an die enstprechenden Kontexte anhängt:

```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        <<
          {
            \voiceOne
            e4 e8 e
          }
```

```
          \new Voice = "splitpart" {
            \voiceTwo
            c4 c
          }
        >>
        \oneVoice
        c4 c |
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "splitpart" {
      We will
    }
  >>
}
```

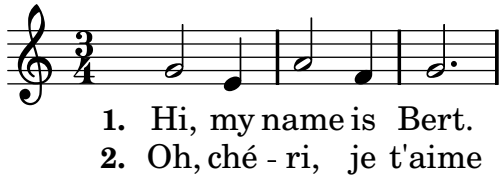


2.1.3 Strophen

Strophennummern hinzufügen

Strophennummerierung kann hinzugefügt werden:

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = #"2. "
  Oh, ché -- ri, je t'aime
}
```
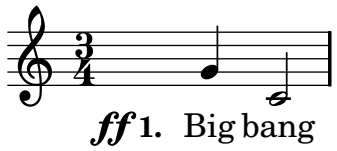


Die Zahl wird direkt vor die erste Silbe gesetzt.

Lautstärkebezeichnung zu Strophen hinzufügen

Dynamikzeichen können zur Strophennummer hinzugefügt werden. In LilyPond muss alles, was vor einer Strophe gesetzt wird, im `StanzaNumber`-Objekt definiert werden, also auch Dynamikbezeichnung. Aus technischen Gründen muss die Strophe außerhalb von `lyricmode` gesetzt werden:

```
text = {
  \set stanza = \markup { \dynamic "ff" "1. " }
  \lyricmode {
    Big bang
  }
}

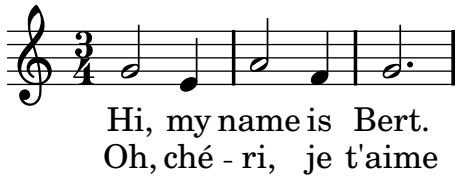
<<
  \new Voice = "tune" {
    \time 3/4
    g'4 c'2
  }
\new Lyrics \lyricsto "tune" \text
>>
```



Sängernamen zu Strophen hinzufügen

Namen von Sängern können auch eingefügt werden. Sie werden zu Beginn der Zeile gesetzt, ähnlich wie eine Instrumentenbezeichnung. Sie werden mit der `vocalName`-Eigenschaft erstellt. Eine Kurzversion kann mit `shortVocalName` definiert werden.

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = #"Bert "
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = #"Ernie "
  Oh, ché -- ri, je t'aime
}
```



Strophen mit unterschiedlichem Rhythmus

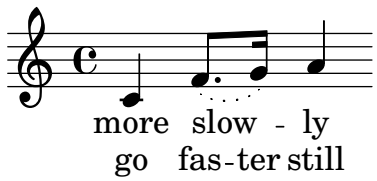
Oft haben unterschiedliche Strophen eines Liedes leicht unterschiedliche Silbenzahlen und werden darum auf andere Art zur Melodie gesungen. Derartige Variationen können mit `\lyricsto` bewältigt werden.

Melismen ignorieren

Teilweise wird zu einer Silbe ein Melisma in einer Strophe gesungen, während in einer anderen jede Note eine Silbe erhält. Eine Möglichkeit ist, dass die Strophe mit mehr Text das Melisma ignoriert. Das wird mit der `ignoreMelismata`-Eigenschaft im `Lyrics`-Kontext vorgenommen.

```
<<
  \relative c' \new Voice = "lahlah" {
```

```
    \set Staff.autoBeaming = ##f
    c4
    \slurDotted
    f8.[( g16])
    a4
  }
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    go
    \set ignoreMelismata = ##t
    fas -- ter
    \unset ignoreMelismata
    still
  }
>>
```



Bekannte Probleme und Warnungen
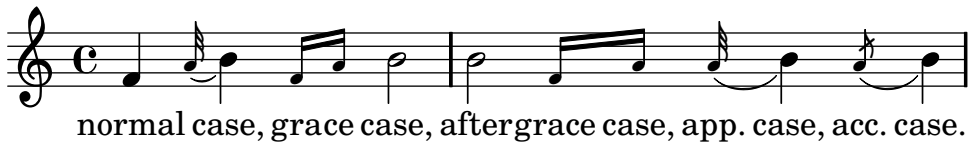
Anders als die meisten `\set`-Befehle funktioniert `\set ignoreMelismata` nicht zusammen mit `\once`. Es ist notwendig, explizit `\set` und `\unset` zu verwenden, um den Text einzugrenzen, für den Melismen ignoriert werden sollen.

Silben zu Verzierungsnoten hinzufügen

Normalerweise werden Verzierungsnoten (z.B. durch `\grace`) bei `\lyricsto` keine Silben zugeordnet. Dieses Verhalten kann geändert werden, wie das folgende Beispiel zeigt.

```
<<
  \new Voice = melody \relative c' {
    f4 \appoggiatura a32 b4
    \grace { f16[ a16] } b2
    \afterGrace b2 { f16[ a16] }
    \appoggiatura a32 b4
    \acciaccatura a8 b4
  }
  \new Lyrics
  \lyricsto melody {
    normal
    \set includeGraceNotes = ##t
    case,
    gra -- ce case,
    after -- grace case,
    \set ignoreMelismata = ##t
    app. case,
    acc. case.
  }
```
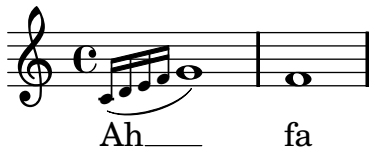
```
>>
```



Bekannte Probleme und Warnungen

Wie bei `associatedVoice` muss `includeGraceNotes` spätestens eine Silbe vor derjenigen gesetzt werden, die unter einer Verzierungsnote stehen soll. Im Fall, dass eine Verzierungsnote die erste des Musikstückes ist, kann ein `\with`- oder `\context`-Block verwendet werden:

```
<<
  \new Voice = melody \relative c' {
    \grace { c16[( d e f] }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>
```

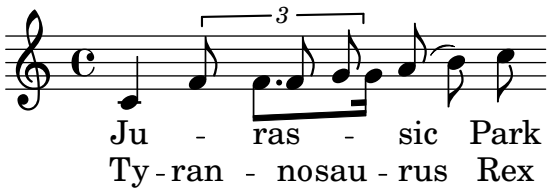


Zu einer alternativen Melodie umschalten

Es ist auch möglich, die Silben von verschiedenen Textzeilen an unterschiedlichen Melodien auszurichten. Das wird mit der `associatedVoice`-Eigenschaft vorgenommen:

```
<<
  \relative c' \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c4
    <<
      \new Voice = "alternative" {
        \voiceOne
        \times 2/3 {
          % show associations clearly.
          \override NoteColumn #'force-hshift = #-3
          f8 f g
        }
      }
      {
        \voiceTwo
        f8.[ g16]
        \oneVoice
      } >>
    a8( b) c
  }
```

```
\new Lyrics \lyricsto "lahlah" {
  Ju -- ras -- sic Park
}
\new Lyrics \lyricsto "lahlah" {
  % Tricky: need to set associatedVoice
  % one syllable too soon!
  \set associatedVoice = "alternative" % applies to "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = "lahlah" % applies to "rus"
  sau -- rus Rex
} >>
```



Der Text der ersten Strophe wird an der Stimme „lahlah“ ausgerichtet, aber die zweite Strophe wird zuerst zum `lahlah`-Kontext gesetzt und dann zur `alternative`-Melodie für die Silben „ran“ bis „sau“:

```
\set associatedVoice = "alternative" % applies to "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % applies to "rus"
sau -- rus Rex
```

Hier ist `alternative` die Bezeichnung des `Voice`-Kontexts mit der Triole.

Der `\set associatedVoice`-Befehl tritt eine Silbe zu früh auf, aber das ist in diesem Fall richtig.

Achtung: Der `\set associatedVoice`-Befehl muss eine Silbe *vor* der Silbe notiert werden, auf welcher der Wechsel zur neuen Stimme stattfinden soll. Anders gesagt: der Wechsel der assoziierten Stimme geschieht eine Silber später, als man erwarten würde. Das geschieht aus technischen Gründen – es handelt sich nicht um einen Fehler.

Die Strophen am Ende ausdrucken

Manchmal soll nur eine Strophe mit der Melodie gesetzt werden und die weiteren Strophen als Text unter den Noten hinzugefügt werden. Dazu wird der Text in einer `markup`-Umgebung außerhalb der `\score`-Umgebung gesetzt. Es gibt zwei Arten, die Zeilen auszurichten, wie das Beispiel zeigt:

```
melody = \relative c' {
e d c d | e e e e |
d d e d | c1 |
}

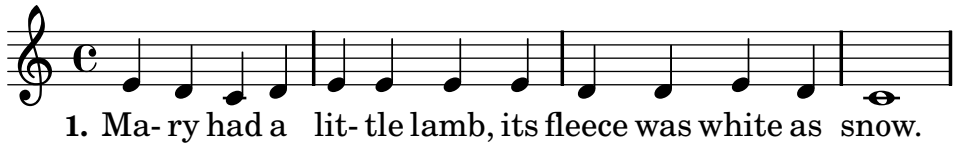
text = \lyricmode {
\set stanza = #"1." Ma- ry had a lit- tle lamb,
```

```
its fleece was white as snow.
}

\score{ <<
  \new Voice = "one" { \melody }
  \new Lyrics \lyricsto "one" \text
>>
  \layout { }
}
\markup { \column{
  \line{ Verse 2. }
  \line{ All the children laughed and played }
  \line{ To see a lamb at school. }
  }
}
\markup{
  \wordwrap-string #"
  Verse 3.

  Mary took it home again,

  It was against the rule."
}
```



Verse 2.
All the children laughed and played
To see a lamb at school.

Verse 3.
Mary took it home again,
It was against the rule.

Die Strophen am Ende in mehreren Spalten drucken
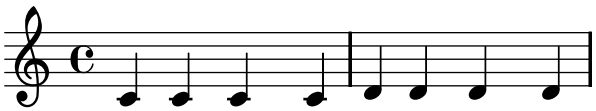
Wenn in einem Lied sehr viele Strophen vorkommen, werden sie oft in mehreren Spalten unter den Noten gesetzt. Eine nach außen versetzte Zahl zeigt die Strophennummer an. Dieses Beispiel zeigt eine Methode, diese Art von Notensatz zu produzieren.

```
melody = \relative c' {
  c c c c | d d d d
}

text = \lyricmode {
  \set stanza = #"1." This is verse one.
  It has two lines.
}
```

```
\score{ <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
   >>
  \layout { }
}

\markup {
  \fill-line {
    \hspace #0.1 % moves the column off the left margin;
        % can be removed if space on the page is tight
     \column {
      \line { \bold "2."
        \column {
          "This is verse two."
          "It has two lines."
        }
      }
      \hspace #0.1 % adds vertical spacing between verses
      \line { \bold "3."
        \column {
          "This is verse three."
          "It has two lines."
        }
      }
    }
    \hspace #0.1  % adds horizontal spacing between columns;
        % if they are still too close, add more " " pairs
        % until the result looks good
     \column {
      \line { \bold "4."
        \column {
          "This is verse four."
          "It has two lines."
        }
      }
      \hspace #0.1 % adds vertical spacing between verses
      \line { \bold "5."
        \column {
          "This is verse five."
          "It has two lines."
        }
      }
    }
  \hspace #0.1 % gives some extra space on the right margin;
      % can be removed if page space is tight
  }
}
```



1. This is verse one. It has two lines.

2. This is verse two.
It has two lines.

3. This is verse three.
It has two lines.

4. This is verse four.
It has two lines.

5. This is verse five.
It has two lines.

Siehe auch

Referenz der Interna: Abschnitt “LyricText” in *Referenz der Interna*, Abschnitt “StanzaNumber” in *Referenz der Interna*.

2.1.4 Lieder

Verweise für Lieder

Lieder werden normalerweise auf drei Systemen notiert: das oberste System für den Sänger und zwei Systeme für die Klavierbegleitung darunter. Der Gesangstext der ersten Strophe wird direkt unter dem ersten System ausgegeben. Wenn es nur eine kleine Anzahl weiterer Strophen gibt, können sie sofort unter der ersten gesetzt werden, aber wenn es viele Strophen gibt, werden die zweite und alle weiteren Strophen als Text unter den Noten gesetzt.

Alle Notationselemente, die für die Notation von Liedern benötigt werden, werden woanders beschrieben:

- Um das Systemlayout einzurichten, siehe Abschnitt 1.6.1 [Systeme anzeigen lassen], Seite 161.
- Zur Notation von Klaviermusik siehe Abschnitt 2.2 [Tasteninstrumente und andere Instrumente mit mehreren Systemen], Seite 287.
- Zur Notation von Gesangstext zu einer Notenzeile siehe Abschnitt 2.1.1 [Übliche Notation für Vokalmusik], Seite 226.
- Zur Platzierung des Gesangstext siehe [Gesangstext vertikal verschieben], Seite 240.
- Zur Notation von Strophen siehe Abschnitt 2.1.3 [Strophen], Seite 255.
- Lieder werden oft auch mit Akkorden gesetzt, die als Symbole über dem Notensystem notiert werden. Das wird in Abschnitt 2.7.2 [Akkorde anzeigen], Seite 376 beschrieben.
- Zur Notation von Bunddiagrammen für die Akkorde einer Gitarrenbegleitung oder anderer Bundinstrumente siehe „Bund-Diagramm-Beschriftung“ in Abschnitt 2.4.1 [Übliche Notation für Saiteninstrumente mit Bünden], Seite 303.

Siehe auch

Handbuch zum Lernen: Abschnitt “Lieder” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 2.1.1 [Übliche Notation für Vokalmusik], Seite 226, Abschnitt 2.7.2 [Akkorde anzeigen], Seite 376, Abschnitt 1.6.1 [Systeme anzeigen lassen], Seite 161, Abschnitt 2.2 [Tasteninstrumente und andere Instrumente mit mehreren Systemen], Seite 287, [Gesangstext vertikal verschieben], Seite 240, Abschnitt 2.1.3 [Strophen], Seite 255.

Schnipsel: Abschnitt “Vocal music” in *Schnipsel*.

Liedblätter

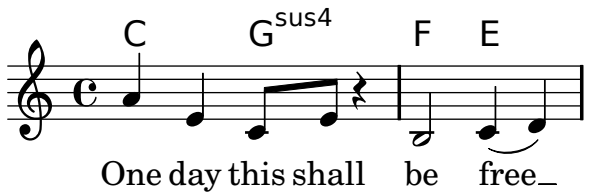
Liedblätter können erstellt werden, indem man Gesangstext mit Akkorden im Akkord-Modus kombiniert; die Syntax ist erklärt in Abschnitt 2.7 [Notation von Akkorden], Seite 371.

Ausgewählte Schnipsel

Ein einfaches Liedblatt

Ein Liedblatt besteht aus Akkordbezeichnungen, einer Melodie und dem Liedtext:

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Siehe auch

Notationsreferenz: Abschnitt 2.7 [Notation von Akkorden], Seite 371.

2.1.5 Chormusik

Dieser Abschnitt zeigt Eigenheiten der Notation von Chormusik. Hierzu gehören Hymnen, mehrstimmige Lieder, Oratorien, Kantaten usw.

Verweise für Chormusik

Chormusik wird normalerweise auf zwei, drei oder vier Systemen innerhalb einer `ChoirStaff`-Gruppe notiert. Begleitung wird darunter als `PianoStaff`-Klaviersystem gesetzt, oft auch in kleinerer Größe, wenn es sich um eine Übungshilfe für ein *a capella*-Chorwerk handelt. Die Noten jeder Stimme werden in einem `Voice`-Kontext notiert, welche entweder einzeln auf einem eigenen Notensytem notiert werden oder zu zweit auf dem gleichen System gesetzt werden.

Gesangstext wird in `Lyrics`-Kontext gesetzt, entweder unter dem zugehörigen System oder ein Text über dem System, der andere darunter, wenn das System die Noten von zwei Stimmen enthält.

Einige häufig anzutreffende Sachverhalte für Chormusik sind woanders behandelt:

- Eine Einleitung, um SATB-Chorpartituren zu erstellen, findet sich in Abschnitt “Vierstimmige SATB-Partitur” in *Handbuch zum Lernen*.
- Einige Vorlagen, die sich für unterschiedliche Chormusik eignen, finden sich im Handbuch zum Lernen, siehe Abschnitt “Vokalensemble” in *Handbuch zum Lernen*.
- Zu Information über `ChoirStaff` und `PianoStaff` siehe [Systeme gruppieren], Seite 162.
- Besondere Notenköpfe, wie etwa für die „Sacred Harp“-Notation benutzt, finden sich in [Notenköpfe mit besonderen Formen], Seite 37.
- Wenn zwei Vokalstimmen sich ein System teilen, werden Hälse, Bögen usw. der oberen Stimme nach oben gerichtet und die der unteren Stimme nach unten. Dieses Verhalten erreicht man mit `\voiceOne` und `\voiceTwo`. Siehe [Mehrstimmigkeit in einem System], Seite 146.

Vordefinierte Befehle

`\oneVoice`, `\voiceOne`, `\voiceTwo`.

Siehe auch

Handbuch zum Lernen: Abschnitt “Vierstimmige SATB-Partitur” in *Handbuch zum Lernen*, Abschnitt “Vokalensemble” in *Handbuch zum Lernen*.

Notationsreferenz: [Systeme gruppieren], Seite 162, Abschnitt 5.4.3 [Reihenfolge des Kontextlayouts], Seite 539, [Notenköpfe mit besonderen Formen], Seite 37, [Mehrstimmigkeit in einem System], Seite 146.

Referenz der Interna: Abschnitt “ChoirStaff” in *Referenz der Interna*, Abschnitt “Lyrics” in *Referenz der Interna*, Abschnitt “PianoStaff” in *Referenz der Interna*.

Partiturbeispiele für Chormusik

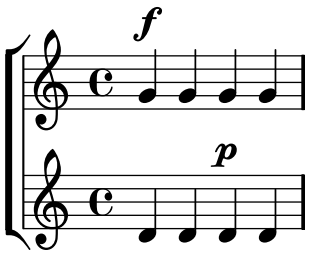
Chormusik auf vier Systemen, mit oder ohne Klavierbegleitung, wird meistens mit zwei Systemgruppen pro Seite gesetzt. Abhängig von der Seitengröße kann das erfordern, dass die Standardgröße der Notensysteme geändert wird. Die folgenden Einstellungen sollten in Betracht gezogen werden:

- Die globale Systemgröße kann verändert werden, um die Größe aller Elemente einer Partitur zu ändern. Siehe Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472.
- Die Abstände zwischen den Systemen, den Systemgruppen und den Gesangstexten können alle einzeln eingestellt werden. Siehe Abschnitt 4.4 [Vertikale Abstände], Seite 481.
- Die Dimensionen der vertikalen Layout-Variablen können angezeigt werden, um beim Anpassen der vertikalen Platzverteilung zu helfen. Das und andere Möglichkeiten, die Noten auf weniger Seiten zu zwingen, finden sich in Abschnitt 4.6 [Die Musik auf weniger Seiten zwingen], Seite 509.
- Wenn die Anzahl der Systemgruppen pro Seite zwischen einer und mehreren wechselt, wird dies üblicherweise mit einem Trennzeichen zwichen den Systemgruppen angezeigt. Siehe [Systeme trennen], Seite 168.
- Zu Details für andere Eigenschaften der Seitenformatierung siehe Abschnitt 4.1 [Seitenlayout], Seite 461.

Dynamikzeichen werden in den Grundeinstellungen unter dem System notiert, aber in Chormusik werden sie oft über dem System gesetzt um nicht mit dem Gesangstext zu kollidieren. Der vordefiniert Befehl `\dynamicUp` erledigt das für einen `Voice`-Kontext auf einem eigenen Notensystem. Wenn mehrere `Voice`-Kontexte vorhanden sind, müsste man den Befehl in jedem einzeln setzen. Um alle Dynamikzeichen in einer Partitur über den Systemen zu setzen, kann eine erweiterte Form genutzt werden, wie das Beispiel zeigt:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice {
        \relative c'' { g4\f g g g }
      }
    }
    \new Staff {
      \new Voice {
        \relative c' { d4 d d\p d }
      }
    }
  >>
```

```
  \layout {
    \context {
      \Score
      \override DynamicText #'direction = #UP
      \override DynamicLineSpanner #'direction = #UP
    }
  }
}
```



Vordefinierte Befehle

`\dynamicUp`.

Siehe auch

Notationsreferenz: Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472, Abschnitt 4.4 [Vertikale Abstände], Seite 481, Abschnitt 4.6.1 [Abstände anzeigen lassen], Seite 509, Abschnitt 4.6.2 [Abstände verändern], Seite 511, Abschnitt 4.2 [Partiturlayout], Seite 471, Abschnitt 4.3.7 [Eine zusätzliche Stimme für Umbrüche benutzen], Seite 479, Abschnitt 4.1 [Seitenlayout], Seite 461, [Systeme trennen], Seite 168, Abschnitt 4.6 [Die Musik auf weniger Seiten zwingen], Seite 509.

Referenz der Interna: Abschnitt “VerticalAxisGroup” in *Referenz der Interna*, Abschnitt “StaffGrouper” in *Referenz der Interna*.

Geteilte Stimmen

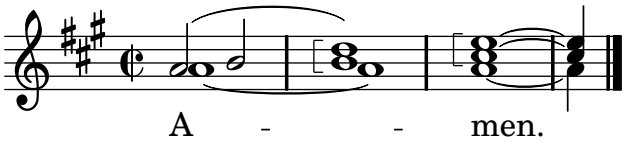
Arpeggio-Klammern benutzen um geteilte Stimmen besser sichtbar zu machen

Das `arpeggioBracket`-Klammerobjekt kann benutzt werden, um geteilte Stimmen anzuzeigen, wenn keine Hälse diese Information bieten. Das sieht man oft in Chormusik.

```
\include "english.ly"

\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
          <b d>1\arpeggio)
          <cs e>\arpeggio ~
          <cs e>4
        }
        \addlyrics { \lyricmode { A -- men. } }
```

```
      >>
      \new Voice = "lower"
      { \voiceTwo
        a1 ~
        a
        a ~
        a4 \bar "|."
      }
    >>
  }
  \layout { ragged-right = ##t }
}
```



Siehe auch

Notationsreferenz: Abschnitt 1.3.3 [Ausdrucksbezeichnungen als Linien], Seite 122.

2.1.6 Oper und Musical

Noten, Text und Dialoge von Oper und Singspielen werden normalerweise auf eine der folgenden Weisen notiert:

- Eine *Aufführungspartitur* enthält alle Orchester- und Gesangsstimmen sowie Libretto-Stichworte der gesprochenen Abschnitte.
- *Orchesterstimmen* enthalten die Noten für einzelne Instrumente des Orchesters oder der Band.
- Ein *Klavierauszug* enthält alle Gesangsstimmen mit Klavierbegleitung. Die Begleitung ist normalerweise ein Auszug der Orchesterstimmen, in dem often das originale Instrument bezeichnet ist. Klavierauszüge enthalten teilweise auch Regieanweisungen und Libretto-Stichworte.
- Ein *Vocal Book* enthält nur die Gesangstimmen (ohne Begleitung), teilweise zusammen mit dem Libretto.
- Ein *Libretto* enthält die ausführlichen gesprochenen Abschnitte, wie man sie oft in Musicals oder Operetten findet, sowie den gesungenen Text. Normalerweise sind auch Regieanweisungen enthalten. LilyPond kann eingesetzt werden, um Libretti zu setzen, aber es kann günstiger sein, dafür eine andere Methode zu benutzen, da sie keine Noten enthalten.

Die Abschnitte der LilyPond-Dokumentation, die nützlich zum Setzen von Oper und Musical sind, sind in den Verweisen unten aufgezeigt. Darauf folgen einige Abschnitte, die bestimmte Techniken behandeln, die besonders spezifisch für das Setzen von Singspiel-Partituren sind.

Verweise für Oper und Musical

- Eine große Partitur hat viele gruppierte Systeme und Gesangstext. Arten, Notensysteme zu gruppieren, finden sich in [Systeme gruppieren], Seite 162. Wie Gruppen geschachtelt werden, findet sich in [Verschachtelte Notensysteme], Seite 166.
- Die Ausgabe von leeren Notensystemen in Paritur und Klavierauszug wird oft verhindert. Um eine komprimierte Paritur ohne leere Systeme zu erstellen siehe [Systeme verstecken], Seite 175.

- Wie Orchetserstimmen notiert werden, ist dokumentiert in Abschnitt 1.6.3 [Orchesterstimmen erstellen], Seite 178. Andere Abschnitte des Kapitels „Spezielle Notation“ können auch relevant sein, abhängig von der benutzten Orchestration. Viele Instrumente sind transponierend, siehe [Transposition von Instrumenten], Seite 22.
- Wenn die Anzahl der Notensystemgruppen pro Seite sich ändert, wird normalerweise zwischen zwei Systemgruppen ein Trenner gesetzt. Siehe [Systeme trennen], Seite 168.
- Zu Einzelheiten der Seitenformatierung siehe Abschnitt 4.1 [Seitenlayout], Seite 461.
- Stichworte der Dialoge und Regieanweisungen können mit Textbeschriftungen eingefügt werden. Siehe Abschnitt 1.8 [Text], Seite 201. Ausführliche Regieanweisungen können mit Abschnitten von alleinstehenden Textbeschriftungen zwischen zwei `\score`-Umgebungen gesetzt werden. Siehe [Separater Text], Seite 208.

Siehe auch

Glossar: Abschnitt “Frenched score” in *Glossar*, Abschnitt “Frenched staves” in *Glossar*, Abschnitt “transposing instrument” in *Glossar*.

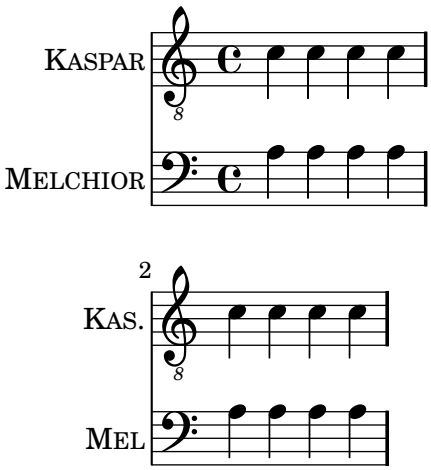
Notationsreferenz: [Systeme gruppieren], Seite 162, [Systeme verstecken], Seite 175, [Transposition von Instrumenten], Seite 22, [Verschachtelte Notensysteme], Seite 166, Abschnitt 4.1 [Seitenlayout], Seite 461, [Systeme trennen], Seite 168, [Transposition], Seite 10, Abschnitt 1.6.3 [Orchesterstimmen erstellen], Seite 178, Abschnitt 1.8.1 [Text eingeben], Seite 201.

Schnipsel: Abschnitt “Vocal music” in *Schnipsel*.

Namen von Figuren
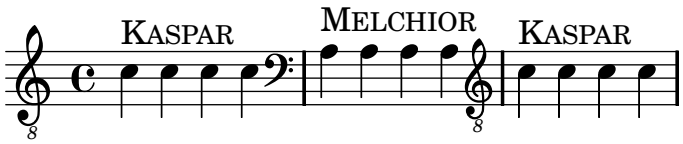
Namen von Figuren werden normalerweise links des Notensystems angezeigt, wenn auf dem System nur die Stimme einer Figure notiert ist:

```
\score {
  <<
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Kaspar
      \set Staff.shortVocalName = \markup \smallCaps Kas.
      \relative c' {
        \clef "G_8"
        c4 c c c
        \break
        c4 c c c
      }
    }
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Melchior
      \set Staff.shortVocalName = \markup \smallCaps Mel
      \clef "bass"
      \relative c' {
        a4 a a a
        a4 a a a
      }
    }
  >>
}
```



Wenn zwei oder mehr Figuren sich ein System teilen, wird der Name normalerweise über dem System immer dann gesetzt, wenn der kommende Abschnitt von der Figur gesungen werden soll. Das kann man mit Textbeschriftungen vornehmen. Oft wird eine bestimmte Schriftart hierfür benutzt.

```
\clef "G_8"
c4^\markup \fontsize #1 \smallCaps Kaspar
c c c
\clef "bass"
a4^\markup \fontsize #1 \smallCaps Melchior
a a a
\clef "G_8"
c4^\markup \fontsize #1 \smallCaps Kaspar
c c c
```
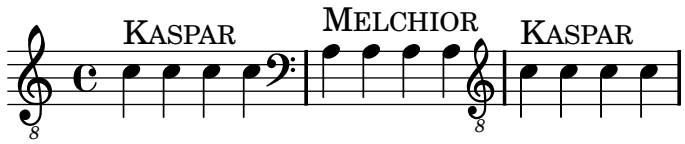


Wenn sehr viele Figurenwechsel vorkommen, kann es auch einfacher sein, „Instrument“-Definitionen für jeden Namen auf oberster Dateiebene zu definieren, sodass `\instrumentSwitch` der Wechsel der Figur angezeigt werden kann.

```
\addInstrumentDefinition #"kaspar"
  #`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
     (shortInstrumentName . "Kas.")
     (clefGlyph . "clefs.G")
     (clefOctavation . -7)
     (middleCPosition . 1)
     (clefPosition . -2)
     (instrumentCueName . ,(markup #:fontsize 1 #:smallCaps "Kaspar"))
     (midiInstrument . "voice oohs"))

\addInstrumentDefinition #"melchior"
  #`((instrumentTransposition . ,(ly:make-pitch 0 0 0))
     (shortInstrumentName . "Mel.")
     (clefGlyph . "clefs.F")
     (clefOctavation . 0)
     (middleCPosition . 6)
     (clefPosition . 2)
     (instrumentCueName . ,(markup #:fontsize 1 #:smallCaps "Melchior"))
```

```
    (midiInstrument . "voice aahs"))

\relative c' {
  \instrumentSwitch "kaspar"
  c4 c c c
  \instrumentSwitch "melchior"
  a4 a a a
  \instrumentSwitch "kaspar"
  c4 c c c
}
```



Siehe auch

Notationsreferenz: [Instrumentenbezeichnungen], Seite 178, Abschnitt A.18 [Scheme-Funktionen], Seite 680, Abschnitt 1.8 [Text], Seite 201, Abschnitt A.9 [Textbeschriftungsbefehle], Seite 597.

LilyPond Erweitern: Abschnitt “Beschriftungskonstruktionen in Scheme” in *Extending*.

Musikalische Stichnoten

Stichnoten können in Klavierauszügen, Vocal Books und Orchesterstimmen eingesetzt werdne, um anzuziegen, was für Noten eine andere Stimme direkt vor dem eigenen Einsatz spielt. Stichnoten und Instrumentenbezeichnungen werden auch im Klavierauszug eingesetzt, um anzuzeigen, welches Instrument in der Orchesterfassung den Abschnitt spielt. Das hilft dem Dirigenten, wenn man keine große Partitur parat hat.

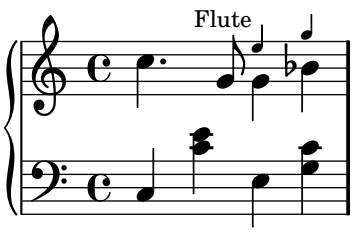
Der grundlegende Mechanismus, um Stichnoten einzufügen, findet sich in [Stichnoten], Seite 181 und [Stichnoten formatieren], Seite 185. Aber wenn man viele Stichnoten etwa in einen Klavierauszug einfügen will, um dem Dirigenten zu helfen, muss man sehr sorgfältig mit der Positionierung der Instrumentenbezeichnungen sein. Im folgenden Beispiel gibt es dazu einige Hilfestellungen.

```
flute = \relative c'' {
  s4 s4 e g
}
\addQuote "flute" { \flute }

pianoRH = \relative c'' {
  c4. g8
  % position name of cue-ing instrument just before the cue notes,
  % and above the staff
  s1*0^\markup { \right-align { \tiny "Flute" } }
  \cueDuring "flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  \new PianoStaff <<
    \new Staff {
```

```
      \pianoRH
    }
    \new Staff {
      \clef "bass"
      \pianoLH
    }
  >>
}
```



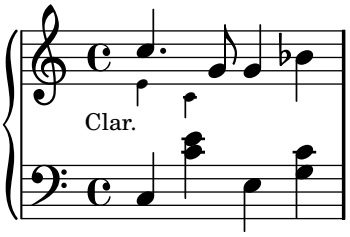
Wenn ein transponierendes Instrument zitiert wird, sollte die Orchsterstimme die Tonart angeben, damit die Transposition der Stichnoten automatisch geschehen kann. Das Beispiel unten zeigt, wie man das vornimmt. Die Noten im Beipiel sind sehr tief auf dem System, sodass `#DOWN` (nach unten) im `\cueDuring` definiert ist, damit die Hälse nach unten zeigen. Die Instrumentbezeichnung wird auch unter dem System platziert. Auch die Stimme für die rechte Hand des Klaviers ist explizit definiert. Das ist wichtig, weil die Stichnoten dieses Beispiels direkt am Anfang des ersten Taktes beginnen und sonst die gesamte Rechte Hand der Klaviernoten als `CueVoice`-(Stichnoten)-Kontext definiert werden würde!

```
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % position name of cue-ing instrument below the staff
  s1*0_\markup { \right-align { \tiny "Clar." } }
  \cueDuring "clarinet" #DOWN { c4. g8 }
  g4 bes4
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
```

```
    >>
  >>
}
```



Aus diesen zwei Beispielen wird klar, dass es sehr viele Probleme bereiten kann, ausgiebig Stichnoten in einen Klavierauszug einzufügen, und die Noten für das Klavier würden unleserlich. Im folgenden Schnipsel wird jedoch gezeigt, wie man eine Musikfunktion definiert, die Tipparbeit erspart und die Klaviernoten klarer macht.

Ausgewählte Schnipsel

Orchesterstichnoten zu einem Klavierauszug hinzufügen

Hier wird gezeigt, wie man das Hinzufügen von vielen Orchsterstichnoten zu einem Klavierauszug hinzufügen kann. Die musikalische Funktion `\cueWhile` braucht vier Argumente: Die Noten, von denen die Stichnoten formatiert werden sollen, definiert durch `\addQuote`, die Bezeichnung, die mit den Noten angegeben werden soll, dann entweder `#UP` (hoch) oder `#DOWN` (runter) zur Angabe von entweder `\voiceOne` mit der Bezeichnung über dem System oder `\voiceTwo` mit der Bezeichnung unter dem System, und schließlich die Klaviermusik, die parallel zu den Stichnoten gespielt werden soll. Die Bezeichnung des Stichnoteninstruments wird links der Stichnoten positioniert. Viele Abschnitte können zitiert werden, aber sie dürfen sich nicht überlappen.

```
cueWhile =
#(define-music-function
   (parser location instrument name dir music)
   (string? string? ly:dir? ly:music?)
   #{
     \cueDuring $instrument #$dir {
       \once \override TextScript #'self-alignment-X = #RIGHT
       \once \override TextScript #'direction = $dir
       s1*0-\markup { \tiny $name }
       $music
     }
   #})
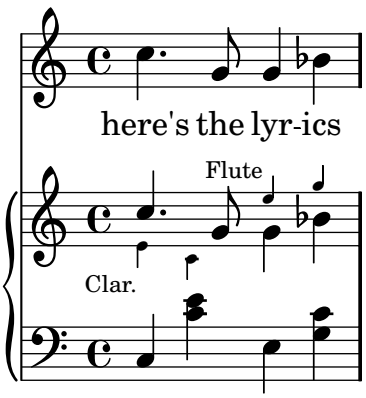
flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
\addQuote "flute" { \flute }

clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }
```

```
singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



Siehe auch

Glossar: Abschnitt “cue-notes” in *Glossar*.

Notationsreferenz: ⟨undefined⟩ [Aligning objects], Seite ⟨undefined⟩, ⟨undefined⟩ [Direction and placement], Seite ⟨undefined⟩, ⟨undefined⟩ [Formatting cue notes], Seite ⟨undefined⟩, ⟨unde-

fined⟩ [Quoting other voices], Seite ⟨undefined⟩, ⟨undefined⟩ [Using music functions], Seite ⟨undefined⟩.

Schnipsel: Abschnitt “Vocal music” in *Schnipsel*.

Referenz der Interna: Abschnitt “InstrumentSwitch” in *Referenz der Interna*, Abschnitt “CueVoice” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

`\cueDuring` fügt automatisch einen `CueVoice`-Kontext ein, in dem alle Noten platziert werden. Das heißt, dass es nicht möglich ist, überlappende Abschnitte an Stichnoten mit dieser Technik zu haben. Überlappende Abschnitt könnten notiert werden, indem man explizit unterschiedliche `CueVoice`-Kontexte definiert und mit `\quoteDuring` die Noten ausschneidet und als Stichnoten einfügt.
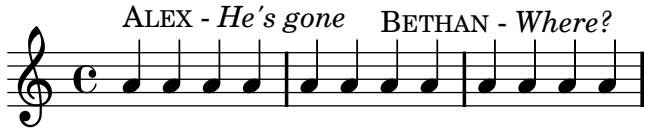
Gesprochene Musik

Effekte wie „Parlato“ bzw. „Sprechgesang“ erfordern, dass die Noten ohne Tonhöhe, aber mit dem notierten Rhythmus gesprochen werden. Solche Noten werden mit einem Kreuz als Notenkopf notiert, siehe hierzu [Besondere Notenköpfe], Seite 34.

Dialog zur Musik

Dialoge zur Musik wird üblicherweise über den Notensystemen gesetzt, meistens in kursiver Schrift, wobei der Beginn der Phrasen mit einem musikalischen Moment verklammert ist.

```
a4^\markup { \smallCaps { Alex - } \italic { He's gone } } a a a
a4 a a^\markup { \smallCaps { Bethan - } \italic Where? } a
a4 a a a
```



Für längere Abschnitt kann es nötig sein, die Noten zu dehnen, damit die Wörter besser passen. Es gibt keine Möglichkeit, das vollautomatisch von LilyPond erledigen zu lassen, und einige manuelle Änderungen am Seitenlayout sind nötig.

Für lange Phrasen und Passagen mit viel dicht gepackten Dialogen hilft es, einen `Lyrics`-Kontext zu benutzen. Der Kontext sollte nicht mit einer Stimme verknüpft sein, sondern jeder Abschnitt des Dialogs sollte eine spezifische Dauer haben. Wenn es eine Pause im Dialog gibt, sollte das letzte Wort vom Rest getrennt werden und die Dauer zwischen ihnen aufgeteilt werdne, sodass die Noten darunter sich gut verteilen.

Wenn sich der Dialog über mehr als eine Zeile erstreckt, ist es nötig, manuell Umbrüche mit `\break` einzufügen und die Platzierung des Dialogs anzupassen, damit er nicht in den rechten Seitenrand läuft. Das letzte Wort des letzten Taktes einer Zeile sollte wie oben erklärt getrennt werden.

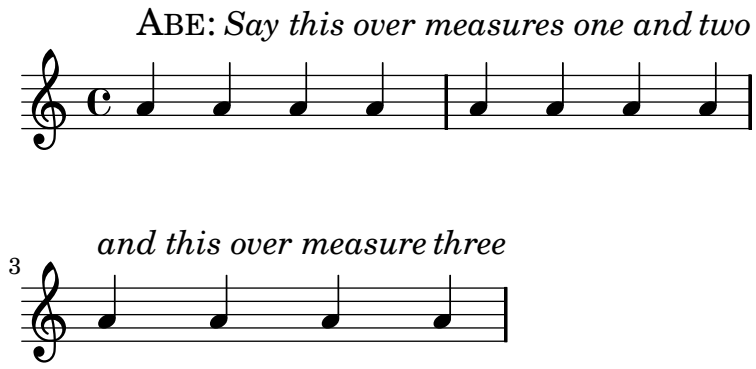
Hier ein Beispiel, das zeigt, wie das gesetzt werden kann.

```
music = \relative c'' {
  \repeat unfold 3 { a4 a a a }
}

dialogue = \lyricmode {
  \markup {
    \fontsize #1 \upright \smallCaps Abe:
```

```
    "Say this over measures one and"
  }4*7
  "two"4 |
  \break
  "and this over measure"4*3
  "three"4 |
}

\score {
  <<
    \new Lyrics \with {
      \override LyricText #'font-shape = #'italic
      \override LyricText #'self-alignment-X = #LEFT
    }
    { \dialogue }
    \new Staff {
      \new Voice { \music }
    }
  >>
}
```



Siehe auch

Notationsreferenz: [Manuelle Silbendauer], Seite 233, Abschnitt 1.8 [Text], Seite 201.

Referenz der Interna: Abschnitt “LyricText” in *Referenz der Interna*.

2.1.7 Psalmengesänge und Hymnen

Noten und Text für Psalmengesänge, Hymnen und Kirchengesänge haben eine spezifische Form in jeder Kirche. Auch wenn die Form sich unterscheidet, sind jedoch die typographischen Probleme sehr ähnlich und werden hier gesammelt behandelt.

Verweise für Psalmen und Hymnen

Wie der Gregorianische Choral in verschiedenen alten Notationsstilen gesetzt wird, findet sich in Abschnitt 2.9 [Notation von alter Musik], Seite 391.

Siehe auch

Notationreferenz: Abschnitt 2.9 [Notation von alter Musik], Seite 391.

Schnkpsel: Abschnitt “Vocal music” in *Schnipsel*.
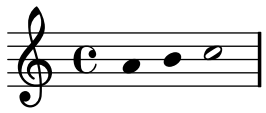
Kirchengesang notieren

Moderne Kirchengesänge benutzen eine Notation mit einer wechselnden Anzahl von Notationselementen der Notation alter Musik. Einige dieser Elemente und Methoden werden hier vorgestellt.

Kirchengesänge werden oft mit Viertelnoten ohne Hälse notiert, um die Tonhöhen darzustellen, während der Rhythmus sich am Rhythmus der gesprochenen Worte orientiert.
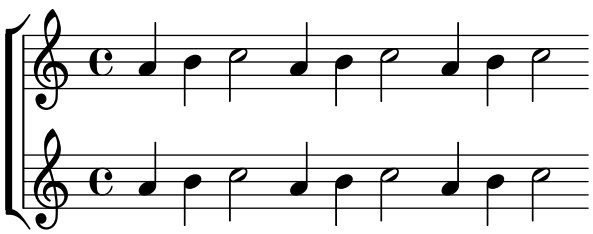
```
stemOff = { \override Staff.Stem #'transparent = ##t }

\relative c' {
  \stemOff
  a'4 b c2 |
}
```
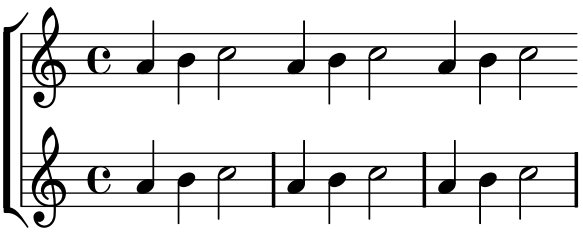


Kirchengesänge verzichten üblicherweise auf die Taktstriche oder setzen gekürzte oder punktierte Taktstriche ein. Um Taktstriche auszulassen, kann der `Bar_engraver` entfert werden.

```
\score {
  \new StaffGroup <<
    \new Staff {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
  \layout {
    \context {
      \Staff
      \remove Bar_engraver
    }
  }
}
```
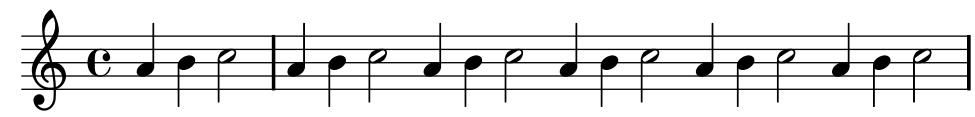


Taktstriche können auf nur für ein System entfernt werden:
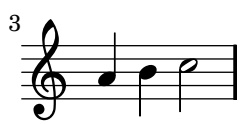
```
\score {
  \new ChoirStaff <<
    \new Staff
    \with { \remove Bar_engraver } {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative c'' {
        a4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
}
```



Um Taktstriche nur von einem Abschnitt zu entfernen, kann die Musik als Kadenz notiert werdne. Wenn der Abschnitt lang ist, müssen unsichtbare Taktstriche mit `\bar ""` eingefügt werden, wo Zeilenumbrüche stattfinden sollen.
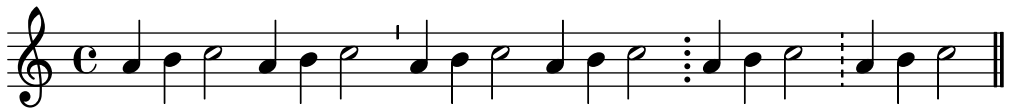
```
a4 b c2 |
\cadenzaOn
a4 b c2
a4 b c2
\bar ""
a4 b c2
a4 b c2
\cadenzaOff
a4 b c2 |
a4 b c2 |
```





Pausen können als modifizierte Taktlinien notiert werden:

```
a4
\cadenzaOn
b c2
a4 b c2
\bar "'"
a4 b c2
a4 b c2
\bar ":"
a4 b c2
\bar "dashed"
a4 b c2
\bar "||"
```
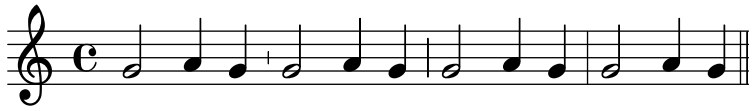


Alternativ werden die Pausenzeichen der Notation des Gregorianischen Chorals eingesetzt, obwohl die Noten selber modern sind. Das erreicht man durch Veränderung des `\breathe`-Zeichens:

```
divisioMinima = {
  \once \override BreathingSign  #'stencil = #ly:breathing-sign::divisio-minima
  \once \override BreathingSign  #'Y-offset = #0
  \breathe
}
divisioMaior = {
  \once \override BreathingSign  #'stencil = #ly:breathing-sign::divisio-maior
  \once \override BreathingSign  #'Y-offset = #0
  \breathe
}
divisioMaxima = {
  \once \override BreathingSign  #'stencil = #ly:breathing-sign::divisio-maxima
  \once \override BreathingSign  #'Y-offset = #0
  \breathe
}
finalis = {
  \once \override BreathingSign  #'stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign  #'Y-offset = #0
  \breathe
}
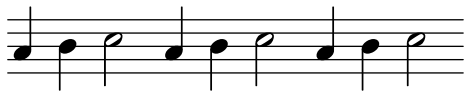
\score {
  \relative c'' {
    g2 a4 g
    \divisioMinima
    g2 a4 g
    \divisioMaior
    g2 a4 g
    \divisioMaxima
    g2 a4 g
    \finalis
  }
```

```
  \layout {
    \context {
      \Staff
      \remove Bar_engraver
    }
  }
}
```



Im Choral wird oft die Taktangabe und teilweise auch der Schlüssel weggelassen.

```
\score {
  \new Staff {
    \relative c'' {
      a4 b c2 |
      a4 b c2 |
      a4 b c2 |
    }
  }
  \layout {
    \context {
      \Staff
      \remove Bar_engraver
      \remove Time_signature_engraver
      \remove Clef_engraver
    }
  }
}
```



Gesänge für Psalmen der Anglikanischen Kirche werden normalerweise entweder *einfach*, mit 7 Takten, oder *doppelt*, mit zwei 7-Takt-Phrasen, notiert. Jede Siebener-Gruppe ist in zwei Hälften geteilt, die den Hälften jeder Strophe entsprechen, normalerweise durch eine Doppellinie getrennt. Nur halbe und ganze Noten werden genutzt. Der erste Takt jeder Hälfte hat immer eine ganze Note. Das ist der „Rezitationston“. Gesänge werden üblicherweise auf der Seite zentriert.

```
SopranoMusic = \relative g' {
  g1 | c2 b | a1 | \bar "||"
  a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative c' {
  e1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

TenorMusic = \relative a {
  c1 | c2 c | c1 |
```

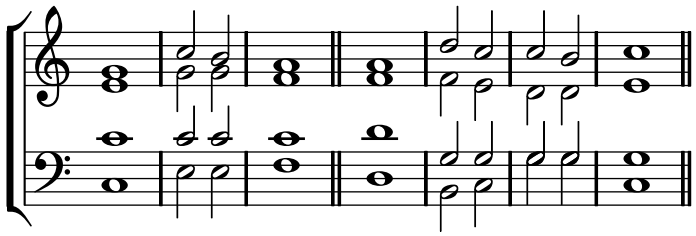
```
  d1 | g,2 g | g g | g1 |
}

BassMusic =  \relative c {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
}

% Use markup to center the chant on the page
\markup {
  \fill-line {
    \score {  % centered
      <<
        \new ChoirStaff <<
          \new Staff <<
            \global
            \clef "treble"
            \new Voice = "Soprano" <<
              \voiceOne
              \SopranoMusic
            >>
            \new Voice = "Alto" <<
              \voiceTwo
              \AltoMusic
            >>
          >>
          \new Staff <<
            \clef "bass"
            \global
            \new Voice = "Tenor" <<
              \voiceOne
              \TenorMusic
            >>
            \new Voice = "Bass" <<
              \voiceTwo
              \BassMusic
            >>
          >>
        >>
      >>
      \layout {
        \context {
          \Score
          \override SpacingSpanner
          #'base-shortest-duration = #(ly:make-moment 1 2)
        }
        \context {
          \Staff
```

```
          \remove "Time_signature_engraver"
        }
      }
    }  % End score
  }
}  % End markup
```



Einige andere Herangehensweisen derartige Gesänge zu notieren, finden sich im ersten der folgenden Schnipsel.

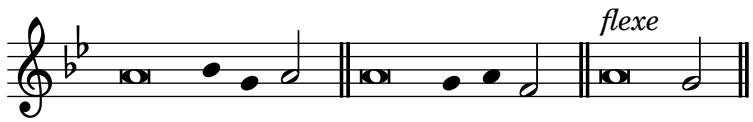
Ausgewählte Schnipsel

Psalmennotation

Diese Form der Notation wird benutzt für die Notation von Psalmen, in denen die Strophen nicht die gleiche Länge haben.

```
stemOn = { \revert Staff.Stem #'transparent }
stemOff = { \override Staff.Stem #'transparent = ##t }

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve^\markup { \italic flexe }
    \stemOn g'2 \bar "||"
  }
}
```



Cantica und andere liturgische Texte können freier gesetzt werdne, und können auch Elementer der Notation alter Musik benutzen. Oft weren die Worte unter den Noten und an ihnen ausgerichtet gesetzt. In diesem Fall werden die Noten entsprechend der Ausdehnung des Textes und nicht der Notenlänge gesetzt.

Vorlage für Alte Notation – moderne Transkription des gregorianischen Chorals
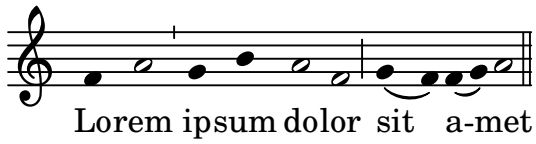
Dieses Beispiel zeigt eine moderne Transkription des Gregorianischen Chorals. Hier gibt es keine Takte, keine Notenhälse und es werden nur halbe und Viertelnoten verwendet. Zusätzliche Zeichen zeigen die Länge von Pausen an.

```
\include "gregorian.ly"

chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \override Stem #'transparent = ##t
    }
    \context {
      \Voice
      \override Stem #'length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```



Siehe auch

Handbuch zum Lernen: Abschnitt “Sichtbarkeit und Farbe von Objekten” in *Handbuch zum Lernen*, Abschnitt “Vokalensemble” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 2.9 [Notation von alter Musik], Seite 391, Abschnitt 5.4.7 [Sichtbarkeit von Objekten], Seite 546, Abschnitt 1.2.5 [Takte], Seite 87, [Musik ohne Metrum], Seite 67, Abschnitt 5.1.4 [Umgebungs-Plugins verändern], Seite 519, Abschnitt 2.9.4 [Gregorianischen Choral setzen], Seite 403.

Einen Psalm notieren

Der Text zu einem Anglikanischen Psalm wird normalerweise in separaten Versen zentriert under den Noten gesetzt.

Einfache Gesänge (mit sieben Takten) werden für jeden Vers wiederholt. Doppelte Gesänge (mit 14 Takten) werden für jeweils zwei Verse wiederholt. Zeichen zwischen den Wörtern zeigen an, wie man sie auf die Melodie anpasst. Jeder Vers wird in zwei Hälften geteilt. Ein Doppelpunkt wird benutzt, um die Teilung anzuzeigen. Das enstpricht einem doppelten Taktstrich in den Noten. Die Worte vor dem Doppelpunkt werden zu den ersten drei Takten gesungen, die Worte nach dem Doppelpunkt zu den vier letzten Takten.

Einfache Taktstriche (oder in einigen Psalmen ein umgedrehtes Komma) werden zwischen Wörtern eingefügt, um anzuzeigen, wie die Taktstriche der Noten positioniert werden. Im Beschriftungsmodus kann ein einfacher Taktstrich mit | notiert werden.

```
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing | unto the | Lord : let }
        \line { us heartily rejoice in the | strength of | our }
        \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

Andere Symbole benötigen möglicherweise Zeichen aus den `fetaMusic`-Schriftarten. Siehe Abschnitt 1.8.3 [Schriftarten], Seite 223.

```
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing \tick unto the \tick Lord : let }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```

O come let us sing ' unto the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

Wenn in einem Takt nur eine ganze Note notiert ist, werden alle Worte dieses Taktes auf dieser Note im Sprechrhythmus gesungen. Wenn im Takt zwei Noten notiert sind, gibt es normalerweise auch nur eine oder zwei Silben. Wenn mehr Silben auf einen Takt gesungen werden sollen, wird mit einem Punkt angegeben, an welcher Stelle die Note gewechselt werden soll.

```
dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us sing \tick unto \dot the \tick Lord : let
        }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```

O come let us sing′ unto • the′ Lord : let
us heartily rejoice in the′ strength of′ our
sal′ vation.

In einigen Psaltern wird ein Asterisk benutzt, um eine Pause in einem rezitierten Abschnitt anzuzeigen, und betonte oder verlängerte Silben werden mit fettem Text angezeigt:

```
dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { Today if ye will hear his voice * }
        \line {
          \concat { \bold hard en }
          | not your | hearts : as in the pro-
        }
        \line { vocation * and as in the \bold day of tempt- | }
        \line { -ation | in the | wilderness. }
      }
    }
```

```
  }
}
```

Today if ye will hear his voice *
harden | not your | hearts : as in the pro-
vocation * and as in the **day** of tempt- |
-ation | in the | wilderness.

Andere Psalter setzen einen Akzent über die Silbe, um eine Betonung anzuzeigen:

```
tick = \markup {
  \raise #2 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us \concat {
            si \combine \tick ng
          }
          | unto the | Lord : let
        }
        \line {
          us heartily \concat {
            rejo \combine \tick ice
          }
          in the | strength of | our
        }
        \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing′ | unto the | Lord : let
us heartily rejoice′ in the | strength of | our
sal- | -vation.

Der Einsatz von Beschriftung, um den Text zu zentrieren und die Zeilen in Spalten zu formatieren, findet sich in ⟨undefined⟩ [], Seite ⟨undefined⟩.

Die meisten dieser Elemente werden in einem der beiden Strophen der „Psalm“-Vorlage demonstriert, siehe Abschnitt “Vokalensemble” in *Handbuch zum Lernen*.

Siehe auch

Handbuch zum : Abschnitt “Vokalensemble” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 1.8.3 [Schriftarten], Seite 223, Abschnitt 1.8.2 [Text formatieren], Seite 210.

Unvollständige Takte in Hymnen

Hymnen beginnen und enden oft jede Zeile der Noten mit einem unvollständigen Takt, sodass jede Notenzeile exakt mit eine Textzeile übereinstimmt. Dazu setzt man den `\partial`-Befehl zu Beginn der Musik ein und `\bar "|"` oder `\bar "||"`, um die schließende Taktlinie am Ende der Zeile zu setzen.

Hymnus-Vorlage

Dieses Beispiel zeigt eine Möglichkeit, eine Hymnusmelodie zu setzen, in der jede Zeiel mit einem Auftakt beginnt und einem unvollständigen Takt abschließt. Es zeigt auch, wie man die Strophen als allein stehenden Text unter die Noten hinzufügt.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic =  \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}
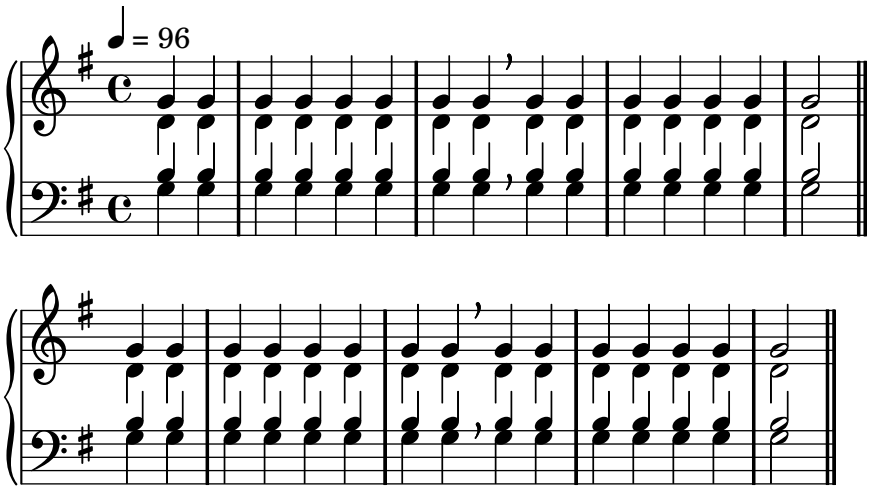
global = {
 \key g \major
}

\score {  % Start score
  <<
    \new PianoStaff <<  % Start pianostaff
      \new Staff <<  % Start Staff = RH
        \global
        \clef "treble"
        \new Voice = "Soprano" <<  % Start Voice = "Soprano"
          \Timeline
          \voiceOne
          \SopranoMusic
```

```
        >>  % End Voice = "Soprano"
        \new Voice = "Alto" <<  % Start Voice = "Alto"
          \Timeline
          \voiceTwo
          \AltoMusic
        >>  % End Voice = "Alto"
      >>  % End Staff = RH
      \new Staff <<  % Start Staff = LH
        \global
        \clef "bass"
        \new Voice = "Tenor" <<  % Start Voice = "Tenor"
          \Timeline
          \voiceOne
          \TenorMusic
        >>  % End Voice = "Tenor"
        \new Voice = "Bass" <<  % Start Voice = "Bass"
          \Timeline
          \voiceTwo
          \BassMusic
        >>  % End Voice = "Bass"
      >>  % End Staff = LH
    >>  % End pianostaff
  >>
}  % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
    ""
  }
}

\paper {  % Start paper block
  indent = 0     % don't indent first system
  line-width = 130   % shorten line length to suit music
}  % End paper block
```



This is line one of the first verse
This is line two of the same
And here's line three of the first verse
And the last line of the same
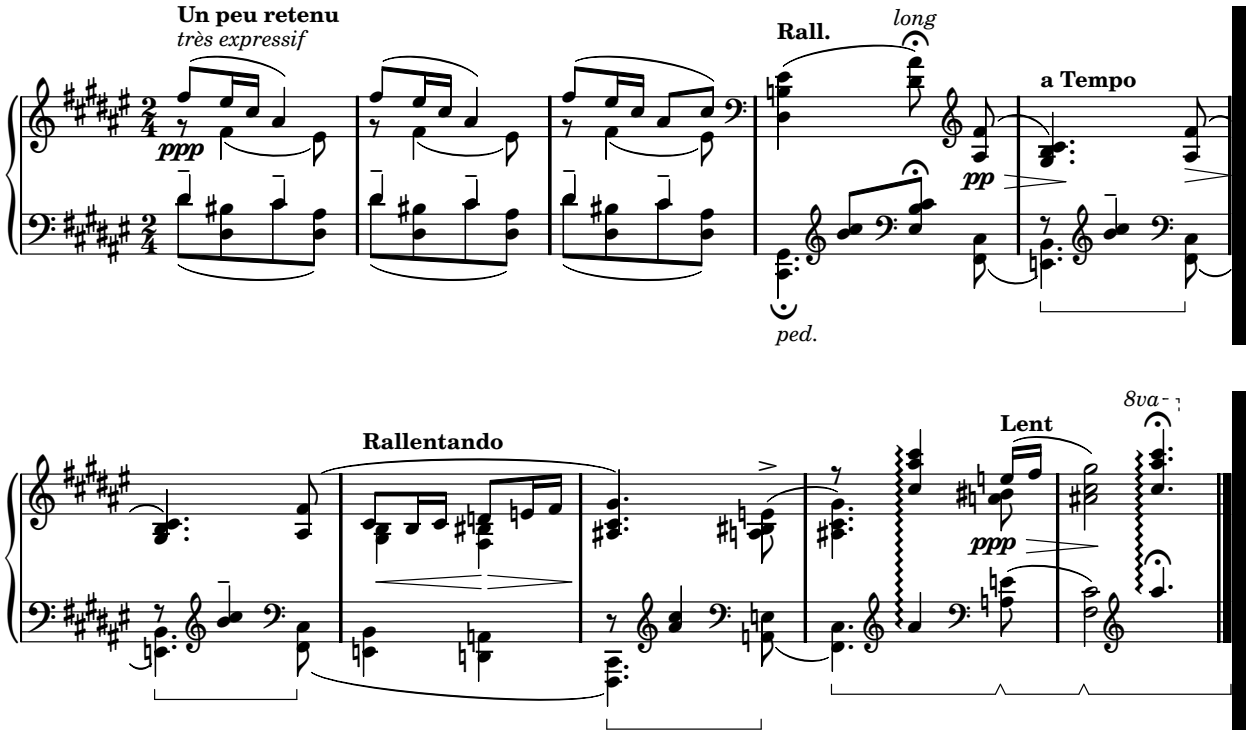
2.1.8 Alte Vokalmusik

Alte Vokalmusik ist unterstützt, wie erklärt in Abschnitt 2.9 [Notation von alter Musik], Seite 391.

Siehe auch

Notationsreferenz: Abschnitt 2.9 [Notation von alter Musik], Seite 391.

2.2 Tasteninstrumente und andere Instrumente mit mehreren Systemen



Dieser Abschnitt behandelt verschiedene Notationsaspekte, die typischerweise in Noten für Tasteninstrumente und andere Instrumente auf mehreren Notensystemen auftreten, wie etwa

Harfe und Vibraphon. Hier wird die gesamte Gruppe von Instrumenten, die auf mehreren Systemen notiert werden, als „Tasteninstrumente“ bezeichnet, auch wenn einige von ihnen keine Tasten aufweisen.

2.2.1 Übliche Notation für Tasteninstrumente

Dieser Abschnitt zeigt allgemeine Eigenschaften des Notensatzes, die für die meisten Instrumente mit mehreren Systemen benötigt werden.

Referenz für Tasteninstrumente

Tasteninstrumente werden normalerweise auf einem Klaviersystem notiert. Es besteht aus zwei Notensystemen, die durch eine Klammer verbunden sind. Die gleiche Notation wird auch für andere Tasteninstrumente sowie Harfen verwendet. Orgelmusik wird normalerweise auf zwei Systemen innerhalb eines Klaviersystems notiert, denen noch ein drittes normales Notensystem für die Pedaltöne hinzugefügt wird.

Die Systeme eines Klaviersystems sind ziemlich unabhängig, aber Stimmen können bei Bedarf zwischen den Systemen wechseln.

Einige häufige Besonderheiten von Notation für Tasteninstrumenten wird an anderen Stellen besprochen:

- Noten für Tasteninstrumente haben oft mehrere Stimmen und die Anzahl der Stimmen kann sich häufig ändern. Das ist beschrieben in [Auflösung von Zusammenstößen], Seite 149.
- Noten für Tasteninstrumente kann auch parallel, Takt für Takt notiert werden, wie gezeigt in [Musik parallel notieren], Seite 158.
- Dynamikbezeichnung kann in einem `Dynamics`-Kontext notiert werden, der zwischen zwei `Staff`-Kontexten steht und dann horizontal zwischen diesen beiden zentriert wird; siehe [Dynamik], Seite 109.
- Fingersatz wird erklärt in [Fingersatzanweisungen], Seite 191.
- Orgelpedal-Zeichen werden als Artikulationszeichen notiert, siehe Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640.
- Vertikale Rasterlinien können erstellt werden, siehe [Gitternetzlinien], Seite 198.
- Noten für Tasteninstrumente beinhalten oft *Laissez vibrer*-Bögen und Bindebögen mit Arpeggio oder Tremolo, siehe hierzu [Bindebögen], Seite 48.
- Arpeggios können auch zwischen den Systemen verbunden werden, siehe hierzu [Arpeggio], Seite 123.
- Tremolo-Zeichen finden sich in [Tremolo-Wiederholung], Seite 140.
- Viele der Optimierungen, die für Tastenmusik nötig sein können, sind demonstriert in Abschnitt “Beispiele aus dem Leben” in *Handbuch zum Lernen*.
- Unsichtbare Noten können eingesetzt werden, um Überbindungen zwischen Stimmen zu setzen, siehe Abschnitt “Andere Benutzung von Optimierungen” in *Handbuch zum Lernen*.

Siehe auch

Handbuch zum Lernen: Abschnitt “Beispiele aus dem Leben” in *Handbuch zum Lernen*, Abschnitt “Andere Benutzung von Optimierungen” in *Handbuch zum Lernen*.

Notationsreferenz: [Systeme gruppieren], Seite 162, [Instrumentenbezeichnungen], Seite 178, [Auflösung von Zusammenstößen], Seite 149, [Musik parallel notieren], Seite 158, [Fingersatzanweisungen], Seite 191, Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640, [Gitternetzlinien], Seite 198, [Bindebögen], Seite 48, [Arpeggio], Seite 123, [Tremolo-Wiederholung], Seite 140.

Schnipsel: Abschnitt “Keyboards” in *Schnipsel*.

Referenz der Interna: Abschnitt “PianoStaff” in *Referenz der Interna*.

Notensysteme manuell verändern

Stimmen können mit dem Befehl

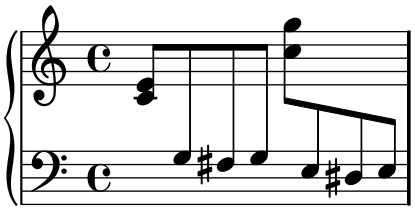
`\change Staff = *Systembezeichnung*`

manuell erzielt werden. Die Zeichenkette *Systembezeichnung* ist die Bezeichnung des Systems. Damit wird die aktuelle Stimme vom aktuellen System zu dem System mit der *Systembezeichnung* gewechselt. Üblicherweise ist die Systembezeichnung `"up"` oder `"down"`, `"RH"` oder `"LH"`.

Das System, zu dem die Stimme wechseln soll, muss zum Zeitpunkt des Wechsels existieren. Wenn notwendig, müssen Systeme „künstlich am Leben gehalten werden“, siehe Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516.

Balken zwischen den Systemen werden automatisch erstellt:

```
\new PianoStaff <<
  \new Staff = "up" {
    <e' c'>8
    \change Staff = "down"
    g8 fis g
    \change Staff = "up"
    <g'' c''>8
    \change Staff = "down"
    e8 dis e
    \change Staff = "up"
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```

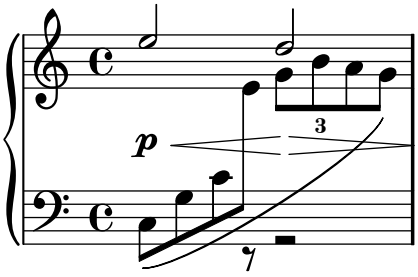


Wenn die Balken verändert werden müssen, sollte zuerst die Richtung des Balkens beeinflusst werden. Die Balkenposition wird dann von der Mitte des Systems gemessen, dass näher am Balken ist. Ein einfaches Beispiel ist gezeigt in Abschnitt “Überlappende Notation in Ordnung bringen” in *Handbuch zum Lernen*.

Bei Stimmen, die zwischen den Systemen wechseln, kann es zu überlappender Notation kommen:

```
\new PianoStaff <<
  \new Staff = "up" {
    \voiceOne
    % Make space for fingering in the cross-staff voice
    \once\override DynamicLineSpanner #'staff-padding = #3.4
    e''2\p\< d''\> s1*0\!
  }
  \new Staff = "down" <<
    {
```

```
      \clef bass
      s4. e,8\rest g,2\rest
    } \\ {
      c8\( g c'
      \change Staff = "up"
      e' g' b'-3 a' g'\)
    }
  >>
>>
```



Die Hälse und Bögen überlappen sich mit der dazwischenstehenden Dynamik-Zeile, weil die automatische Zusammenstoßauflösung für Balken, Bögen und andere Strecker, die Noten zwischen unterschiedlichen Systemen verbinden, ausgeschaltet ist. Das gilt auch für Hälse und Artikulationszeichen, wenn ihre Positionierung durch einen Strecker zwischen Systemen verändert wird. Die resultierenden Zusammenstöße müssen manuell aufgelöst werden, wo es nötig ist, dabei kann man die Methoden anwenden, die in Abschnitt "Überlappende Notation in Ordnung bringen" in *Handbuch zum Lernen* gezeigt werden.

Siehe auch

Handbuch zum Lernen: Abschnitt "Überlappende Notation in Ordnung bringen" in *Handbuch zum Lernen*.

Notationsreferenz: [Hälse], Seite 196, [Automatische Balken], Seite 74, Abschnitt 5.1.3 [Kontexte am Leben halten], Seite 516.

Schnipsel: Abschnitt "Keyboards" in *Schnipsel*.

Referenz der Interna: Abschnitt "Beam" in *Referenz der Interna*, Abschnitt "ContextChange" in *Referenz der Interna*.
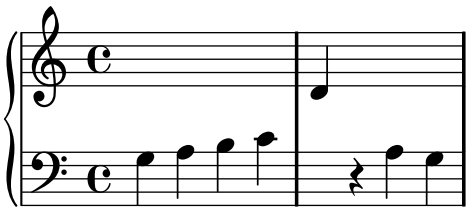
Automatischer Systemwechsel

Stimmen können angewiesen werden, automatisch zwischen dem oberen und unteren System zu wechseln. Die Syntax hierfür lautet:

`\autochange ...`*`Noten`*`...`

Damit werden zwei Notensysteme innerhalb des aktiven Klaviersystems erstellt, die „oben" (`up`) und „unten" (`down`) genannt werden. Auf dem unteren System wird als Standard der Bassschlüssel gesetzt. Der Wechsel wird automatisch basierend auf der Tonhöhe der Note vorgenommen (als Wechselpunkt gilt das eingestrichene C). Dabei wird die Richtung auch über Pausen hinweg im Voraus bestimmt.

```
\new PianoStaff {
  \autochange {
    g4 a b c'
    d'4 r a g
  }
}
```



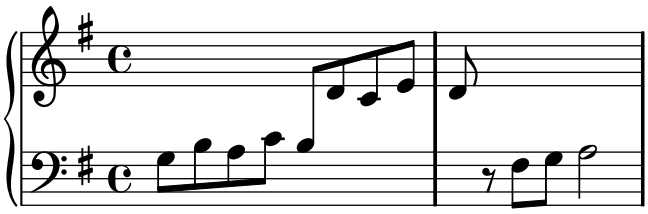
Ein `\relative`-Abschnitt, der sich außerhalb des `\autochange`-Abschnittes befindet, hat keinen Einfluss auf die Notenhöhen.

Wenn individuelle Kontrolle über die einzelnen Systeme benötigt wird, können sie manuell mit den Bezeichnungen `"up"` und `"down"` erstellt werden. Der `\autochange`-Befehl wechselt dann die Stimme zwischen den Systemen.

Achtung: Wenn Systeme manuell erstellt werden, **müssen** sie genau die Bezeichnungen `"up"` und `"down"` bekommen, damit die automatische Wechselfunktion sie erkennen kann.

Systeme müssen etwa manuell erstellt werden, damit die Tonart im unteren System gesetzt werden kann:

```
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melodieEins" {
      \key g \major
      \autochange \relative c' {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



Siehe auch

Notationsreferenz: [Notensysteme manuell verändern], Seite 289.

Schnipsel: Abschnitt “Keyboards” in *Schnipsel*.

Referenz der Interna: Abschnitt “AutoChangeMusic” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

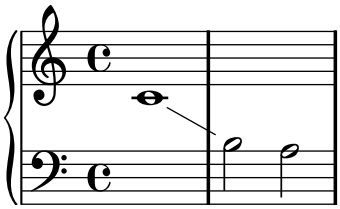
Die Aufteilung auf die Systeme geschieht nicht unbedingt an optimaler Stelle. Für bessere Qualität müssen die Wechsel manuell eingestellt werden.

Akkorde werde nicht über die Systeme verteilt, sie werden dem System zugewiesen, auf dem sich ihre erste Note befinden würde.

Stimmführungslinien

Immer, wenn eine Stimme von einem Klaviersystem zu dem anderen wechselt, kann automatisch eine Linie zur Verdeutlichung des Stimmenverlaufs ausgegeben werden:

```
\new PianoStaff <<
  \new Staff = "one" {
    \showStaffSwitch
    c1
    \change Staff = "two"
    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>
```



Vordefinierte Befehle

`\showStaffSwitch`, `\hideStaffSwitch`.

Siehe auch

Schnipsel: Abschnitt “Keyboards” in *Schnipsel*.

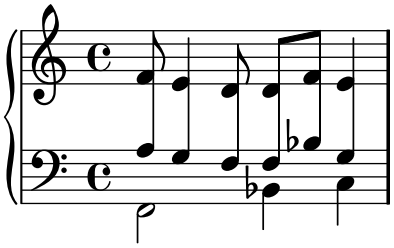
Referenz der Interna: Abschnitt “Note_head_line_engraver” in *Referenz der Interna*, Abschnitt “VoiceFollower” in *Referenz der Interna*.

Hälse über beide Systeme

Akkorde, die über zwei Systeme reichen, können erstellt werden, indem die Länge der Hälse im unteren System vergrößert wird, bis sie zum oberen System hinauf reichen bzw. umgekehrt bei Hälsen, die nach unten zeigen.

```
\new PianoStaff <<
  \new Staff {
    \relative c' {
      f8 e4 d8 d f e4
    }
  }
  \new Staff {
    \relative c' {
      << {
        \clef bass
        % stems may overlap the other staff
        \override Stem #'cross-staff = ##t
        % extend the stems to reach the other staff
        \override Stem #'length = #12
        % do not print extra flags
```

```
        \override Stem #'flag-style = #'no-flag
        % prevent beaming as needed
        a8 g4 f8 f bes\noBeam g4
      }
      \\
      {
        f,2 bes4 c
      } >>
    }
  }
>>
```



Ausgewählte Schnipsel

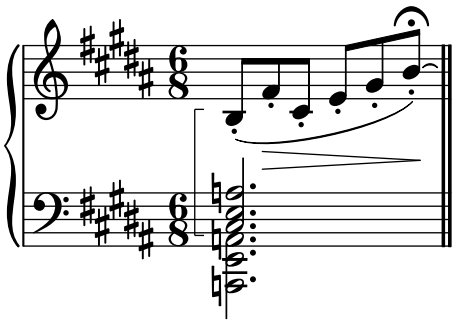
Akkorde auf zwei Systemen mit Arpeggioklammern anzeigen

Eine Arpeggioklammer kann anzeigen, dass Noten auf zwei unterschiedlichen Systemen mit der selben Hand gespielt werden sollen. Damit das notiert werden kann, muss der `PianoStaff`-Kontext so eingestellt werden, dass er Arpeggios über Systeme hinweg akzeptiert und die Form der Arpeggios muss auf eine Klammer eingestellt werden.

(Debussy, Les collines d'Anacapri, T. 65)

```
\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \override PianoStaff.Arpeggio #'stencil = #ly:arpeggio::brew-chord-bracket
  \new Staff {
    \relative c' {
      \key b \major
      \time 6/8
      b8-.(\arpeggio fis'-.\> cis-. e-. gis-. b-.)\!\fermata^\laissezVibrer
      \bar "||"
    }
  }
  \new Staff {
    \relative c' {
      \clef bass
      \key b \major
      <<
        {
          <a e cis>2.\arpeggio
        }
        \\
        {
          <a, e a,>2.
        }
      >>
```

```
    }
  }
>>
```



Siehe auch

Schnipsel: Abschnitt “Keyboards” in *Schnipsel*.

Referenz der Interna: Abschnitt “Stem” in *Referenz der Interna*.
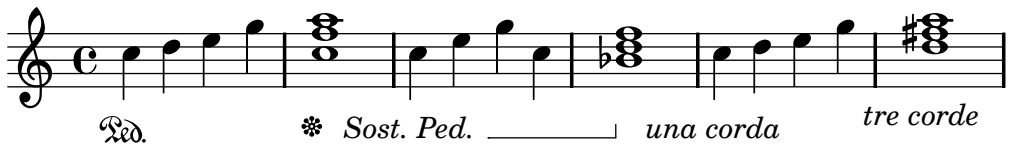
2.2.2 Klavier

Dieser Abschnitt zeigt Eigenheiten der Notation von Klavermusik

Klavierpedal

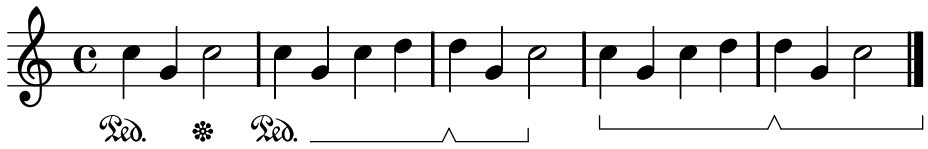
Klaviere (teilweise auch Vibraphone und Celesta) besitzen üblicherweise drei Pedale, das linke oder Haltepedal, das rechte oder Una-corda-Pedal und das Sostenuto-Pedal. Die englischen Begriff hierzu lauten: *sustain*, *sostenuto* und *una corda*.

```
c4\sustainOn d e g
<c, f a>1\sustainOff
c4\sostenutoOn e g c,
<bes d f>1\sostenutoOff
c4\unaCorda d e g
<d fis a>1\treCorde
```



Die Pedalbezeichnung kann auf drei Arten vorgenommen werden: mit Text, Klammern oder einer Mischung aus beidem. Das Haltepedal und das Una-corda-Pedal benutzen als Standard die Textdarstellung, während das Sostenuto-Pedal den gemischten Stil benutzt:

```
c4\sustainOn g c2\sustainOff
\set Staff.pedalSustainStyle = #'mixed
c4\sustainOn g c d
d\sustainOff\sustainOn g, c2\sustainOff
\set Staff.pedalSustainStyle = #'bracket
c4\sustainOn g c d
d\sustainOff\sustainOn g, c2
\bar "|."
```



Die Platzierung der Befehle entspricht der Bewegung der Pedale während des Spielens. Um das Pedal bis zur letzten Taktlinie zu halten, muss der letzte Pedal-hoch-Befehl weggelassen werden.

Pedalbezeichnungen können innerhalb eines `Dynamics`-Kontextes notiert werden, sodass sie an einer horizontalen Linie ausgerichtet werden.

Siehe auch

Notationsreferenz: [Bindebögen], Seite 48.

Schnipsel: Abschnitt “Keyboards” in *Schnipsel*.

Referenz der Interna: Abschnitt “SustainPedal” in *Referenz der Interna*, Abschnitt “SustainPedalLineSpanner” in *Referenz der Interna*, Abschnitt “SustainEvent” in *Referenz der Interna*, Abschnitt “SostenutoPedal” in *Referenz der Interna*, Abschnitt “SostenutoPedalLineSpanner” in *Referenz der Interna*, Abschnitt “SostenutoEvent” in *Referenz der Interna*, Abschnitt “UnaCordaPedal” in *Referenz der Interna*, Abschnitt “UnaCordaPedalLineSpanner” in *Referenz der Interna*, Abschnitt “UnaCordaEvent” in *Referenz der Interna*, Abschnitt “PianoPedalBracket” in *Referenz der Interna*, Abschnitt “Piano_pedal_engraver” in *Referenz der Interna*.

2.2.3 Akkordeon

Dieser Abschnitt behandelt Notation, die nur für Akkordeonmusik benötigt wird.

Diskant-Symbole

Akkordeons werden oft mit mehreren Reihen an Zungen gebaut, welche Unisono oder eine Oktave höher bzw. tiefer erklingen. Jedes Akkordeon hat eigene Bezeichnungen für die Register (engl. shift) wie etwa *Oboe*, *Bandonium* usw. Eine Anzahl an Symbolen wird benutzt um die Wechsel anzuzeigen.

Ausgewählte Schnipsel

Symbole für Akkordeon-Diskantregister

Diskantregister für Akkordeon können mit `\markup` dargestellt werden. Die vertikale Position der einzelnen Elemente werden mit `\raise` angepasst.

```
discant = \markup {
  \musicglyph #"accordion.discant"
}
dot = \markup {
  \musicglyph #"accordion.dot"
}

\layout { ragged-right = ##t }

% 16 voets register
accBasson = ^\markup {
  \combine
  \discant
  \raise #0.5 \dot
}

% een korig 8 en 16 voets register
accBandon = ^\markup {
```

```
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \raise #1.5 \dot
}

accVCello = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \translate #'(1 . 0) \raise #1.5 \dot
}

% 4-8-16 voets register
accHarmon = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \raise #2.5 \dot
}

accTrombon = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \combine
          \translate #'(1 . 0) \raise #1.5 \dot
          \translate #'(-1 . 0) \raise #1.5 \dot
}

% eenkorig 4 en 16 voets register
accOrgan = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \raise #2.5 \dot
}

accMaster = ^\markup {
  \combine
    \discant
```

```
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \combine
          \translate #'(1 . 0) \raise #1.5 \dot
          \combine
            \translate #'(-1 . 0) \raise #1.5 \dot
            \raise #2.5 \dot
}

accAccord = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \combine
        \translate #'(1 . 0) \raise #1.5 \dot
        \combine
          \translate #'(-1 . 0) \raise #1.5 \dot
          \raise #2.5 \dot
}

accMusette = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \combine
        \translate #'(1 . 0) \raise #1.5 \dot
        \translate #'(-1 . 0) \raise #1.5 \dot
}

accCeleste = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \translate #'(-1 . 0) \raise #1.5 \dot
}

accOboe = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \raise #2.5 \dot
}

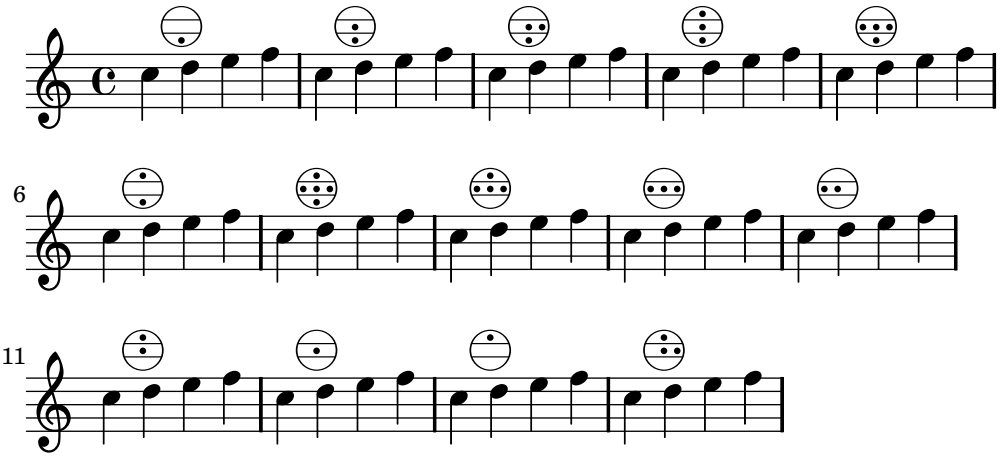
accClarin = ^\markup {
  \combine
    \discant
```

```
    \raise #1.5 \dot
}

accPiccolo = ^\markup {
    \combine
       \discant
       \raise #2.5 \dot
}

accViolin = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \combine
        \translate #'(1 . 0) \raise #1.5 \dot
        \raise #2.5 \dot
}

\relative c'' {
  c4 d\accBasson e f
  c4 d\accBandon e f
  c4 d\accVCello e f
  c4 d\accHarmon e f
  c4 d\accTrombon e f
  \break
  c4 d\accOrgan e f
  c4 d\accMaster e f
  c4 d\accAccord e f
  c4 d\accMusette e f
  c4 d\accCeleste e f
  \break
  c4 d\accOboe e f
  c4 d\accClarin e f
  c4 d\accPiccolo e f
  c4 d\accViolin e f
}
```



Siehe auch

Schnipsel: Abschnitt “Keyboards” in *Schnipsel*.

2.2.4 Harfe

Dieser Abschnitt zeigt Eigenheiten der Notation für Harfe.

Referenzen für Harfe

Einige übliche Notationseigenheiten für Harfe sind woanders behandelt:

- Glissando ist die üblichste Harfentechnik, siehe [Glissando], Seite 122.
- Ein *Bisbigliando* wird als ein Tremolo notiert, siehe [Tremolo-Wiederholung], Seite 140.
- Flageolettöne werden hier beschrieben: [Flageolett], Seite 301.
- Für Arpeggio und non-arpeggio, siehe [Arpeggio], Seite 123.
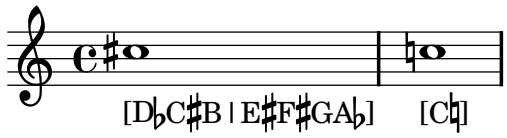
Siehe auch

Notationsreferenz: [Tremolo-Wiederholung], Seite 140, [Glissando], Seite 122, [Arpeggio], Seite 123, [Flageolett], Seite 301.

Harfenpedal

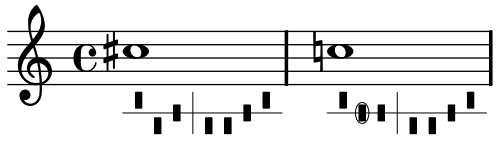
Harfe haben sieben Saiten in einer Oktave, die entweder als normaler Ton, oder aber erhöht bzw. erniedrigt klingen können. Bei einer Hakenharfe kann man jede Saite einzeln einstellen, bei Pedalharfen aber wird jede Saite mit der gleichen Notenbezeichnung von einem einzigen Pedal kontrolliert. Vom Spieler aus gesehen von rechts nach links sind die Pedale: D, C und H für die linke und E, F, G und A für die rechte Seite. Die Position des Pedals kann mit Textbeschriftungselementen:

```
\textLengthOn
cis1_\markup \concat \vcenter {
  [D \flat C \sharp B|E \sharp F \sharp G A \flat] }
c!1_\markup \concat \vcenter {
  [ C \natural ] }
```



oder Pedaldiagrammen angezeigt werden:

```
\textLengthOn
cis1_\markup { \harp-pedal #"^v-|vv-^" }
c!1_\markup { \harp-pedal #"^o--|vv-^" }
```
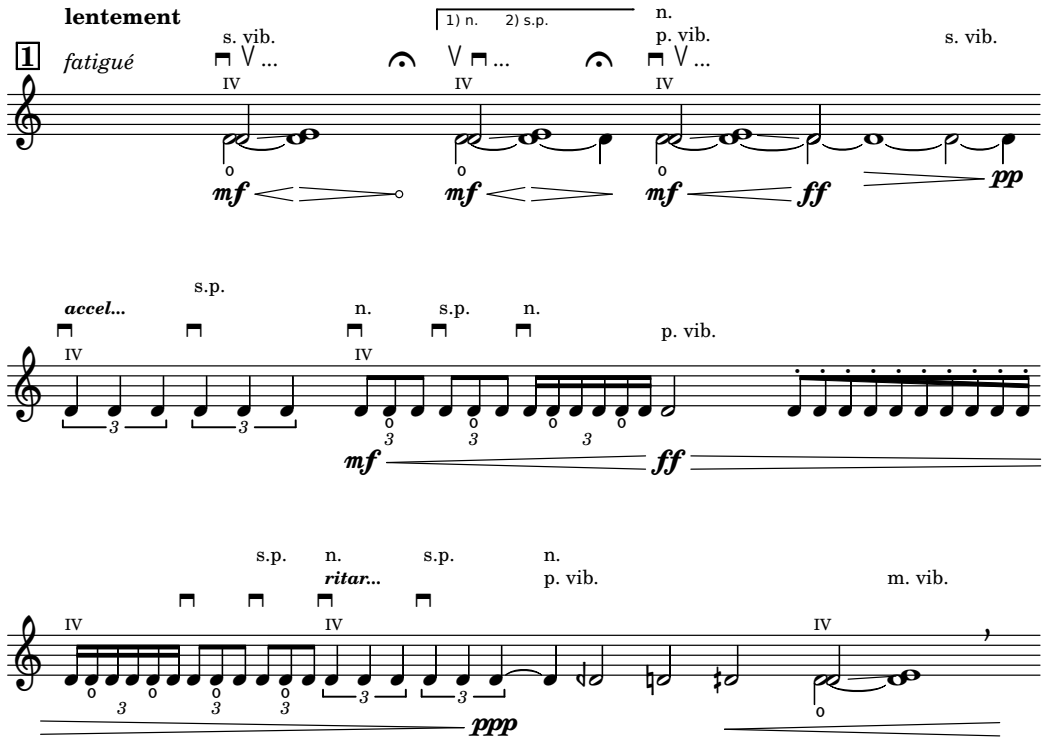


Der `\harp-pedal`-Befehl braucht eine Anzahl an Zeichen, von welchen `^` die höchste Pedalposition (erniedrigte Tonhöhe), `-` die mittlere Pedalposition (normale Tonhöhe, `v` die tiefste Pedalposition (erhöhter Ton) anzeigt. `|` ist ein Trenner. Ein `o` vor der Definition umrandet das Symbol.

Siehe auch

Notationsreferenz: [Textarten], Seite 201, Abschnitt A.9.5 [Instrument Specific Markup], Seite 630.

2.3 Bundlose Saiteninstrumente



Dieser Abschnitt stellt Information und Referenzen zur Verfügung, die beim Setzen von Noten für Saiteninstrumente ohne Bund herangezogen werden können.

2.3.1 Übliche Notation für bundlose Saiteninstrumente

Es gibt wenige Spezifikationen für die Notation von Saiteninstrumenten ohne Bünde. Die Noten werden auf einem System notiert und meistens ist auch nur eine Stimme erforderlich. Zwei Stimmen können für Doppelgriff- oder Divisi-Stellen erforderlich sein.

Hinweise für bundlose Saiteninstrumente

Die meisten Notationseigenschaften, die für Orchesterstreicher eingesetzt werden, sind an anderer Stelle beschrieben:

- Textanweisungen wie „pizz.“ oder „arco“ werden als einfacher Text eingefügt, siehe [Textarten], Seite 201.
- Fingersatz, auch das Zeichen für den Daumen, ist erklärt in [Fingersatzanweisungen], Seite 191.
- Doppelgriffe werden normalerweise als Akkord notiert, siehe hierzu [Noten mit Akkorden], Seite 142. Anweisungen, wie Akkorde gespielt werden sollen, können auch hinzugefügt werden, siehe [Arpeggio], Seite 123.
- Eine Vorlage für Streichquartett findet sich in Abschnitt “Streichquartett” in *Handbuch zum Lernen*. Andere sind als Schnipsel zur Verfügung gestellt.

Siehe auch

Handbuch zum Lernen: Abschnitt “Streichquartett” in *Handbuch zum Lernen*.

Notationsreferenz: [Textarten], Seite 201, [Fingersatzanweisungen], Seite 191, [Noten mit Akkorden], Seite 142, [Arpeggio], Seite 123.
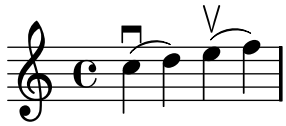
Schnipsel: Abschnitt “Unfretted strings” in *Schnipsel*.

Bezeichnung des Bogens

Hinweise zur Bogenfügung können als Artikulationen erstellt werden, wie beschrieben in [Artikulationszeichen und Verzierungen], Seite 106.

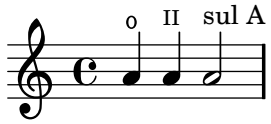
Die Befehle `\upbow` und `\downbow` werden mit Legatobögen in folgender Weise eingesetzt:

```
c4(\downbow d) e(\upbow f)
```



und das nächste Beispiel zeigt drei Arten, eine offene A-Saite auf der Geige anzuzeigen:

```
a4 \open
a^\markup { \teeny "II" }
a2^\markup { \small "sul A" }
```



Vordefinierte Befehle

`\downbow`, `\upbow`, `\open`.

Siehe auch
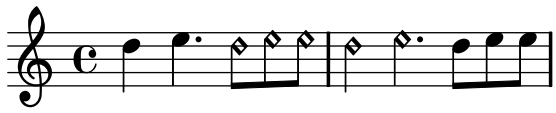
Notation Reference: [Artikulationszeichen und Verzierungen], Seite 106, [Legatobögen], Seite 116.

Flageolett

Natürliches Flageolett

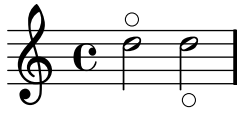
Flageolett-Töne können auf verschiedene Arten notiert werden. Üblicherweise werden sie mit einem Rautenkopf notiert, wenn ein Ton angezeigt werde, bei dem die Saite berührt wird, wo sie sonst abgegriffen würde.

```
d4 e4.
\harmonicsOn
d8 e e
d4 e4.
\harmonicsOff
d8 e e
```



Alternativ kann auch eine normale Noten die Tonhöhe anzeigen, die erklingen soll, wobei ein kleiner Kreis angibt, dass es sich um einen Flageolett-Ton handelt:
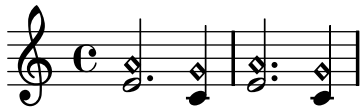
```
d2^\flageolet d_\flageolet
```



Künstliches Flageolett

Künstliche Flageoletttöne werden mit zwei Noten notiert, von denen einen einen normalen Notenkopf besitzt und die Griffposition des Fingers angibt, während die andere in Rautenform die Position des leicht aufgesetzten Fingers anzeigt.

```
<e a\harmonic>2.  <c g'\harmonic>4
\set harmonicDots = ##t
<e a\harmonic>2.  <c g'\harmonic>4
```



Achtung: `\harmonic` **muss** innerhalb einer Akkordkonstruktion gesetzt werden, auch wenn nur eine Note gesetzt wird. Normalerweise würde `\harmonicsOn` in dieser Situation benutzt.

Siehe auch

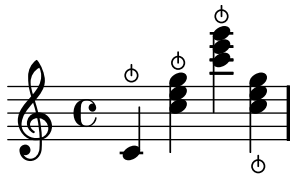
Glossar: Abschnitt “harmonics” in *Glossar*.

Notationsreferenz: [Besondere Notenköpfe], Seite 34, [Hinweise für bundlose Saiteninstrumente], Seite 300.

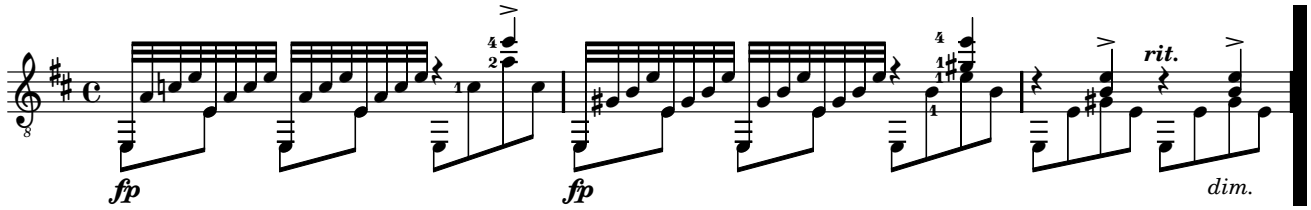
Bartók-Pizzicato
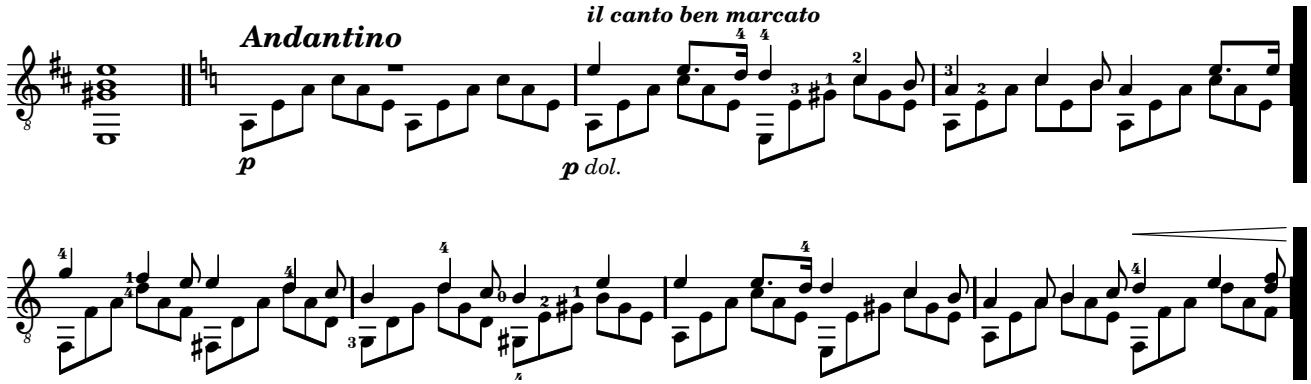
Ein Knallpizzicato, auch als Bartók-Pizzicato bekannt, ist ein hartes Pizzicato, bei dem man die Saite nach oben (und nicht seitlich) zieht, sodass sie beim Schwingen das Griffbrett berührt.

```
c4\snappizzicato
<c' e g>4\snappizzicato
<c' e g>4^\snappizzicato
<c, e g>4_\snappizzicato
```



2.4 Saiteninstrumente mit Bünden





Dieser Abschnitt erklärt bestimmte Eigenheiten der Notation für Saiteninstrumente mit Bünden.

2.4.1 Übliche Notation für Saiteninstrumente mit Bünden

Dieser Abschnitt zeigt Besonderheiten der Notation, die allen Bundinstrumenten eigen ist.

Referenz für Saiteninstrumente mit Bünden

Noten für Bundinstrumente wird normalerweise auf einem einzelnen System notiert, entweder als traditionelles Notensystem oder in Tabulaturform. Manchmal werden beide Arten miteinander verbunden, und besonders in populärer Musik ist es üblich, über dem traditionellen System Griffsymbole zu setzen. Gitarre und Banjo sind transponierende Instrumente, die eine Oktave tiefer klingen als sie notiert werden. Partituren für diese Instrumente sollten den „Tenorschlüssel“ (`"treble_8"` bzw. `\transposition c`) benutzen, um korrekte MIDI-Dateien zu erhalten. Einige Spezifika für Instrumente mit Bünden sind an anderer Stelle erklärt:

- Fingersatz kann notiert werden, siehe [Fingersatzanweisungen], Seite 191.
- Anweisungen für *Laissez vibrer*-Bögen und Bögen zwischen Arpeggios und Tremolos sind beschrieben in [Bindebögen], Seite 48.
- Hinweise, wie mehrere Stimmen gesetzt werden können, finden sich in [Auflösung von Zusammenstößen], Seite 149.
- Instructions for indicating harmonics can be found in [Flageolett], Seite 301.

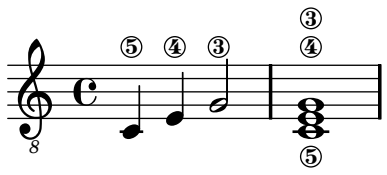
Siehe auch

Notationsreferenz: [Fingersatzanweisungen], Seite 191, [Bindebögen], Seite 48, [Auflösung von Zusammenstößen], Seite 149, [Instrumentenbezeichnungen], Seite 178, [Musik parallel notieren], Seite 158, [Arpeggio], Seite 123, Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640, [Notenschlüssel], Seite 16 [Transposition von Instrumenten], Seite 22.

Seitennummerbezeichnung

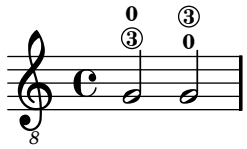
Die Nummer der Saite, auf der gespielt werden soll, kann angezeigt werden, indem `\Zahl` an eine Note innerhalb eines Akkord-Konstrukts gesetzt wird:

```
\clef "treble_8"
<c\5>4 <e\4> <g\3>2
<c,\5 e\4 g\3>1
```



Wenn Fingersatz und Saitennummer zusammen benutzt werden, wird ihre Position anhand der Reihenfolge entschieden, mit der sie im Code auftauchen:

```
\clef "treble_8"
<g\3-0>2
<g-0\3>
```

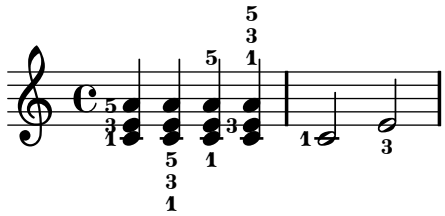


Ausgewählte Schnipsel

Position von Fingersatz in Akkorden kontrollieren

Die Position von Fingersatzzahlen kann exakt kontrolliert werden.

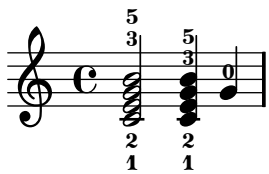
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Fingersatz auch innerhalb des Systems setzen

Normalerweise werden vertikal orientierte Fingersatzzahlen außerhalb des Systems gesetzt. Das kann aber verändert werden.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering #'staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 <g'-0>
}
```



Siehe auch

Notationsreferenz: [Fingersatzanweisungen], Seite 191.

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

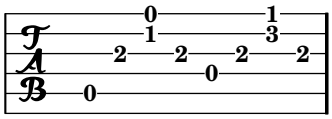
Referenz der Interna: Abschnitt “StringNumber” in *Referenz der Interna*, Abschnitt “Fingering” in *Referenz der Interna*.

Standardtabulaturen

Musik für gezupfte Saiteninstrumente wird oft notiert, indem man eine Finger/Berührungsnotation bzw. Tabulatur benutzt. Im Gegensatz zur traditionellen Notation werden hier Tonhöhen nicht mit Notenköpfen notiert, sondern mit Zahlen (oder buchstabenartigen Symbolen in historischen Tabulaturen). Die Notenlinien einer Tabulatur zeigen die Saite an, auf der eine Note gespielt werden soll, und eine Zahl auf einer Notenlinie zeigt an, welcher Bund für eine Note gespielt werden muss. Die Zahlen werden vertikal übereinander geschrieben, wenn sie gleichzeitig gespielt werden sollen.

Standardmäßig ist Saite 1 die höchste Saite und entspricht der höchsten Notenlinie des `TabStaff` (der Tabulatur). Die voreingestellte Saitenstimmung der Tabulatur ist die normale Gitarrenstimmung (mit 6 Saiten). Die Noten werden als Tabulatur ausgegeben, wenn man den `TabStaff`-Kontext und darin den `TabVoice`-Kontext benutzt. Ein kalligraphischer Tabulaturschlüssel wird automatisch hinzugefügt.

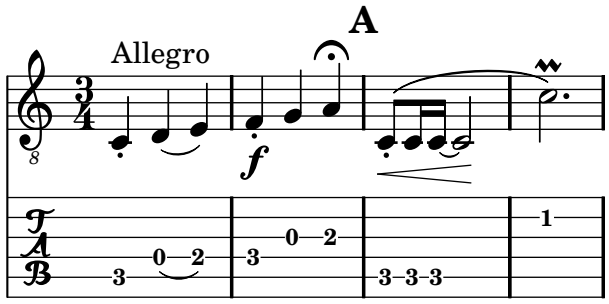
```
\new TabStaff \relative c' {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```



Standard-Tabulaturen haben weder Symbole, die Notendauern anzeigen, noch andere musikalische Symbole wie etwa Ausdrucksbezeichnungen.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ c2\!
  c'2.\prall\)
}

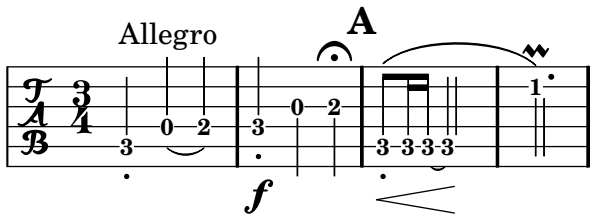
\score {
  <<
    \new Staff { \clef "G_8" \symbols }
    \new TabStaff { \symbols }
  >>
}
```



Wenn alle musikalischen Symbole, die in der traditionellen Notation eingesetzt werden, auch in der Tabulatur gedruckt werden sollen, muss man den Befehl `\tabFullNotation` in einem `TabStaff`-Kontext hinzufügen. Dabei ist zu beachten, dass halbe Noten in einer Tabulatur mit zwei Hälsen dargestellt werden, um sie von Viertelnoten zu unterscheiden.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ c2\!
  c'2.\prall\)
}

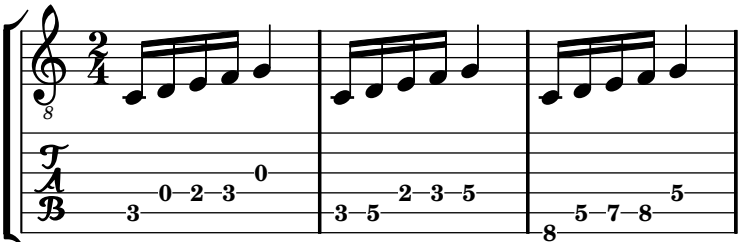
\score {
  \new TabStaff {
    \tabFullNotation
    \symbols
  }
}
```



Normalerweise werden Tonhöhen der tiefstmöglichen Spielposition auf dem Bundbrett zugewiesen (erste Lage). Offene Saiten werden automatisch bevorzugt. Wenn Sie eine bestimmte Tonhöhe auf einer bestimmten Saite gespielt haben wollen, können Sie eine Saitennummeranweisung zur Tonhöhe hinzufügen. Wenn Sie Tonhöhe und Saitenzahlanweisung nicht innerhalb einer Akkord-Konstruktion (`<>`) notieren, erscheint die Saitenzahlanweisung nicht in der traditionellen Notation. Es ist jedoch sehr viel bequemer, die Spielposition unter Benutzung von `minimumFret` zu definieren. Der Standardwert von `minimumFret` beträgt 0.

```
\new StaffGroup <<
   \new Staff \relative c {
     \clef "treble_8"
     \time 2/4
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     c,16 d e f g4
   }
   \new TabStaff \relative c {
```

```
    c16 d e f g4
    c,16\5 d\5 e\4 f\4 g4\4
    \set TabStaff.minimumFret = #5
    c,16 d e f g4
  }
>>
```

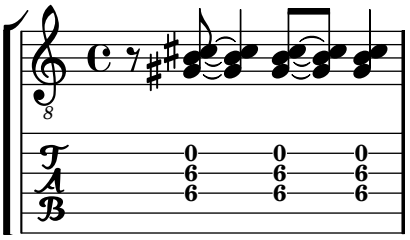


Akkord-Konstruktionen können mit dem Akkord-Wiederholungssymbol `q` wiederholt werden. Um diese Eigenschaft in einer Tabulatur benutzen zu können, gibt es den Befehl `\tabChordRepetition`. Er speichert die Saiteninformationen, die innerhalb von Akkord-Konstruktionen gegeben werden, sodass wiederholte Akkorde identische Darstellungen erhalten.

```
\tabChordRepetition

guitar = \relative c' {
  r8 <gis\4 cis\3 b\2>~ q4 q8~ q q4
}

\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \override Voice.StringNumber #'transparent = ##t
    \guitar
  }
  \new TabStaff {
    \guitar
  }
>>
```

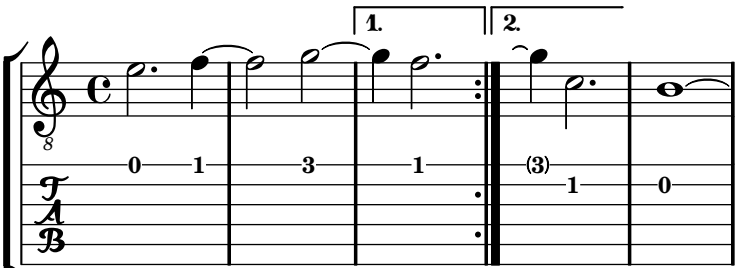


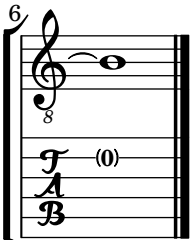
Bindestriche über einen Zeilenumbruch werden normalerweise in Klammern gesetzt. Das gilt auch für die zweite Klammer einer Wiederholung.

```
ties = \relative c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~
```

```
  }
  \alternative {
     { g4 f2. }
     { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \context Staff {
        \clef "treble_8"
        \ties
      }
      \context TabStaff {
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```



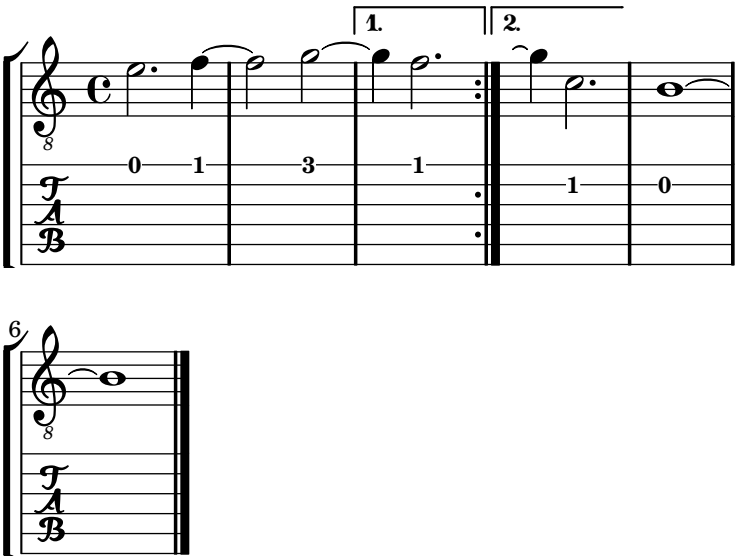


Der Befehl `\hideSplitTiedTabNotes` hebt das Verhalten auf, dass Bundnummern in Klammern gesetzt werden:

```
ties = \relative c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~ }
```

```
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

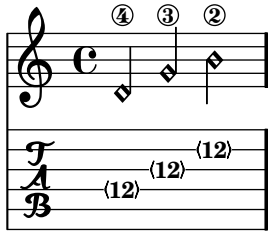
\score {
  <<
    \new StaffGroup  <<
      \context Staff {
        \clef "treble_8"
        \ties
      }
      \context TabStaff {
      \hideSplitTiedTabNotes
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```



Flageolett (engl. *harmonic*) kann zur Tabulaturnotation als klingende Tonhöhe hinzugefügt werden:

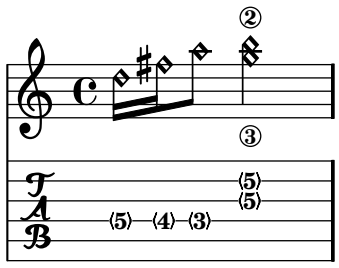
```
firstHarmonic = {
  <d'\4\harmonic>4
  <g'\3\harmonic>4
  <b'\2\harmonic>2
```

```
}
\score {
  <<
    \new Staff { \firstHarmonic }
    \new TabStaff { \firstHarmonic }
  >>
}
```



Dabei ist zu beachten, dass der Befehl `\harmonic` immer innerhalb einer Akkordkonstruktion notiert werden muss. Flageolett ist nur sinnvoll für offene Saiten im 12. Bund. Alle anderen Flageolett-Töne sollten von LilyPond errechnet werden. Das wird erreicht, indem man den Bund angibt, wo der Finger der Greifhand die Saite berühren soll.

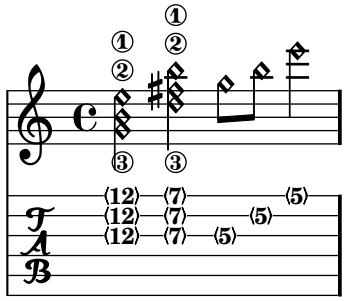
```
fretHarmonics = {
  \harmonicByFret #5 d16\4
  \harmonicByFret #4 d16\4
  \harmonicByFret #3 d8\4
  \harmonicByFret #5 <g\3 b\2>2.
}
\score {
  <<
    \new Staff { \fretHarmonics }
    \new TabStaff { \fretHarmonics }
  >>
}
```



Alternativ können Flageolett-Töne auch errechnet werden, indem man das Verhältnis der Saitenlängen über und unter dem Flageolett-Finger definiert:

```
ratioHarmonics = {
  \harmonicByRatio #1/2 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/3 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }
}
\score {
  <<
    \new Staff { \ratioHarmonics }
    \new TabStaff { \ratioHarmonics }
```

```
  >>
}
```

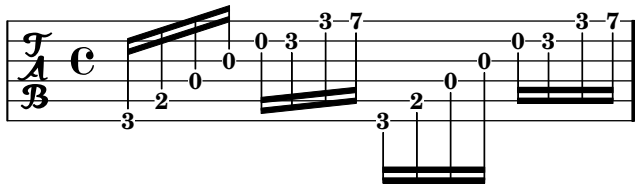


Ausgewählte Schnipsel

Hals- und Balkenverhalten in einer Tabulatur

Die Richtung von Hälsen wird in Tabulaturen genauso wie in normaler Notation eingestellt. Balken können horizontal eingestellt werden, wie das Beispiel zeigt.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam #'damping = #+inf.0
    g,,16 b d g b d g b
  }
}
```



Polyphonie in einer Tabulatur

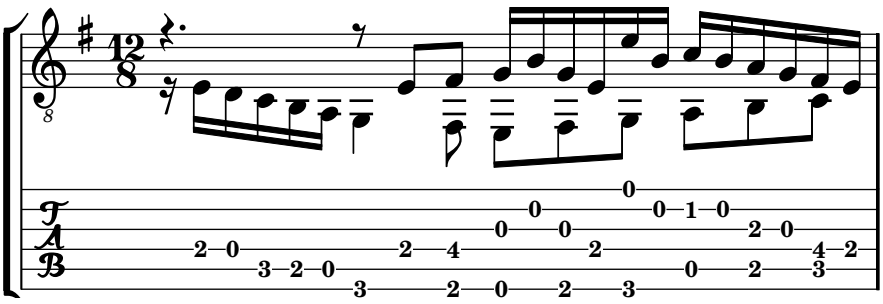
Polyphonie kann in einer Tabulatur (`TabStaff`) genauso wie in einem normalen Notensystem erstellt werden.

```
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  <<
```

```
    \new StaffGroup = "tab with traditional" <<
      \new Staff = "guitar traditional" <<
        \clef "treble_8"
        \context Voice = "upper" \upper
        \context Voice = "lower" \lower
      >>
      \new TabStaff = "guitar tab" <<
        \context TabVoice = "upper" \upper
        \context TabVoice = "lower" \lower
      >>
    >>
  >>
}
```



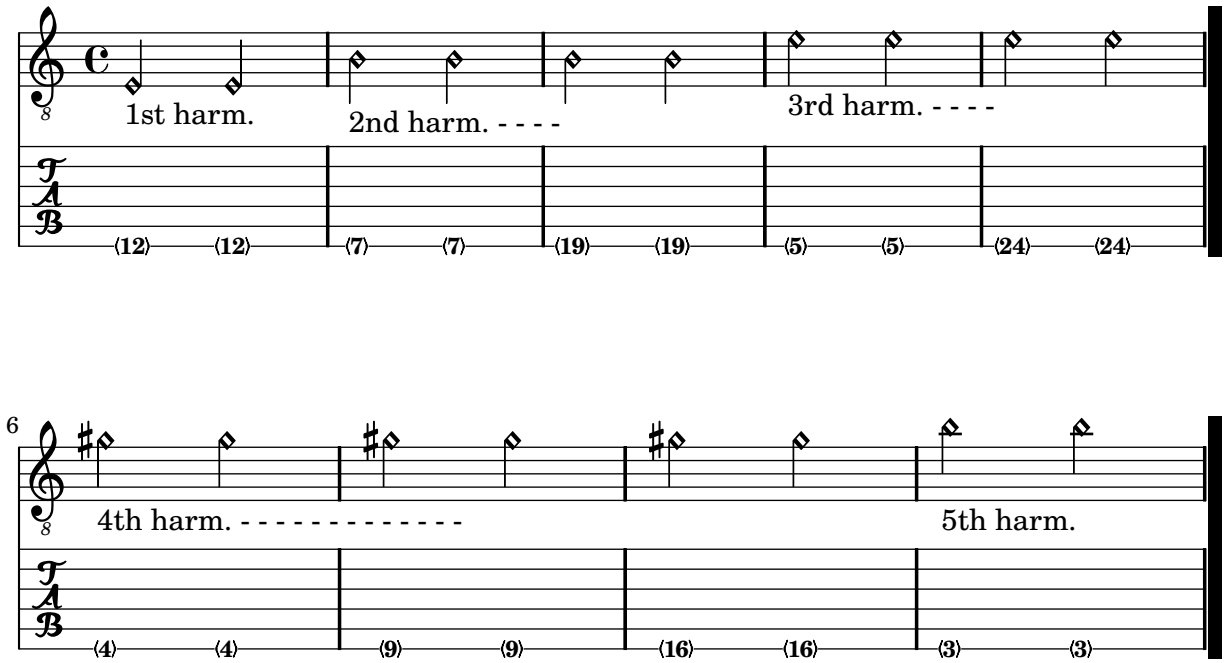
Reference for open-string harmonics

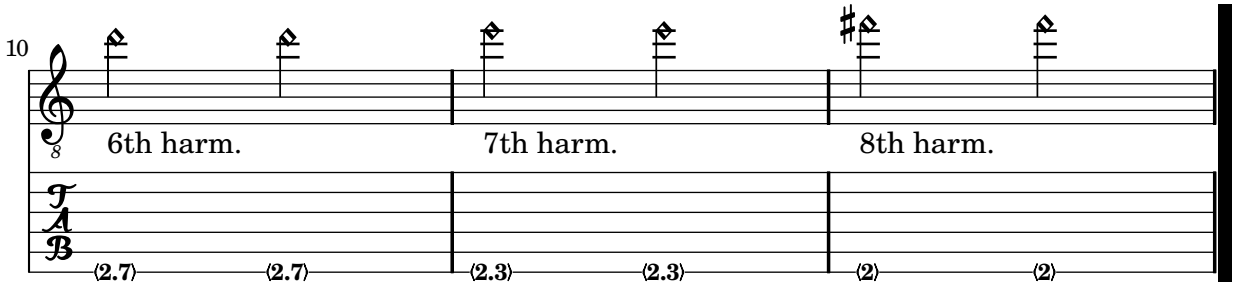
Reference for open-string harmonics:

```
openStringHarmonics = {
  %first harmonic
  \harmonicByFret #12 e,2\6_\markup{"1st harm."}
  \harmonicByRatio #1/2 e,\6
  %second harmonic
  \harmonicByFret #7 e,\6_\markup{"2nd harm. - - - -"}
  \harmonicByRatio #1/3 e,\6
  \harmonicByFret #19 e,\6
  \harmonicByRatio #2/3 e,\6
  %\harmonicByFret #19 < e,\6 a,\5 d\4 >
  %\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >
  %third harmonic
  \harmonicByFret #5 e,\6_\markup{"3rd harm. - - - -"}
  \harmonicByRatio #1/4 e,\6
  \harmonicByFret #24 e,\6
  \harmonicByRatio #3/4 e,\6
  \break
  %fourth harmonic
  \harmonicByFret #4 e,\6_\markup{"4th harm. - - - - - - - - - - - - -"}
  \harmonicByRatio #1/5 e,\6
  \harmonicByFret #9 e,\6
  \harmonicByRatio #2/5 e,\6
  \harmonicByFret #16 e,\6
  \harmonicByRatio #3/5 e,\6
  %fifth harmonic
```

```
  \harmonicByFret #3 e,\6_\markup{"5th harm."}
  \harmonicByRatio #1/6 e,\6
  \break
  %sixth harmonic
  \harmonicByFret #2.7 e,\6_\markup{"6th harm."}
  \harmonicByRatio #1/7 e,\6
  %seventh harmonic
  \harmonicByFret #2.3 e,\6_\markup{"7th harm."}
  \harmonicByRatio #1/8 e,\6
  %eighth harmonic
  \harmonicByFret #2 e,\6_\markup{"8th harm."}
  \harmonicByRatio #1/9 e,\6
}

\score {
  <<
    \new Staff {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}
```





Fretted-string harmonics in tablature

Fretted-string harmonics:

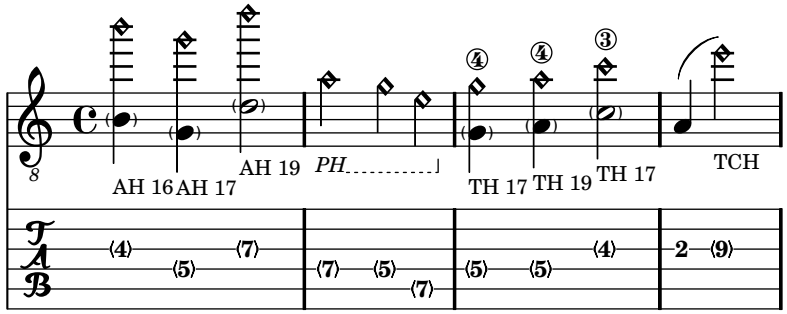
```
pinchedHarmonics = {
   \textSpannerDown
   \override TextSpanner #'bound-details #'left #'text =
      \markup {\halign #-0.5 \teeny "PH" }
      \override TextSpanner #'style =
         #'dashed-line
   \override TextSpanner #'dash-period = #0.6
   \override TextSpanner #'bound-details #'right #'attach-dir = #1
   \override TextSpanner #'bound-details #'right #'text =
      \markup { \draw-line #'(0 . 1) }
   \override TextSpanner #'bound-details #'right #'padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b''\harmonic>4_\markup{ \teeny "AH 16" }
  <\parenthesize g g''\harmonic>4_\markup{ \teeny "AH 17" }
  <\parenthesize d' d'''\harmonic>2_\markup{ \teeny "AH 19" }
  %pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <g'\harmonic>4
  <e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4_\markup{ \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4_\markup{ \teeny "TH 19" }
  <\parenthesize c'\3 c''\harmonic>2_\markup{ \teeny "TH 17" }
  %touch harmonics (TCH)
  a4( <e''\harmonic>2. )_\markup{ \teeny "TCH" }
}

frettedStrings = {
  %artificial harmonics (AH)
  \harmonicByFret #4 g4\3
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 g2\3
  %pinched harmonics (PH)
  \harmonicByFret #7 d2\4
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 a4\5
  %tapped harmonics (TH)
  \harmonicByFret #5 d4\4
```

```
  \harmonicByFret #5 d4\4
  \harmonicByFret #4 g2\3
  %touch harmonics (TCH)
  a4 \harmonicByFret #9 g2.\3
}

\score {
  <<
    \new Staff {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}
```



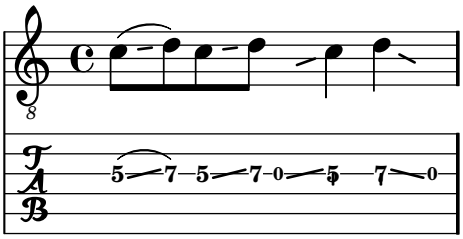
Slides in tablature

Slides can be typeset in both `Staff` and `TabStaff` contexts:

```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\3\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
  \stemDown \hideNotes
  g16\3 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>
```

```
  \layout {
    \context {
      \Score
      \override Glissando #'minimum-length = #4
      \override Glissando #'springs-and-rods =
                          #ly:spanner::set-spacing-rods
      \override Glissando #'thickness = #2
    }
  }
}
```

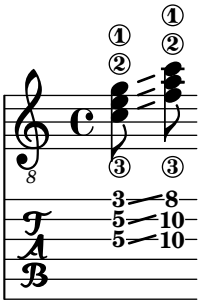


Chord glissando in tablature

Slides for chords can be indicated in both Staff and TabStaff. String numbers are necessary for TabStaff because automatic string calculations are different for chords and for single notes, and `\chordGlissando` draws lines between single notes.

```
myMusic = \relative c' {
  \chordGlissando
  <c\3 e\2 g\1>8 <f\3 a\2 c\1>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff {
      \myMusic
    }
  >>
}
```



Siehe auch

Notationsreferenz: [Hälse], Seite 196, [Flageolett], Seite 301, [Glissando], Seite 122.

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

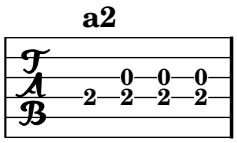
Referenz der Interna: Abschnitt “TabNoteHead” in *Referenz der Interna*, Abschnitt “TabStaff” in *Referenz der Interna*, Abschnitt “TabVoice” in *Referenz der Interna*, Abschnitt “Beam” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Akkorde werden nicht gesondert behandelt, sodass die Saitenauswahlfunktion eventuell die selbe Saite für zwei Töne eines Akkordes auswählen kann.

Damit die Kombination von Stimmen (`\partcombine`) richtig funktioniert, müssen speziell erstellte Stimmen innerhalb des Tabulatursystems (`TabStaff`) benutzt werden:

```
melodia = \partcombine { e4 g g g } { e4 e e e }
<<
  \new TabStaff <<
    \new TabVoice = "one" s1
    \new TabVoice = "two" s1
    \new TabVoice = "shared" s1
    \new TabVoice = "solo" s1
    { \melodia }
  >>
>>
```



Spezialeffekte für Gitarre beschränken sich auf Flageolett und Slide.

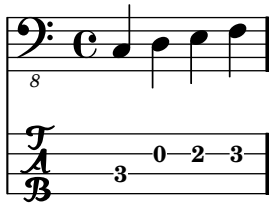
Angepasste Tabulaturen

LilyPond errechnet automatisch den Bund für eine Note auf Grundlage der Saite, zu welcher der Ton zugeordnet ist. Um das tun zu können, muss die Stimmung der Saiten angegeben werden. Die Stimmung wird in der `StringTunings`-Eigenschaften bestimmt.

LilyPond hat vordefinierte Stimmungen für Banjo, Mandoline, Gitarre, Bassgitarre, Ukulele, Geige, Bratsche, Cello und Kontrabass. Für diese Stimmungen wird automatisch die richtige Transposition eingesetzt. Das nächste Beispiel ist für Bassgitarre, welche eine Oktave niedriger erklingt, als sie geschrieben ist:

```
<<
  \new Staff {
    \clef "bass_8"
    \relative c, {
      c4 d e f
    }
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #bass-tuning
    \relative c, {
      c4 d e f
    }
```

```
  }
>>
```

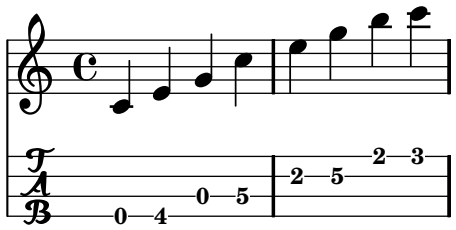


Die Standardstimmung ist die Gitarrenstimmung (`guitar-tuning`) in der EADGHE-Stimmung. Andere vordefinierte Stimmung sind: `guitar-open-g-tuning`, `mandolin-tuning` und `banjo-open-g-tuning`. Die vordefinierten Stimmungen finden sich in '`scm/string-tuning-init.scm`'.

Jede beliebige Stimmung kann erstellt werden. Die Funktion `\contextStringTuning` kann benutzt werden, um eine Saitenstimmung zu definieren und als den Wert von `stringTunings` für den aktuellen Kontext zu bestimmen. `\contextStringTuning` braucht zwei Argumente: das Symbol, wo die Saitenstimmung gespeichert werden soll, und eine Akkordkonstruktion, die die Tonhöhen jeder Saite der Stimmung angibt. Die Akkordkonstruktion muss im absoluten Oktavenmodus angegeben werden, siehe [Absolute Oktavenbezeichnung], Seite 1. Die Saite mit der höchsten Zahl (normalerweise die tiefste Seite) muss im Akkord zuerst geschrieben werden. Eine Stimmung für ein viersaitiges Instrument mit den Tonhöhen `a''`, `d''`, `g'` und `c'` kann folgenderweise erstellt werden:

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \contextStringTuning #'custom-tuning <c' g' d'' a''>
    \mynotes
  }
>>
```



Die `stringTunings`-Eigenschaft wird auch von `FretBoards` benutzt, um automatische Bunddiagramme zu errechnen.

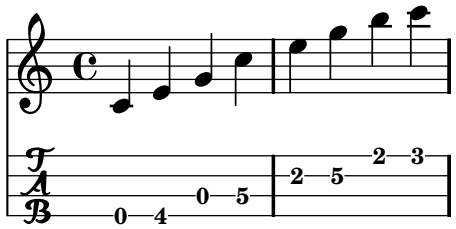
Saitensitmmungen werden als Teil des Hash-Schlüsselwertes für vordefinierte Bunddiagramme eingesetzt (siehe auch [Vordefinierte Bund-Diagramme], Seite 330. Die Funktion

`\makeStringTuning` wird benutzt, um eine Stimmung im aktuellen Kontext zu erstellen, ohne die `stringTunings`-Eigenschaft einzusetzen Die Argumente von `\makeStringTuning` sind das Symbol, das für die neue Stimmung benutzt werden soll und eine Akkordkonstruktion, die die Stimmung definiert. Das vorherige Beispiel könnte auch folgenderweise geschrieben werden:

```
\makeStringTuning #'custom-tuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #custom-tuning
    \mynotes
  }
>>
```



Intern ist die Stimmung eine Scheme-Liste von Tonhöhen der Saiten, eine für jede Saite, geordnet von Saitennummer 1 bis n, wobei 1 die höchste Saite der Tabulatur ist und n die unterste. Normalerweise wird so die Stimmung vom höchsten bis zum tiefsten Ton angegeben, aber bei einige Instrumente (etwa Ukulele) werden die Saiten nicht aufgrund der Tonhöhe angeordnet.

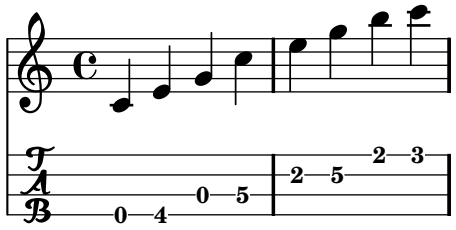
Die Tonhöhe einer Saite in einer Seitenstimmungsliste ist ein Tonhöhenobjekt für LilyPond. Tonhöhenobjekte werden mit der Scheme-Funktion `+ly:make-pitch` erstellt (siehe Abschnitt A.18 [Scheme-Funktionen], Seite 680).

Wenn gewünscht, kann eine Saitenstimmung auch in Scheme-Sprache erstellt werden. Das Beispiel unten bildet das obige Beispiel nach, aber die Stimmung wird nicht als eigenes Objekt gespeichert:

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #`(,(ly:make-pitch 1 5 0)
```

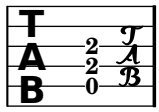
```
                                        ,(ly:make-pitch 1 1 0)
                                        ,(ly:make-pitch 0 4 0)
                                        ,(ly:make-pitch 0 0 0))
    \mynotes
  }
>>
```



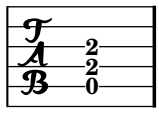
LilyPond errechnet automatisch die Linienanzahl für die Tabulatur und die Zahl der Saiten in dem automatisch erstellten `FretBoard` (Bunddiagramm) aus der Anzahl der Elemente von `stringTunings`.

Auch ein moderner Tabulatur-Schlüssel kann verwendet werden:

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
  <a, e a>1
}
```



2



Der moderne Tabulatur-Schlüssel unterstützt Tabulaturen von 4 bis 7 Saiten.

Siehe auch

Notationsreferenz: [Absolute Oktavenbezeichnung], Seite 1, [Vordefinierte Bund-Diagramme], Seite 330, Abschnitt A.18 [Scheme-Funktionen], Seite 680.

Installierte Dateien: ‘`ly/string-tuning-init.ly`’ ‘`scm/tablature.scm`’.

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Referenz der Interna: Abschnitt “Tab_note_heads_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

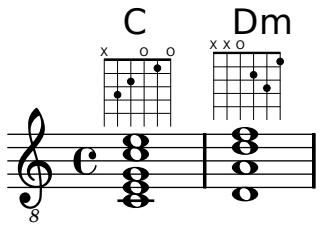
Automatische Tabulatur-Berechnung funktioniert in den meisten Fällen nicht korrekt bei Instrumenten, deren Saitenstimmung nicht monotonisch fortschreitet, wie etwa Ukulele.

Bund-Diagramm-Beschriftung

Bunddiagramme können zu Notation als Textbeschriftung hinzugefügt werden. Die Beschriftung enthält Information zu dem gewünschten Bunddiagramm. Es gibt drei unterschiedliche Darstellungsarten: normal, knapp und ausführlich. Die drei Arten erzeugen die gleiche Ausgabe, aber mit jeweils mehr oder weniger Einzelheiten. Einzelheiten zur Syntax der unterschiedlichen Beschriftungsbefehle, mit denen die Bunddiagramme definiert werden, findet sich in ⟨undefined⟩ [Beschriftung für einzelne Instrumente], Seite ⟨undefined⟩.

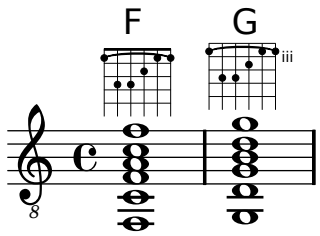
Die Standard-Bunddiagrammbeschriftung beinhaltet die Saitennummer und die Bundnummer für jeden Punkt, der notiert werden soll. Zusätzlich können offenen und nicht gespielte (schwingende) Saiten angezeigt werden.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram #"6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram #"6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



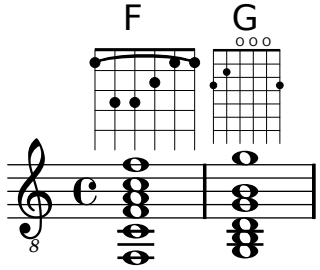
Barré kann hinzugefügt werden:

```
<<
  \context ChordNames {
     \chordmode {
       f1 g
     }
  }
  \context Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram #"c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram #"c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
    }
  }
>>
```



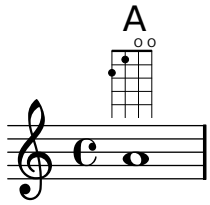
Die Größe des Bunddiagrammes und die Anzahl der Bünde im Diagramm kann geändert werden:

```
<<
  \context ChordNames {
     \chordmode {
       f1 g
     }
  }
  \context Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram #"s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram #"h:6;6-3;5-2;4-o;3-o;2-o;1-3;"
    }
  }
>>
```



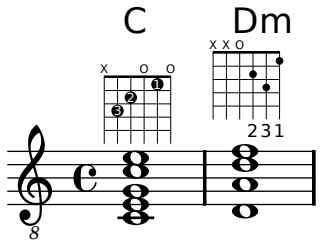
Die Anzahl der Saiten in einem Bunddiagramm kann geändert werden, um sie für andere Instrumente anzupassen, wie etwas Banjo oder Ukulele.

```
<<
  \context ChordNames {
    \chordmode {
      a1
    }
  }
  \context Staff {
    % An 'A' chord for ukulele
    a'1^\markup {
      \fret-diagram #"w:4;4-2-2;3-1-1;2-o;1-o;"
    }
  }
>>
```



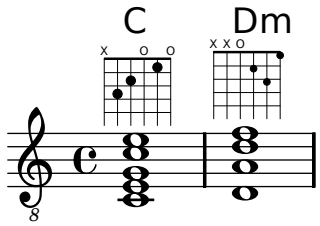
Fingersatz kann auch angezeigt werden, und die Position der Fingersatzzahlen kann kontrolliert werden.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram #"f:1;6-x;5-3-3;4-2-2;3-o;2-1-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram #"f:2;6-x;5-x;4-o;3-2-2;2-3-3;1-1-1;"
    }
  }
>>
```



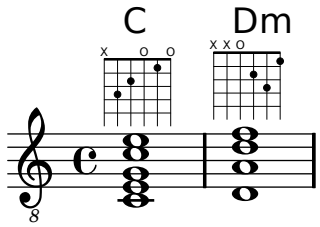
Die Größe und Position der Punkte kann geändert werden:

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram #"d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram #"p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



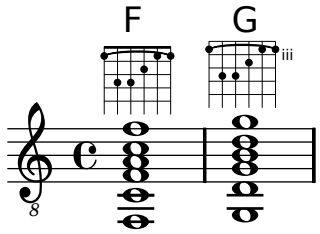
Die Beschriftungsfunktion `fret-diagram-terse` (knappe Version) lässt die Saitennummern aus: das Vorhandensein einer Saite wird durch ein Semikolon ausgedrückt. Für jede Saite des Diagramms muss ein Semikolon gesetzt werden. Das erste Semikolon entspricht der höchsten Saite, das letze der ersten Saite. Stumme und offene Saiten sowie Bundnummern können angezeigt werden.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse #"x;3;2;o;1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse #"x;x;o;2;3;1;"
    }
  }
>>
```



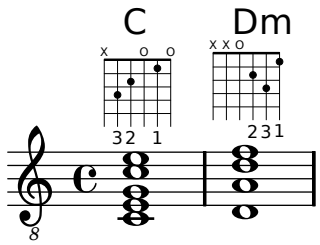
Barré kann im knappen Modus auch angezeigt werden:

```
<<
  \context ChordNames {
    \chordmode {
      f1 g
    }
  }
  \context Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram-terse #"1-(;3;3;2;1;1-);"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram-terse #"3-(;5;5;4;3;3-);"
    }
  }
>>
```



Fingersatz kann im knappen Modus hinzugefügt werden:

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context Staff {
    \override Voice.TextScript
      #'(fret-diagram-details finger-code) = #'below-string
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse #"x;x;o;2-2;3-3;1-1;"
    }
  }
>>
```

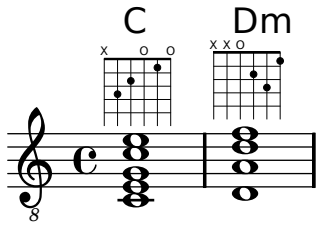


Andere Eigenschaften der Bunddiagramme müssen im knappen Modus mit `\override`-Befehlen angegeben werden.

Die Beschriftungsfunktion `fret-diagram-verbose` (ausführlicher Stil) ist in der Form eine Scheme-Liste. Jedes Element stellt ein Element dar, dass im Bunddiagramm gesetzt werden soll.

```
<<
    \context ChordNames {
      \chordmode {
        c1 d:m
      }
    }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (place-fret 5 3)
        (place-fret 4 2)
        (open 3)
        (place-fret 2 1)
```

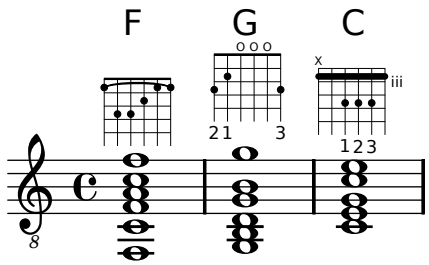
```
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
>>
```



Fingersatz und Barré kann im ausführlichen Modus notiert werden. Nur im ausführlichen Modus kann ein Capo angezeigt werden, das auf dem Bunddiagramm plaziert wird. Die Capo-Anzeige ist ein dicker Strich, der alle Saiten bedeckt. Der Bund mit dem Capo ist der unterste Bund im Diagramm.

```
<<
    \context ChordNames {
      \chordmode {
        f1 g c
      }
    }
  \context Staff {
    \clef "treble_8"
    \override Voice.TextScript
      #'(fret-diagram-details finger-code) = #'below-string
    <f, c f a c' f'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 1)
        (place-fret 5 3)
        (place-fret 4 3)
        (place-fret 3 2)
        (place-fret 2 1)
        (place-fret 1 1)
        (barre 6 1 1)
      )
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 2)
        (place-fret 5 2 1)
```

```
        (open 4)
        (open 3)
        (open 2)
        (place-fret 1 3 3)
      )
    }
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (capo 3)
        (mute 6)
        (place-fret 4 5 1)
        (place-fret 3 5 2)
        (place-fret 2 5 3)
      )
    }
  }
>>
```



Alle anderen Bunddiagramm-Eigenschaften müssen im ausführlichen Modus mit mit `\override`-Befehlen angegeben werden.

Die graphische Erscheinung eines Bunddiagramms kann den Wünschen des Notensetzers angepasst werden. Hierzu werden die Eigenschaften des `fret-diagram-interface` (Bunddiagramm-Schnittstelle) eingesetzt. Einzelheiten hierzu in Abschnitt "fret-diagram-interface" in *Referenz der Interna*. Die Eigenschaften der Schnittstelle gehören dem `Voice.TextScript`-Kontext an.

Ausgewählte Schnipsel

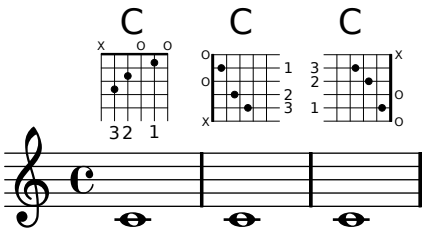
Changing fret orientations

Fret diagrams can be oriented in three ways. By default the top string or fret in the different orientations will be aligned.

```
\include "predefined-guitar-fretboards.ly"

<<
  \chords {
    c1
    c1
    c1
  }
  \new FretBoards {
    \chordmode {
      c1
      \override FretBoard #'(fret-diagram-details orientation) =
        #'landscape
```

```
      c1
      \override FretBoard #'(fret-diagram-details orientation) =
        #'opposing-landscape
      c1
    }
  }
  \new Voice {
    c'1
    c'1
    c'
  }
>>
```



Anpassung von Beschriftungs-Bunddiagrammen

Bunddiagramme können mit der Eigenschaft `'fret-diagram-details` angepasst werden. Bunddiagramme, die als Textbeschriftung eingefügt werden, können Veränderungen im `Voice.TextScript`-Objekt oder direkt in der Beschriftung vorgenommen werden.

```
<<
  \chords { c1 | c | c | d }

  \new Voice = "mel" {
    \textLengthOn
    % Set global properties of fret diagram
    \override TextScript #'size = #'1.2
    \override TextScript
      #'(fret-diagram-details finger-code) = #'in-dot
    \override TextScript
      #'(fret-diagram-details dot-color) = #'white

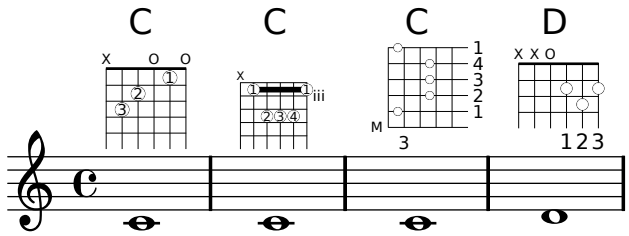
    %% C major for guitar, no barre, using defaults
       % terse style
    c'1^\markup { \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;" }

    %% C major for guitar, barred on third fret
       % verbose style
       % size 1.0
       % roman fret label, finger labels below string, straight barre
    c'1^\markup {
      % standard size
      \override #'(size . 1.0) {
        \override #'(fret-diagram-details . (
                     (number-type . roman-lower)
                     (finger-code . in-dot)
```

```
                    (barre-type . straight))) {
          \fret-diagram-verbose #'((mute 6)
                                   (place-fret 5 3 1)
                                   (place-fret 4 5 2)
                                   (place-fret 3 5 3)
                                   (place-fret 2 5 4)
                                   (place-fret 1 3 1)
                                   (barre 5 1 3))
        }
      }
    }

    %% C major for guitar, barred on third fret
       % verbose style
       % landscape orientation, arabic numbers, M for mute string
       % no barre, fret label down or left, small mute label font
    c'1^\markup {
      \override #'(fret-diagram-details . (
                   (finger-code . below-string)
                   (number-type . arabic)
                   (label-dir . -1)
                   (mute-string . "M")
                   (orientation . landscape)
                   (barre-type . none)
                   (xo-font-magnification . 0.4)
                   (xo-padding . 0.3))) {
        \fret-diagram-verbose #'((mute 6)
                                 (place-fret 5 3 1)
                                 (place-fret 4 5 2)
                                 (place-fret 3 5 3)
                                 (place-fret 2 5 4)
                                 (place-fret 1 3 1)
                                 (barre 5 1 3))
      }
    }

    %% simple D chord
       % terse style
       % larger dots, centered dots, fewer frets
       % label below string
    d'1^\markup {
      \override #'(fret-diagram-details . (
                   (finger-code . below-string)
                   (dot-radius . 0.35)
                   (dot-position . 0.5)
                   (fret-count . 3))) {
        \fret-diagram-terse #"x;x;o;2-1;3-2;2-3;"
      }
    }
  }
>>
```



Siehe auch

Notationsreferenz: ⟨undefined⟩ [Beschriftung für einzelne Instrumente], Seite ⟨undefined⟩.

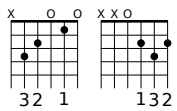
Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Referenz der Interna: Abschnitt “fret-diagram-interface” in *Referenz der Interna*.

Vordefinierte Bund-Diagramme

Bunddiagramme können mit dem `FretBoards`-Kontext angezeigt werden. Standardmäßig zeigt der `FretBoards`-Kontext Bunddiagramme an, die in einer Tabelle definiert sind:

```
\include "predefined-guitar-fretboards.ly"
\context FretBoards {
  \chordmode {
    c1 d
  }
}
```



Die vordefinierten Diagramme sind in der Datei ‘`predefined-guitar-fretboards.ly`’ enthalten. Sie werden basierend auf der Tonhöhe eines Akkordes und dem Wert von `stringTunings` (Saitenstimmung), der gerade benutzt wird, gespeichert. ‘`predefined-guitar-fretboards.ly`’ beinhaltet vordefinierte Diagramme für die Gitarrenstimmung (`guitar-tuning`). Anhand der Beispiele in dieser Datei können auch für andere Instrumente oder Stimmungen Diagramme definiert werden.

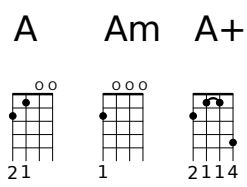
Bunddiagramme für die Ukulele finden sich in der Datei ‘`predefined-ukulele-fretboards.ly`’.

```
\include "predefined-ukulele-fretboards.ly"

myChords = \chordmode { a1 a:m a:aug }

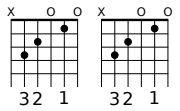
\new ChordNames {
  \myChords
}

\new FretBoards {
  \set stringTunings = #ukulele-tuning
  \myChords
}
```



Tonhöhen von Akkorden können entweder als Akkordkonstrukte oder im Akkordmodus notiert werden (siehe auch [Überblick über den Akkord-Modus], Seite 371).

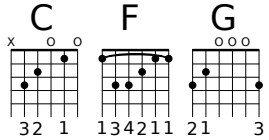
```
\include "predefined-guitar-fretboards.ly"
\context FretBoards {
  \chordmode { c1 }
  <c' e' g'>1
}
```



Oft wird sowohl eine Akkordbezeichnung als ein Bunddiagramm notiert. Das kann erreicht werden, indem ein `ChordNames`-Kontext parallel mit einem `FretBoards`-Kontext gesetzt wird und beiden Kontexten die gleichen Noten zugewiesen werden.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}

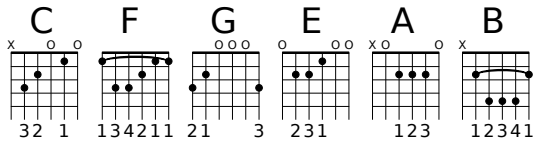
<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



Vordefinierte Bunddiagramme können transponiert werden, solange ein Diagramm für den transponierten Akkord in der Bunddiagramm-Tabelle vorhanden ist.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}

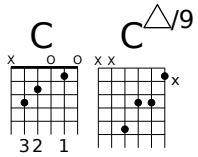
mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \context ChordNames {
    \mychordlist
  }
  \context FretBoards {
    \mychordlist
  }
>>
```



Die Tabelle der vordefinierten Bunddiagramme für Gitarre enthält acht Akkorde (Dur, Moll, übermäßig, vermindert, Dominantseptakkord, große Septime, kleine Septime und Dominantnonenakkord) für alle 17 Tonarten. Die Tabelle der vortdefinierten Bunddiagramme für Ukulele enthält neben diesen Akkorden noch zusätzlich drei weitere (große Sext, Sekundakkord und Quartakkort). Eine vollständige Liste der vordefinierten Bunddiagramme findet sich in Abschnitt A.4 [Die vordefinierten Bund-Diagramme], Seite 568. Wenn in der Tabelle für einen Akkord kein Wert steht, wird ein Bunddiagramm vom `FretBoards`-Engraver errechnet, wobei die automatische Bunddiagrammfunktion zu Anwendung kommt. Siehe hierzu [Automatische Bund-Diagramme], Seite 340.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 c:maj9
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



Bunddiagramme können zu der Tabelle hinzugefügt werden. Um ein Diagramm hinzuzufügen, muss der Akkord des Diagramms, die Stimmung und die Diagramm-Definition angegeben werden. Dies geschieht normalerweise in der Tabelle *default-fret-table*. Die Diagramm-Definition kann entweder eine `fret-diagram-terse`-Definition oder eine `fret-diagram-verbose`-Liste sein.

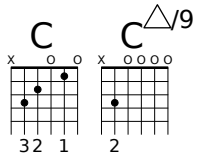
```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c:maj9 }
                        #guitar-tuning
                        #"x;3-2;o;o;o;o;"

mychords = \chordmode {
  c1 c:maj9
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
```

```
    \mychords
  }
>>
```



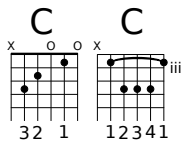
Unterschiedliche Bunddiagramme für den selben Akkord können gespeichert werden, indem unterschiedliche Oktaven für die Tonhöhe benutzt werden. Die unterschiedliche Oktave sollte mindestens zwei Oktaven über oder unter der Standardoktave liegen, die für transponierende Bunddiagramme eingesetzt wird.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c'' }
                        #guitar-tuning
                        #(offset-fret 2 (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
  c1 c''
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



Zusätzlich zu Bunddiagrammen speichert LilyPond auch eine interne Liste an Akkordformen. Die Akkordformen sind Bunddiagramme, die am Hals entlang verschoben werden können und dabei unterschiedliche Akkorde ergeben. Akkordformen können zu der internen Liste hinzugefügt werden und dann benutzt werden, um vordefinierte Bunddiagramme zu definieren. Weil sie auf verschiedenen Positionen auf dem Steg gelegt werden können, beinhalten vordefinierte Akkord üblicherweise keine leeren Saiten. Wie Bunddiagramme können auch Akkordformen entweder als `fret-diagram-terse`-Definition oder als `fret-diagram-verbose`-Liste erstellt werden.

```
\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

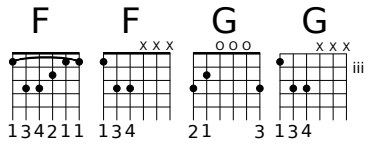
\addChordShape #'powerf #guitar-tuning #"1-1;3-3;3-4;x;x;x;"

% add some new chords based on the power chord shape
```

```
\storePredefinedDiagram #default-fret-table
                        \chordmode { f'' }
                        #guitar-tuning
                        #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
                        \chordmode { g'' }
                        #guitar-tuning
                        #(offset-fret 2 (chord-shape 'powerf guitar-tuning))

mychords = \chordmode{
  f1 f'' g g''
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



Die graphische Form eines Bunddiagramms kann entsprechend den eigenen Wünschen verändert werden, indem man die Eigenschaften der `fret-diagram-interface`-Schnittstelle verändert. Einzelheiten hierzu in Abschnitt "fret-diagram-interface" in *Referenz der Interna*. Die Schnittstelleneigenschaften eines vordefinierten Bunddiagrammes gehören dem `FretBoards.FretBoard`-Kontext an.

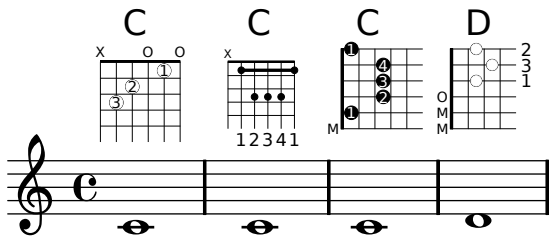
Ausgewählte Schnipsel

Bunddiagramme anpassen

Eigenschaften von Bunddiagrammen können in `'fret-diagram-details` verändert werden. Einstellungen mit dem `\override`-Befehl werden dem `FretBoards.FretBoard`-Objekt zugewiesen. Genauso wie `Voice` ist auch `FretBoards` ein Kontext der niedrigsten Ebene, weshalb der Kontext auch in dem Befehl weggelassen werden kann.

```
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        #"x;1-1-(;3-2;3-3;3-4;1-1-);"
<<
  \new ChordNames {
    \chordmode { c1 | c | c | d }
  }
  \new FretBoards {
    % Set global properties of fret diagram
    \override FretBoards.FretBoard #'size = #'1.2
    \override FretBoard
```

```
      #'(fret-diagram-details finger-code) = #'in-dot
    \override FretBoard
      #'(fret-diagram-details dot-color) = #'white
    \chordmode {
      c
      \once \override FretBoard #'size = #'1.0
      \once \override FretBoard
        #'(fret-diagram-details barre-type) = #'straight
      \once \override FretBoard
        #'(fret-diagram-details dot-color) = #'black
      \once \override FretBoard
        #'(fret-diagram-details finger-code) = #'below-string
      c'
      \once \override FretBoard
        #'(fret-diagram-details barre-type) = #'none
      \once \override FretBoard
        #'(fret-diagram-details number-type) = #'arabic
      \once \override FretBoard
        #'(fret-diagram-details orientation) = #'landscape
      \once \override FretBoard
        #'(fret-diagram-details mute-string) = #"M"
      \once \override FretBoard
        #'(fret-diagram-details label-dir) = #LEFT
      \once \override FretBoard
        #'(fret-diagram-details dot-color) = #'black
      c'
      \once \override FretBoard
        #'(fret-diagram-details finger-code) = #'below-string
      \once \override FretBoard
        #'(fret-diagram-details dot-radius) = #0.35
      \once \override FretBoard
        #'(fret-diagram-details dot-position) = #0.5
      \once \override FretBoard
        #'(fret-diagram-details fret-count) = #3
      d
    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>
```



Eigene vordefinierte Bunddiagramme für andere Instrumente erstellen

Vordefinierte Bunddiagramme können für neue Instrumente hinzugefügt werden neben denen, die schon für die Gitarre definiert sind. Dieses Schnipsel zeigt, wie man eine neue Saitenstim-

mung definiert und dann eigene vordefinierte Bunddiagramme bestimmt. Das Beispiel ist für das venezualische Cuatro.

Dieses Schnipsel zeigt auch, wie Fingersatz in die Akkorde eingebunden werden kann, um als Referenzpunkt für die Akkordauswahl benutzt werden kann. Dieser Fingersatz wird im Bunddiagramm und in der Tabulatur, aber nicht in den Noten angezeigt.

Diese Bunddiagramme sind nicht transponierbar, weil sie Saiteninformationen enthalten. Das soll in der Zukunft verbessert werden.

```
% add FretBoards for the Cuatro
%   Note: This section could be put into a separate file
%      predefined-cuatro-fretboards.ly
%      and \included into each of your compositions

cuatroTuning = #`(,(ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))

dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }

\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        #"o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        #"o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        #"o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        #"o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                        #cuatroTuning
                        #"2-2;o;1-1;o;"

% end of potential include file /predefined-cuatro-fretboards.ly


#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
```

```
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

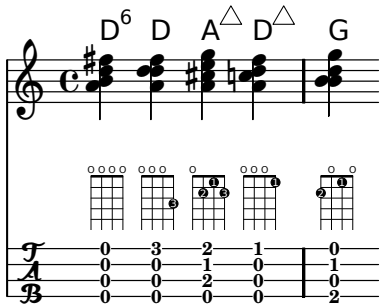
    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

    \new FretBoards {
      \set stringTunings = #cuatroTuning
%      \override FretBoard
%        #'(fret-diagram-details string-count) = #'4
      \override FretBoard
        #'(fret-diagram-details finger-code) = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner
        #'base-shortest-duration = #(ly:make-moment 1 16)
    }
  }
  \midi { }
}
```



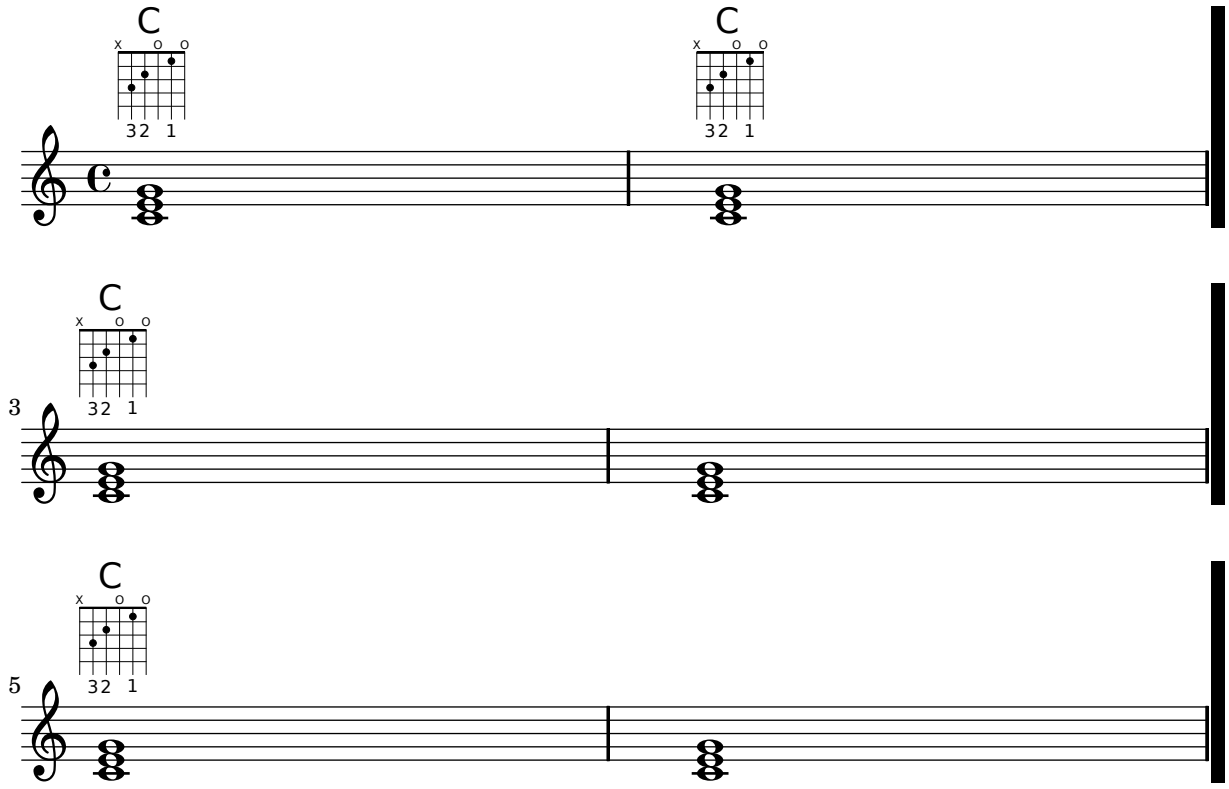
Akkordänderungen für Bunddiagramme

Bunddiagramme können definiert werden, sodass sie nur angezeigt werden, wenn der Akkord sich ändert oder eine neue Zeile anfängt.

```
\include "predefined-guitar-fretboards.ly"

myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1 \break
}

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
```



Alternative Bunddiagrammtabellen

Alternative Bunddiagrammtabellen können erstellt werden. Sie können benutzt werden, um alternative Bunddiagramme für einen bestimmten Akkord zu haben.

Damit eine alternative Bunddiagrammentabelle benutzt werden kann, muss die Tabelle zuerst erstellt werden. Dann werden die Bunddiagramme zur Tabelle hinzugefügt.

Die erstellte Bunddiagrammtabelle kann auch leer sein, oder sie kann aus einer existierenden Tabelle kopiert werden.

Die Tabelle, die eingesetzt wird, um vordefinierte Bunddiagramme anzuzeigen, wird mit der Eigenschaft `\predefinedDiagramTable` ausgewählt.

```
\include "predefined-guitar-fretboards.ly"

% Make a blank new fretboard table
#(define custom-fretboard-table-one (make-fretboard-table))

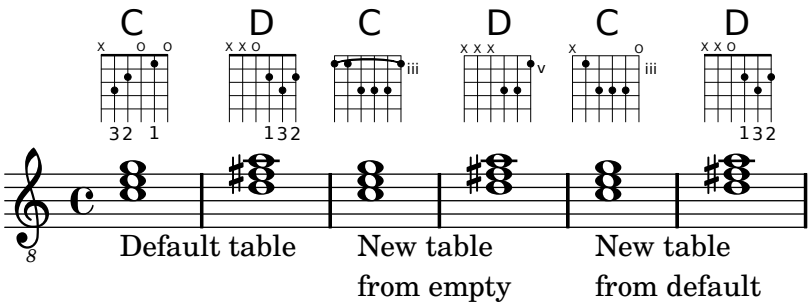
% Make a new fretboard table as a copy of default-fret-table
#(define custom-fretboard-table-two (make-fretboard-table default-fret-table))

% Add a chord to custom-fretboard-table-one
\storePredefinedDiagram #custom-fretboard-table-one
                        \chordmode{c}
                        #guitar-tuning
                        "3-(;3;5;5;5;3-);"

% Add a chord to custom-fretboard-table-two
\storePredefinedDiagram #custom-fretboard-table-two
                        \chordmode{c}
                        #guitar-tuning
                        "x;3;5;5;5;o;"

<<
  \chords {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
  \new FretBoards {
    \chordmode {
      \set predefinedDiagramTable = #default-fret-table
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-one
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-two
      c1 | d1 |
    }
  }
  \new Staff {
    \clef "treble_8"
    <<
      \chordmode {
        c1 | d1 |
        c1 | d1 |
        c1 | d1 |
      }
      {
```

```
        s1_\markup "Default table" | s1 |
        s1_\markup \column {"New table" "from empty"} | s1 |
        s1_\markup \column {"New table" "from default"} | s1 |
      }
    >>
  }
>>
```



Siehe auch

Notationsreferenz: [Angepasste Tabulaturen], Seite 317, [Automatische Bund-Diagramme], Seite 340, [Überblick über den Akkord-Modus], Seite 371, Abschnitt A.4 [Die vordefinierten Bund-Diagramme], Seite 568.

Installierte Dateien: ‘`ly/predefined-guitar-fretboards.ly`’, ‘`ly/predefined-guitar-ninth-fretboards.ly`’, ‘`ly/predefined-ukulele-fretboards.ly`’.

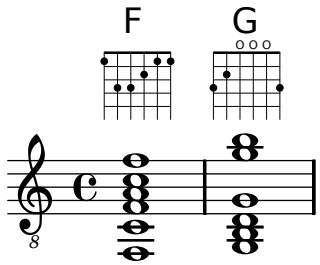
Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Referenz der Interna: Abschnitt “fret-diagram-interface” in *Referenz der Interna*.

Automatische Bund-Diagramme

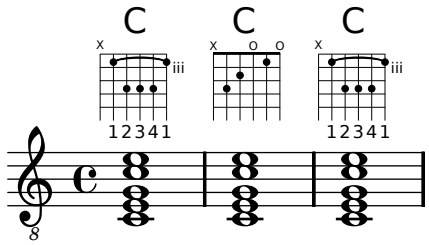
Bunddiagramme können automatisch aus notierten Noten erstellt werden. Hierzu wird der `FretBoards`-Kontext eingesetzt. Wenn keine vordefinierten Diagramme für die entsprechenden Noten mit der aktiven Saitenstimmung (`stringTunings`) vorhanden sind, errechnet der Kontext Saiten und Bünde die benutzt werden können, um die Noten zu spielen.

```
<<
  \context ChordNames {
    \chordmode {
      f1 g
    }
  }
  \context FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \context Staff {
    \clef "treble_8"
    <f, c f a c' f'>1
    <g, b, d g b' g'>1
  }
>>
```



Da in den Standardeinstellungen keine vordefinierten Diagramme geladen werden, ist die automatische Diagrammerstellung das Standardverhalten. Wenn die vordefinierten Diagramme eingesetzt werden, kann die automatische Berechnung an- und ausgeschaltet werden.

```
\storePredefinedDiagram #default-fret-table
                        <c e g c' e'>
                        #guitar-tuning
                        #"x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
  \context ChordNames {
    \chordmode {
      c1 c c
    }
  }
  \context FretBoards {
    <c e g c' e'>1
    \predefinedFretboardsOff
    <c e g c' e'>1
    \predefinedFretboardsOn
    <c e g c' e'>1
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <c e g c' e'>1
    <c e g c' e'>1
  }
>>
```

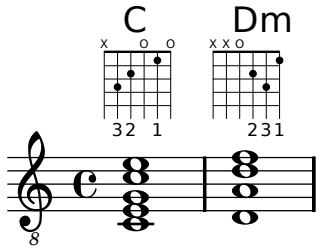


Manchmal kann die Berechnungsfunktion für Bunddiagramme kein passendes Diagramm finden. Das kann oft umgangen werden, indem man manuell einer Note eine bestimmte Saite zuweist. In vielen Fällen muss nur eine Note derart gekennzeichnet werden, der Rest wird dann entsprechend durch den `FretBoards`-Kontext behandelt.

Fingersatz kann zu FretBoard-Bunddiagrammen hinzugefügt werden.

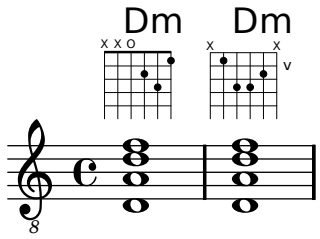
```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
```

```
    }
  }
  \context FretBoards {
    <c-3 e-2 g c'-1 e'>1
    <d a-2 d'-3 f'-1>1
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <d a d' f'>1
  }
>>
```



Der kleinste Bund, der benutzt wird, um Saiten und Bünde im FretBoard-Kontext zu errechnen, kann mit der `minimumFret`-Eigenschaft gesetzt werden.

```
<<
  \context ChordNames {
    \chordmode {
      d1:m d:m
    }
  }
  \context FretBoards {
    <d a d' f'>1
    \set FretBoards.minimumFret = #5
    <d a d' f'>1
  }
  \context Staff {
    \clef "treble_8"
    <d a d' f'>1
    <d a d' f'>1
  }
>>
```



Die Saiten und Bünde des `FretBoards`-Kontextes hängen von der `stringTunings`-Eigenschaft ab, die die gleiche Bedeutung wie im TabStaff-Kontext hat. Siehe auch [Angepasste Tabulaturen], Seite 317 zu Information über die `stringTunings`Eigenschaft.

Die graphische Erscheinung eines Bunddiagrammes kann den Bedürfnissen angepasst werden, indem Eigenschaften der `fret-diagram-interface`-Schnittstelle verändert werden. Einzelheiten finden sich in Abschnitt “fret-diagram-interface” in *Referenz der Interna*. Die Schnittstelleneigenschaften eines `FretBoards`-Diagramms gehören dem `FretBoards.FretBoard`-Kontext an.

Vordefinierte Befehle

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

Siehe auch

Notationsreferenz: [Angepasste Tabulaturen], Seite 317.

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Referenz der Interna: Abschnitt “fret-diagram-interface” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Automatische Bundberechnung funktioniert nicht richtig für Instrumente mit nicht-monotonischer Stimmung.

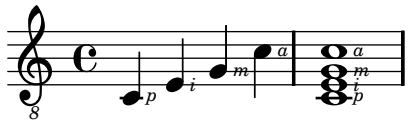
Fingersatz der rechten Hand

cindex Fingersatz der rechten Hand, Bundinstrumente

Fingersatz für die rechte Hand in Akkorden kann mit den Bezeichnungen *p-i-m-a* notiert werden. Er muss innerhalb eines Akkord-Konstruktes notiert werden.

> **Achtung:** Vor dem `\rightHandFinger` **muss** ein Minuszeichen gesetzt werden und ein Leerzeichen nach dem schließenden `>`.

```
\clef "treble_8"
<c-\rightHandFinger #1 >4
<e-\rightHandFinger #2 >
<g-\rightHandFinger #3 >
<c-\rightHandFinger #4 >
<c,-\rightHandFinger #1 e-\rightHandFinger #2
 g-\rightHandFinger #3 c-\rightHandFinger #4 >1
```



Zur Erleichterung kann der Befehl `\rightHandFinger` zu ein paar Buchstaben abgekürzt werden, etwa `RH`.

```
#(define RH rightHandFinger)
```

Ausgewählte Schnipsel

Positionierung von Fingersatz der rechten Hand

Man kann die Positionierung von Fingersatz der rechten Hand besser kontrollieren, wenn eine bestimmte Eigenschaft gesetzt wird, wie das folgende Beispiel zeigt:

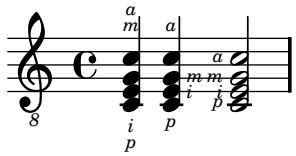
```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
```

```
  \set strokeFingerOrientations = #'(up down)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >4

  \set strokeFingerOrientations = #'(up right down)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >4

  \set strokeFingerOrientations = #'(left)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >2
}
```

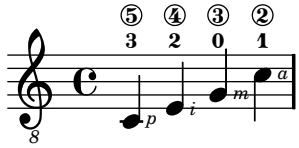


Fingersatz Saitennummern und Fingersatz für die rechte Hand

Dieses Beispiel kombiniert Fingersatz für die linke Hand, Saitennummern und Fingersatz für die rechte Hand.

```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  <c-3\5-\RH #1 >4
  <e-2\4-\RH #2 >4
  <g-0\3-\RH #3 >4
  <c-1\2-\RH #4 >4
}
```



Siehe auch

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Referenz der Interna: Abschnitt “StrokeFinger” in *Referenz der Interna*.
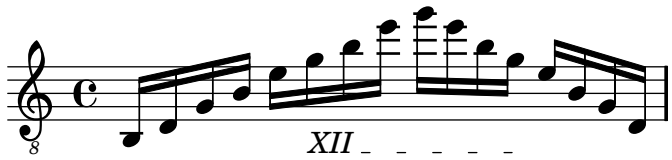
2.4.2 Gitarre

Die meisten der Besonderheiten von Gitarrennotation wurden im allgemeinen Abschnitt behandelt, aber es gibt noch einige, die hier gezeigt werden sollen. Teilweise soll ein Lead-sheet nur die Akkordsymbole und den Gesangstext enthalten. Da LilyPond ein Notensatzprogramm ist, wird es nicht für derartige Projekte empfohlen, die keine eigentliche Notation beinhalten. Anstatt dessen sollte ein Textbearbeitungsprogramm oder ein Satzprogramm wie GuitarTeX (für erfahrende Benutzer) eingesetzt werden.

Position und Barré anzeigen

Das Beispiel zeigt, wie man Griff- und Barréposition notieren kann.

```
\clef "treble_8"
b16 d g b e
\textSpannerDown
\override TextSpanner #'(bound-details left text) = #"XII "
g16\startTextSpan
b16 e g e b g\stopTextSpan
e16 b g d
```



Siehe auch

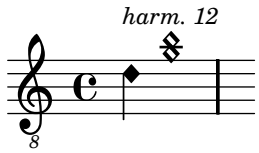
Notationsreferenz: [Text mit Verbindungslinien], Seite 202.

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*, Abschnitt “Expressive marks” in *Schnipsel*.

Flageolett und gedämpfte Noten

Besondere Notenköpfe können eingesetzt werden, um gedämpfte Noten oder Flageoletttöne anzuzeigen. Flageoletttöne werden normalerweise mit einem Text erklärt.

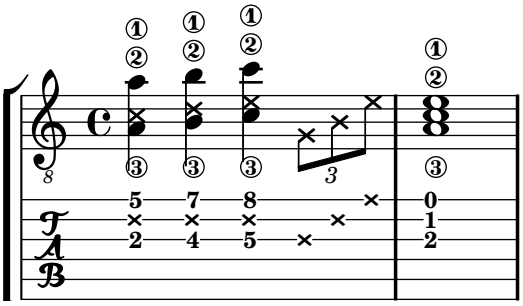
```
\relative c' {
  \clef "treble_8"
  \override Staff.NoteHead #'style = #'harmonic-mixed
  d^\markup { \italic { \fontsize #-2 { "harm. 12" }}} <g b>1
}
```



Gedämpfte oder gestoppte Noten werden in normalen und Tabulatur-Systemen unterstützt:

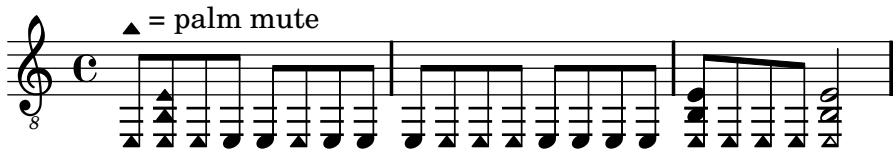
```
music = \relative c' {
  < a\3 \deadNote c\2 a'\1 >4
  < b\3 \deadNote d\2 b'\1 >
  < c\3 \deadNote e\2 c'\1 >
  \deadNotesOn
  \times 2/3 { g8 b e }
  \deadNotesOff
  < a,\3 c\2 e\1 >1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \music
  }
```

```
  \new TabStaff {
    \music
  }
>>
```



Eine andere Spieltechnik (insbesondere bei elektrischen Gitarren benutzt) ist *palm mute*. Hierbei wird die Saite teilweise durch die Handfläche der Schlaghand gedämpft. LilyPond unterstützt die Notation dieser Art von Technik, indem die Notenköpfe der so gedämpften Noten durch Dreiecke ersetzt werden.

```
\new Voice { % Warning: explicit Voice instantiation is
             %    required to have palmMuteOff work properly
             %    when palmMuteOn comes at the beginning of
             %    the piece.
  \relative c, {
    \clef "G_8"
    \palmMuteOn
    e8^\markup { \musicglyph #"noteheads.u2do"  = palm mute }
    < e b' e > e
    \palmMuteOff
    e e  \palmMute e e e |
    e8 \palmMute { e e e } e e e e |
    < \palmMute e b' e >8 \palmMute { e e e } < \palmMute e b' e >2
  }
}
```



Siehe auch

Notationsreferenz: [Besondere Notenköpfe], Seite 34, Abschnitt A.8 [Notenkopfstile], Seite 597.

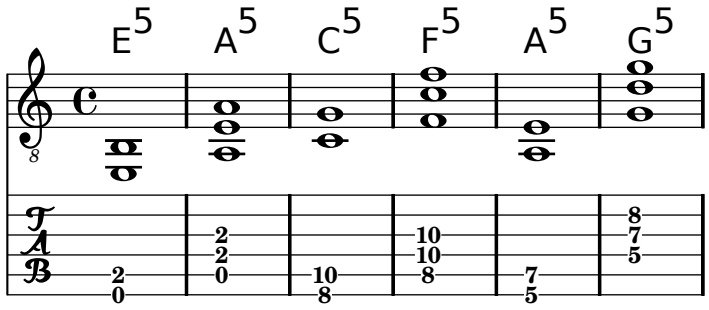
Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Powerakkorde anzeigen

Powerakkorde und ihre Symbole können im Akkordmodus oder als Akkordkonstruktionen gesetzt werden:

```
ChordsAndSymbols = {
  \chordmode {
    \powerChords
```

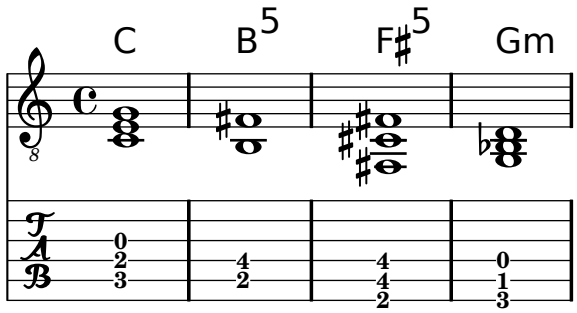
```
    e,,1:1.5
    a,,1:1.5.8
    \set minimumFret = #8
    c,1:1.5
    f,1:1.5.8
  }
  \set minimumFret = #5
  <a, e>1
  <g d' g'>1
}
\score {
  <<
    \new ChordNames {
    \ChordsAndSymbols
    }
    \new Staff {
      \clef "treble_8"
      \ChordsAndSymbols
    }
    \new TabStaff {
      \ChordsAndSymbols
    }
  >>
}
```



Powerakkord-Symbole werden automatisch ausgeschaltet, wenn einer der anderen normalen Akkord-Modifikatoren verwendet wird:

```
mixedChords = \chordmode {
  c,1
  \powerChords
  b,,1:1.5
  fis,,1:1.5.8
  g,,1:m
}
\score {
  <<
    \new ChordNames {
      \mixedChords
    }
    \new Staff {
      \clef "treble_8"
      \mixedChords
```

```
    }
    \new TabStaff {
      \mixedChords
    }
  >>
}
```



Siehe auch

Glossar: Abschnitt “power chord” in *Glossar*.

Notationsreferenz: [Erweiterte und modifizierte Akkorde], Seite 374, [Akkordbezeichnungen drucken], Seite 376.

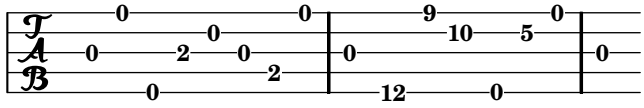
Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

2.4.3 Banjo

Banjo-Tabulaturen

LilyPond hat grundlegende Unterstützung für fünfsaitige Banjo. Die Banjo-Tabulatur-Funktion sollte zum Notieren von Banjo-Tabulaturen verwendet werden, damit die richtigen Bund-Nummern für die fünfte Saite gesetzt werden:

```
\new TabStaff <<
  \set TabStaff.tablatureFormat = #fret-number-tablature-format-banjo
  \set TabStaff.stringTunings = #banjo-open-g-tuning
  {
    \stemDown
    g8 d' g'\5 a b g e d' |
    g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
    g4
  }
>>
```



Eine Anzahl von üblichen Stimmungen für Banjo sind in LilyPond vordefiniert: `banjo-c-tuning` (gCGBD), `banjo-modal-tuning` (gDGCD), `banjo-open-d-tuning` (aDF#AD) und `banjo-open-dm-tuning` (aDFAD).

Diese Stimmungen können für das viersaitige Banjo angepasst werden, indem die `four-string-banjo`-Funktion eingesetzt wird:

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

Siehe auch

Schnipsel: Abschnitt “Fretted strings” in *Schnipsel*.

Installierte Dateien: ‘`scm/tablature.scm`’ enthält vordefinierte Banjo-Stimmungen.

2.5 Schlagzeug

2.5.1 Übliche Notation für Schlagzeug

Rhythmusnotation wird vor allem für Schlaginstrumente eingesetzt, aber hiermit kann auch der Rhythmus einer Melodie dargestellt werden.

Referenz für Schlagzeug

- Viele Schlagzeugmusik kann auf einem rhythmischen System notiert werden. Das wird gezeigt in [Melodierhythmus anzeigen], Seite 72 und [Neue Notensysteme erstellen], Seite 161.
- MIDI-Ausgabe wird behandelt in Abschnitt 3.5.6 [Schlagzeug in MIDI], Seite 459.

Siehe auch

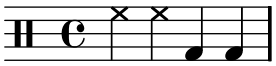
Notationsreferenz: [Melodierhythmus anzeigen], Seite 72, [Neue Notensysteme erstellen], Seite 161. Abschnitt 3.5.6 [Schlagzeug in MIDI], Seite 459.

Schnipsel: Abschnitt “Percussion” in *Schnipsel*.

Grundlagen der Schlagzeugnotation

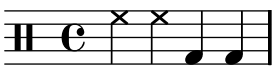
Schlagzeug-Noten können im `\drummode`-Modus notiert werden, der sich ähnlich verhält wie der Standardmodus für die Noteneingabe. Am einfachsten kann der `\drums`-Befehl benutzt werden, der sich um den richtigen Kontext und Eingabemodus kümmert:

```
\drums {
  hihat4 hh bassdrum bd
}
```



Das ist die Kurzschreibweise für:
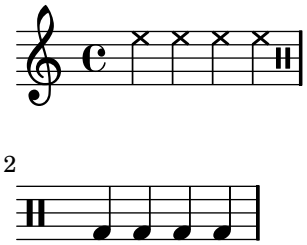
```
\new DrumStaff {
  \drummode {
    hihat4 hh bassdrum bd
  }
}
```



Jedes Schlagzeuginstrument hat einen langen Namen und eine Abkürzung, und beide können nebeneinander benutzt werden. Eine Liste der Notenbezeichnungen für Schlagzeug findet sich in Abschnitt A.12 [Schlagzeugnoten], Seite 641.

Beachten Sie, dass normale Tonhöhen (wie `cis4`) in einem `DrumStaff`-Kontext eine Fehlernachricht erzielen. Schlüssel für Schlagzeug werden automatisch hinzugefügt, aber sie können auch explizit gesetzt Werden. Auch andere Schlüssel können benutzt werden.

```
\drums {
  \clef treble
  hh4 hh hh hh
  \break
  \clef percussion
  bd4 bd bd bd
}
```



Es gibt einige Probleme mit der MIDI-Unterstützung für Schlagzeuginstrumente. Details finden sich in Abschnitt 3.5.6 [Schlagzeug in MIDI], Seite 459.

Siehe auch

Notationsreferenz: Abschnitt 3.5.6 [Schlagzeug in MIDI], Seite 459, Abschnitt A.12 [Schlagzeugnoten], Seite 641.
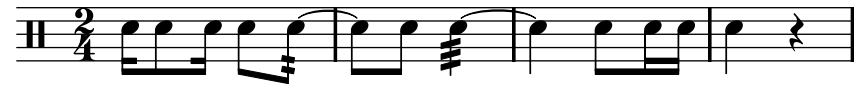
Installierte Dateien: ‘`ly/drumpitch-init.ly`’.

Schnipsel: Abschnitt “Percussion” in *Schnipsel*.

Trommelwirbel
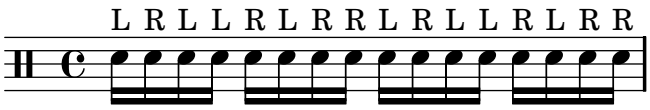
Trommelwirbel werden mit drei Balken durch den Notenhals notiert. Für Viertelnoten oder längere Noten werden die drei Balken explizit notiert, Achtel werden mit zwei Balken gezeigt (und der dritte ist der eigentliche Balken), und Trommelwirbel mit kürzeren Werten als Achtelnoten haben einen Balken zusätzlich zu den eigentlichen Balken der Noten. Dieses Verhalten wird mit der Tremolonotation erreicht, wie in [Tremolo-Wiederholung], Seite 140 gezeigt. Hier ein Beispiel kleinerer Wirbel:

```
\drums {
  \time 2/4
  sn16 sn8 sn16 sn8 sn8:32 ~
  sn8 sn8 sn4:32 ~
  sn4 sn8 sn16 sn16
  sn4 r4
}
```



Benutzung der Stöcke kann angezeigt werden durch `^"R"` oder `^"L"` nach jeder Note. Die `staff-padding`-Eigenschaft kann verändert werden, um eine Orientierung an einer gemeinsamen Linie zu ermöglichen.

```
\drums {
  \repeat unfold 2 {
    sn16 ^"L" sn^"R" sn^"L" sn^"L" sn^"R" sn^"L"  sn^"R" sn^"R"
  }
}
```



Siehe auch

Schnipsel: Abschnitt “Percussion” in *Schnipsel*.

Schlagzeug mit Tonhöhe

Bestimmte Schlagzeuginstrumente mit Tonhöhe (z. B. Xylophone, vibraphone und Pauken) werden auf normalen Systemen geschrieben. Das wird in anderen Abschnitten des Handbuchs behandelt.

Siehe auch

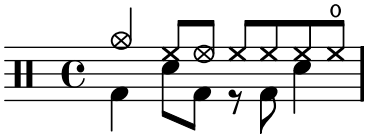
Notationsreferenz: Abschnitt 3.5.6 [Schlagzeug in MIDI], Seite 459.

Schnipsel: Abschnitt “Percussion” in *Schnipsel*.

Schlagzeugsysteme

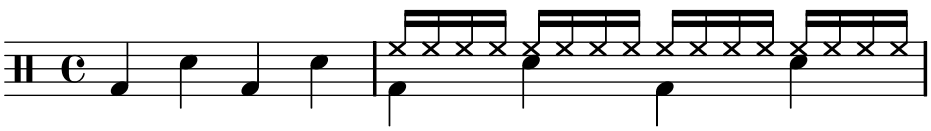
Ein Schlagzeug-System besteht üblicherweise aus einem Notensystem mit mehreren Linien, wobei jede Linie ein bestimmtes Schlagzeug-Instrument darstellt. Um die Noten darstellen zu können, müssen sie sich innerhalb von einem `DrumStaff`- und einem `DrumVoice`-Kontext befinden.

```
up = \drummode {
  crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat
}
down = \drummode {
  bassdrum4 snare8 bd r bd sn4
}
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



Das Beispiel zeigt ausdrücklich definierte mehrstimmige Notation. Die Kurznotation für mehrstimmige Musik, wie sie im Abschnitt Abschnitt “Ich höre Stimmen” in *Handbuch zum Lernen* beschrieben wird, kann auch verwendet werden.

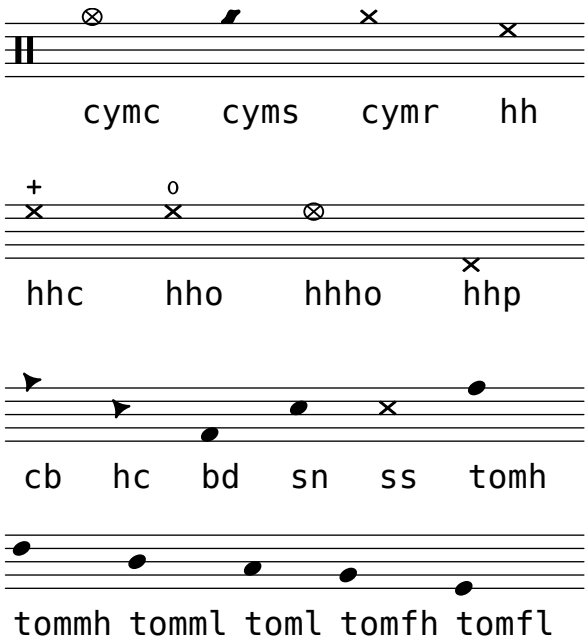
```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
      \repeat unfold 16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    } >>
  }
>>
```



Es gibt auch weitere Layout-Einstellungen. Um diese zu verwenden, muss die Eigenschaft `drumStyleTable` im `DrumVoice`-Kontext entsprechend eingestellt werden. Folgende Variablen sind vordefiniert:

`drums-style`

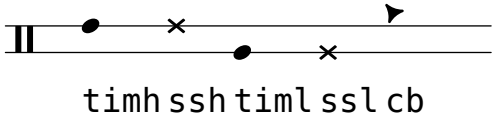
Das ist die Standardeinstellung. Hiermit wird ein typisches Schlagzeug-System auf fünf Notenlinien erstellt.



Die Schlagzeugdefinitionen unterstützen sechs unterschiedliche Tom Toms. Falls eine geringere Anzahl verwendet wird, kann man einfach die Tom Toms auswählen, deren Notation man haben will. Tom Toms auf den drei mittleren Linien werden mit den Bezeichnungen `tommh`, `tomml` und `tomfh` notiert.

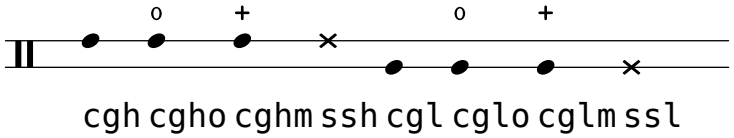
`timbales-style`

Hiermit werden Timbale auf zwei Notenlinien gesetzt.



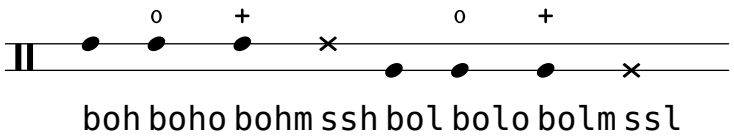
`congas-style`

Hiermit werden Congas auf zwei Linien gesetzt.



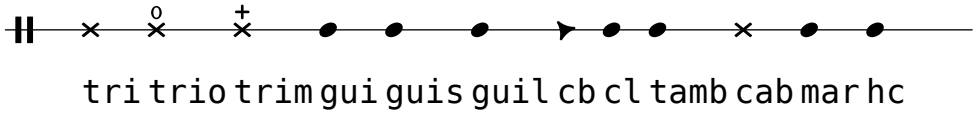
`bongos-style`

Hiermit werden Bongos auf zwei Linien gesetzt.



`percussion-style`

Dieser Stil ist für alle einfachen Perkussionsinstrumente auf einer Notenlinie.

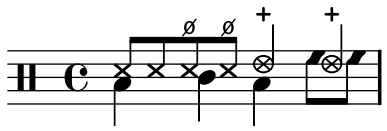


Eigene Schlagzeugsysteme

Wenn ihnen keine der vordefinierten Stile gefällt, können Sie auch eine eigene Liste der Positionen und Notenköpfe am Anfang ihrer Datei erstellen.

```
#(define mydrums '(
         (bassdrum        default   #f           -1)
         (snare           default   #f           0)
         (hihat           cross     #f           1)
         (halfopenhihat   cross     "halfopen"   1)
         (pedalhihat      xcircle   "stopped"    2)
         (lowtom          diamond   #f           3)))
up = \drummode { hh8 hh hhho hhho hhp4 hhp }
down = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff <<
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



Ausgewählte Schnipsel

Hier einige Beispiele:

Zwei Holzblöcke, notiert mit wbh (hoch) und wbl (tief)

```
% These lines define the position of the woodblocks in the stave;
% if you like, you can change it or you can use special note heads
% for the woodblocks.
#(define mydrums '((hiwoodblock default #t  3)
                   (lowoodblock default #t -2)))

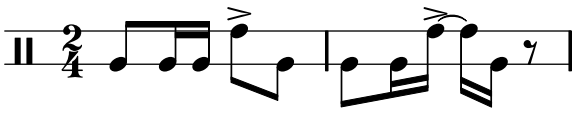
woodstaff = {
  % This defines a staff with only two lines.
  % It also defines the positions of the two lines.
  \override Staff.StaffSymbol #'line-positions = #'(-2 3)

  % This is necessary; if not entered, the barline would be too short!
  \override Staff.BarLine #'bar-extent = #'(-1.5 . 1.5)
}

\new DrumStaff {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
```

```
  % with this you load your new drum style table
  \woodstaff

  \drummode {
    \time 2/4
    wbl8 wbl16 wbl wbh8-> wbl |
    wbl8 wbl16 wbh-> ~ wbh wbl16 r8 |
  }
}
```



In diesem Spezialfalls muss die Länge der Taktlinie mit `\override Staff.BarLine #'bar-extent #'(von . bis)` angepasst werden. Andernfalls wäre sie zu kurz. Die Position der beiden Linien muss auch definiert werden.

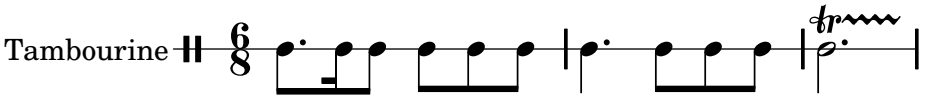
Tamburin, notiert mit `tamb`:

```
#(define mydrums '((tambourine default #t 0)))

tambustaff = {
  \override Staff.StaffSymbol #'line-positions = #'( 0 )
  \override Staff.BarLine #'bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Tambourine"
}

\new DrumStaff {
  \tambustaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    \time 6/8
    tamb8. tamb16 tamb8 tamb tamb tamb |
    tamb4. tamb8 tamb tamb |
    % the trick with the scaled duration and the shorter rest
    % is neccessary for the correct ending of the trill-span!
    tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
  }
}
```



Noten für Tam-Tam (notiert mit `tt`):

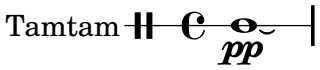
```
#(define mydrums '((tamtam default #t 0)))

tamtamstaff = {
  \override Staff.StaffSymbol #'line-positions = #'( 0 )
  \override Staff.BarLine #'bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Tamtam"
```

```
}

\new DrumStaff {
  \tamtamstaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    tt 1 \pp \laissezVibrer
  }
}
```

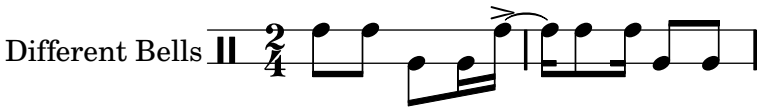


Zwei Glocken, notiert mit `cb` (Kuhglocke) und `rb` (Reiterglocke)

```
#(define mydrums '((ridebell default #t  3)
                   (cowbell  default #t -2)))

bellstaff = {
  \override DrumStaff.StaffSymbol #'line-positions = #'(-2 3)
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.BarLine #'bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Different Bells"
}

\new DrumStaff {
  \bellstaff
  \drummode {
    \time 2/4
    rb8 rb cb cb16 rb-> ~ |
    rb16 rb8 rb16 cb8 cb |
  }
}
```



Hier ein kurzes Beispiel von Stravinsky (aus „L'histoire du Soldat“):

```
#(define mydrums '((bassdrum   default #t  4)
                   (snare      default #t -4)
                   (tambourine default #t  0)))

global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}

drumsA = {
  \context DrumVoice <<
```
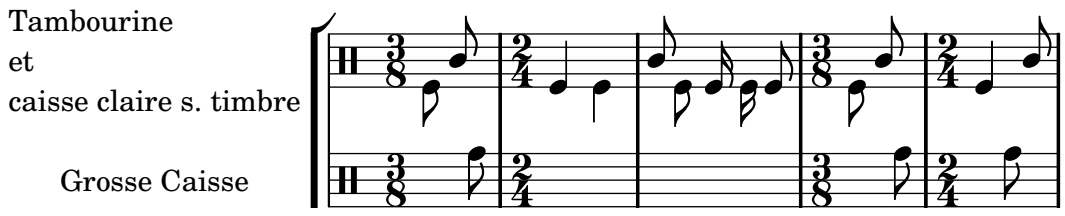
```
    { \global }
    { \drummode {
        \autoBeamOff
        \stemDown sn8 \stemUp tamb s8 |
        sn4 \stemDown sn4 |
        \stemUp tamb8 \stemDown sn8 \stemUp sn16 \stemDown sn \stemUp sn8 |
        \stemDown sn8 \stemUp tamb s8 |
        \stemUp sn4 s8 \stemUp tamb
      }
    }
  >>
}

drumsB = {
  \drummode {
    s4 bd8 s2*2 s4 bd8 s4 bd8 s8
  }
}

\layout {
  indent = #40
}

\score {
  \new StaffGroup <<
    \new DrumStaff {
      \set DrumStaff.instrumentName = \markup {
        \column {
          "Tambourine"
          "et"
          "caisse claire s. timbre"
        }
      }
      \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
      \drumsA
    }

   \new DrumStaff {
     \set DrumStaff.instrumentName = #"Grosse Caisse"
     \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
     \drumsB }
  >>
}
```



Siehe auch

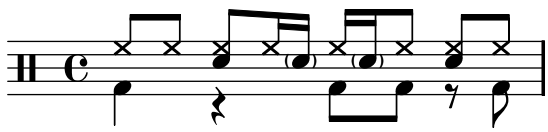
Schnipsel: Abschnitt “Percussion” in *Schnipsel*.

Referenz der Interna: Abschnitt “DrumStaff” in *Referenz der Interna*, Abschnitt “DrumVoice” in *Referenz der Interna*.

Geisternoten

Geisternoten für Schlagzeug und Perkussion können mit dem Klammer- (`\parenthesize`)-Befehl, beschrieben in [Klammern], Seite 195, erstellt werden. Im Standard-`\drummode`-Modus ist aber das `Parenthesis_engraver`-Plugin nicht automatisch enthalten.

```
\new DrumStaff \with {
  \consists "Parenthesis_engraver"
}
<<
  \context DrumVoice  = "1" { s1 }
  \context DrumVoice  = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ hh] <hh sn> hh16
        < \parenthesize sn > hh
        < \parenthesize sn > hh8 <hh sn> hh
      } \\
      {
        bd4 r4 bd8 bd r8 bd
      }
    >>
  }
>>
```

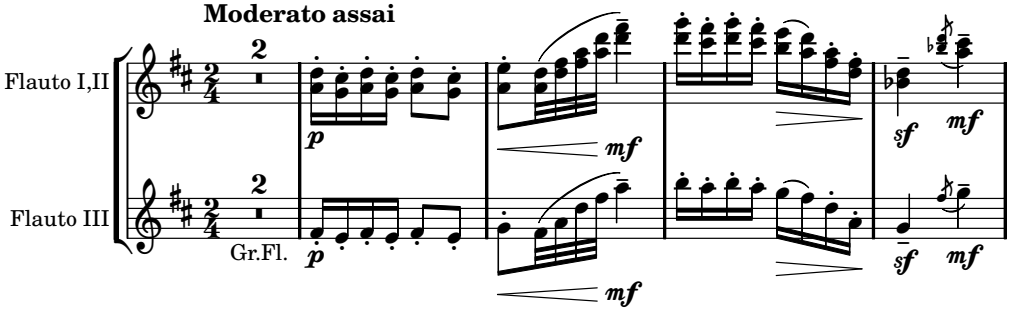


Um jede Klammer-Definition (`\parenthesize`) müssen zusätzlich die spitzen Klammern für Akkorde (`< >`) gesetzt werden.

Siehe auch

Schnipsel: Abschnitt “Percussion” in *Schnipsel*.

2.6 Blasinstrumente



Dieser Abschnitt beinhaltet einige Notationselemente, die bei der Notation von Blasinstrumenten auftreten.

2.6.1 Übliche Notation für Bläser

Dieser Abschnitt erklärt Notation, die für die meisten Blasinstrumente gültig sind.

Referenz für Blasinstrumente

Viele Besonderheiten der Blasinstrumentenotation haben mit Atmung und Spielart zu tun:

- Atmung kann durch Pausen oder mit Atemzeichen angezeigt werden,, siehe [Atemzeichen], Seite 120.
- Legato kann durch Legatobögen angezeigt werden, siehe [Legatobögen], Seite 116.
- Unterschiedliche Artikulationen, Legato, Portato, Staccato, werden normalerweise mit Artikulationszeichen angemerkt, teilweise auch in Verbindung mit Legatobögen, siehe [Artikulationszeichen und Verzierungen], Seite 106 und Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640.
- Flatterzunge wird angezeigt, indem ein Tremolozeichen und eine Anmerkung für die entsprechende Note gesetzt wird. Siehe [Tremolo-Wiederholung], Seite 140.

Es gibt auch noch weitere Aspekte der Notation, die für Blasinstrumente relevant sein können:

- Viele Instrumente sind transponierend, siehe [Transposition von Instrumenten], Seite 22.
- Das Zug-Glissando ist charakteristisch für die Posaune, aber auch andere Instrumente können Glissandos ausführen. Siehe [Glissando], Seite 122.
- Obertonreihenglissandi, die auf allen Blechblasinstrumenten möglich, aber besonders üblich für das Waldhorn sind, werden üblicherweise mit Verzierungsnoten geschrieben. Siehe [Verzierungen], Seite 99.
- Tonhöhenschwankungen am Ende eines Tons werden gezeigt in [Glissando zu unbestimmter Tonhöhe], Seite 121.
- Ventil- oder Klappenschläge werden oft als Kreuznoten dargestellt, siehe [Besondere Notenköpfe], Seite 34.
- Holzbläser können tiefe Noten überblasen. Derartige Noten werden als `flageolet`-Artikulation notiert. Siehe Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640.
- Die Benutzung von Dämpfern für Blechblasinstrumente wird meistens durch Text gefordert, aber bei schnellem Wechsel bietet es sich an, die Artikulationszeichen `stopped` und `open` zu benutzen. Siehe [Artikulationszeichen und Verzierungen], Seite 106 und Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640.
- Gestopfte Hörner werden mit dem `stopped`-Artikulationszeichen notiert. Siehe [Artikulationszeichen und Verzierungen], Seite 106.

Ausgewählte Schnipsel

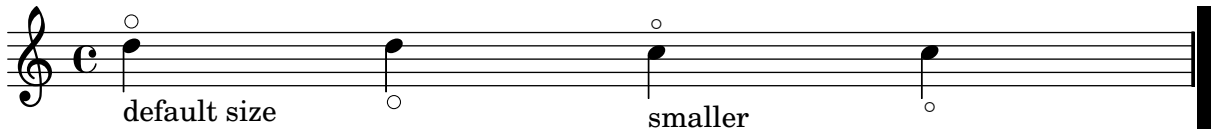
\flageolet-Zeichen verkleinern

Um den `\flageolet`-Kreis kleiner zu machen, kann diese Scheme-Funktion eingesetzt werden.

```
smallFlageolet =
#(let ((m (make-articulation "flageolet")))
   (set! (ly:music-property m 'tweaks)
         (acons 'font-size -3
                (ly:music-property m 'tweaks)))
   m)

\layout { ragged-right = ##f }
```

```
\relative c'' {
  d4^\flageolet_\markup { default size } d_\flageolet
  c4^\smallFlageolet_\markup { smaller } c_\smallFlageolet
}
```



Siehe auch

Notationsreferenz: [Atemzeichen], Seite 120, [Legatobögen], Seite 116, [Artikulationszeichen und Verzierungen], Seite 106, Abschnitt A.11 [Liste der Artikulationszeichen], Seite 640, [Tremolo-Wiederholung], Seite 140, [Transposition von Instrumenten], Seite 22, [Glissando], Seite 122, [Verzierungen], Seite 99, [Glissando zu unbestimmter Tonhöhe], Seite 121, [Besondere Notenköpfe], Seite 34,

Schnipsel: Abschnitt “Winds” in *Schnipsel*.

Fingersatz

Alle Blasinstrumente außer der Posaune benötigen mehrere Finger, um verschiedene Tonhöhen zu produzieren. Einige Fingersatzbeispiele zeigen die Schnipsel unten.

Diagramme für Holzbläser können erstellt werden nach den Anweisungen in Abschnitt 2.6.3.1 [Holzbläserdiagramme], Seite 363.

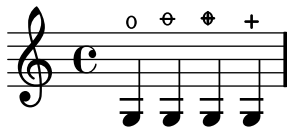
Ausgewählte Schnipsel

Fingering symbols for wind instruments

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

```
centermarkup = {
  \once \override TextScript #'self-alignment-X = #CENTER
  \once \override TextScript #'X-offset =#(ly:make-simple-closure
    `(,+
      ,(ly:make-simple-closure (list
        ly:self-alignment-interface::centered-on-x-parent))
      ,(ly:make-simple-closure (list
        ly:self-alignment-interface::x-aligned-on-self))))
}
\score
{\relative c'
  {
    g\open
    \once \override TextScript #'staff-padding = #-1.0 \centermarkup
    g^\markup{\combine \musicglyph #"scripts.open" \musicglyph
    #"scripts.tenuto"}
    \centermarkup g^\markup{\combine \musicglyph #"scripts.open"
    \musicglyph #"scripts.stopped"}
    g\stopped
  }
```

```
}
```



Recorder fingering chart

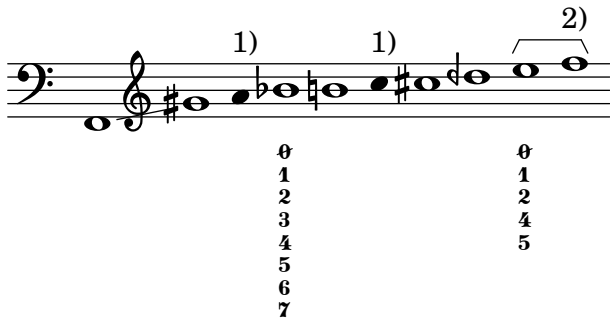
The following example demonstrates how fingering charts for wind instruments can be realized.

```
% range chart for paetzold contrabass recorder

centermarkup = {
  \once \override TextScript #'self-alignment-X = #CENTER
  \once \override TextScript #'X-offset =#(ly:make-simple-closure
  `(,+
  ,(ly:make-simple-closure (list
  ly:self-alignment-interface::centered-on-x-parent))
  ,(ly:make-simple-closure (list
  ly:self-alignment-interface::x-aligned-on-self))))
}

\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
    \override Stem #'stencil = ##f
    \consists "Horizontal_bracket_engraver"
  }
  {
    \clef bass
    \set Score.timing = ##f
    f,1*1/4 \glissando
    \clef violin
    gis'1*1/4
    \stemDown a'4^\markup{1)}
    \centermarkup
    \once \override TextScript #'padding = #2
    bes'1*1/4_\markup{\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 3 \finger 4
    \finger 5 \finger 6 \finger 7} }
    b'1*1/4
    c''4^\markup{1)}
    \centermarkup
    \once \override TextScript #'padding = #2
    cis''1*1/4
    deh''1*1/4
    \centermarkup
    \once \override TextScript #'padding = #2
    \once \override Staff.HorizontalBracket #'direction = #UP
    e''1*1/4_\markup{\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 4
```

```
    \finger 5} }\startGroup
    f''1*1/4^\markup{2)}\stopGroup
  }
}
```



Siehe auch

Notationsreferenz: Abschnitt 2.6.3.1 [Holzbläserdiagramme], Seite 363.

Snippets: Abschnitt “Winds” in *Schnipsel*.

2.6.2 Dudelsack

Dieser Abschnitt beinhaltet die Notation von Dudelsackmusik.

Dudelsack-Definitionen

LilyPond besitzt spezielle Definitionen, mit der die Musik des schottischen Hochland-Dudelsacks notiert wird. Um sie zu benutzen, muss

```
\include "bagpipe.ly"
```
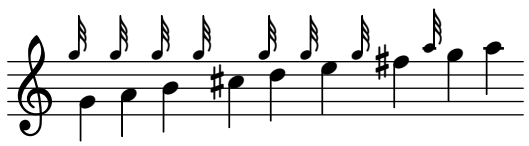
am Anfang der LilyPond-Quelldatei eingefügt werden. Hierdurch können dann bestimmte Verzierungsnoten, die für die Dudelsackmusik üblich sind, mit kurzen Befehlen eingefügt werden. So reicht etwa der Befehl `\taor`, anstatt

```
\grace { \small G32[ d G e] }
```

zu schreiben.

‘`bagpipe.ly`’ enthält außerdem Definitionen für Tonhöhen von Dudelsacknoten in bestimmten Oktaven, so dass man sich nicht mehr um `\relative` oder `\transpose` kümmern muss.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



Musik für den Dudelsack wird in D-Dur geschrieben (auch wenn das eigentlich nicht stimmt). Weil das aber die einzige Tonart ist, die benutzt werden kann, werden die Vorzeichen meistens nicht geschrieben. Damit das funktioniert, müssen die Noten immer mit `\hideKeySignature` beginnen. Wenn die Vorzeichen hingegen angezeigt werden sollen, kann das mithilfe des Befehls `\showKeySignature` vorgenommen werden.

Einige moderne Dudelsacknoten benutzen halbe Finger auf c und f, um diese Noten zu erniedrigen. Das kann angezeigt werden mit `cflat` bzw. `fflat`. Gleicherweise kann das piobaireachd hohe g als `gflat` geschrieben werden, wenn es in leichter Musik vorkommt.

Siehe auch

Schnipsel: Abschnitt “Winds” in *Schnipsel*.
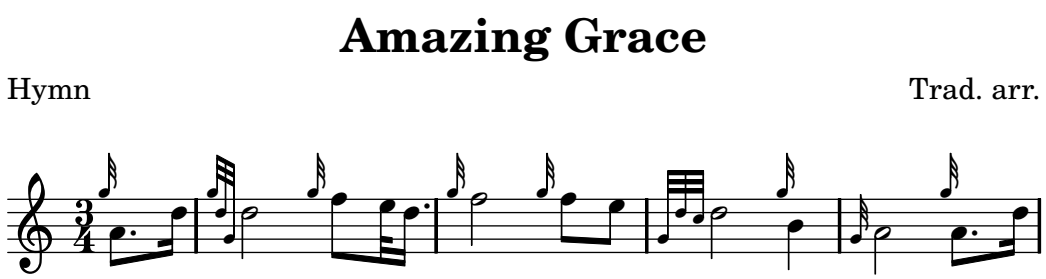
Dudelsack-Beispiele
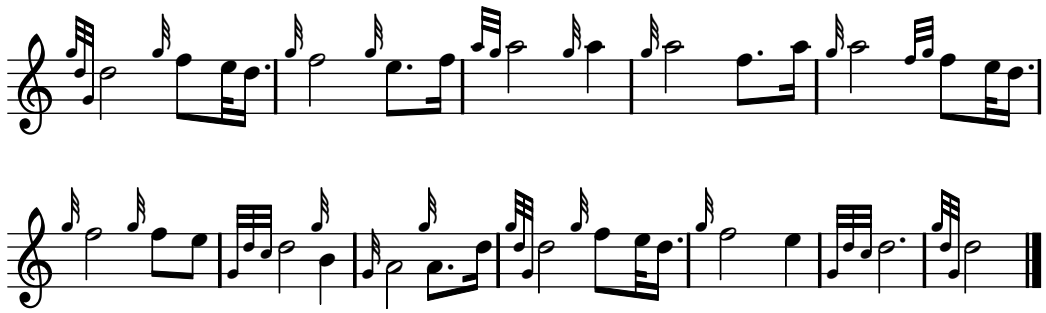
So sieht die bekannte Melodie Amazing Grace aus, wenn man sie für Dudelsack notiert.

```
\include "bagpipe.ly"
\layout {
  indent = 0.0\cm
  \context { \Score \remove "Bar_number_engraver" }
}

\header {
  title = "Amazing Grace"
  meter = "Hymn"
  arranger = "Trad. arr."
}

{
  \hideKeySignature
  \time 3/4
  \grg \partial 4 a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg e8. f16
  \dblA A2 \grg A4
  \grg A2 f8. A16
  \grg A2 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
  \thrwd d2.
  \slurd d2
  \bar "|."
}
```





Siehe auch

Schnipsel: Abschnitt “Winds” in *Schnipsel*.

2.6.3 Holzbläser

Dieser Abschnitt zeigt Notation, die spezifisch für Holzbläser ist.

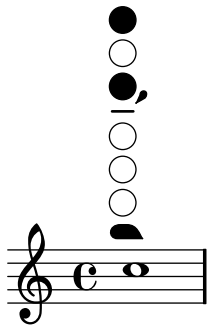
2.6.3.1 Holzbläserdiagramme

Holzbläserdiagramme können benutzt werden, um die Griffe für eine bestimmte Note darzustellen. Diagramme gibt es für folgende Instrumente:

- Piccolo
- Flöte
- Oboe
- Clarinette
- BassClarinette
- Saxophon
- Fagott
- Kontrafagott

Holzbläserdiagramme werden als Beschriftung erstellt:

```
c1^\markup {
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                 (cc . (one three))
                                 (rh . (ees)))
}
```



Löcher können offen, halboffen, Ring oder geschlossen sein:

```
\textLengthOn
c1^\markup {
  \center-column {
    "Ein Viertel"
    \woodwind-diagram #'flute #'((cc . (one1q))
                                 (lh . ())
```

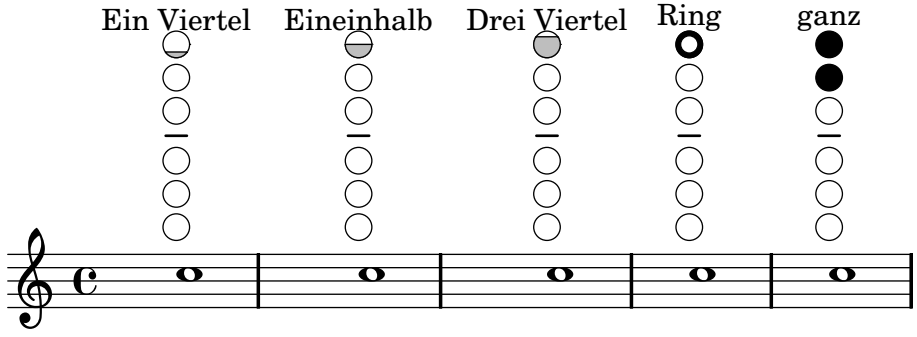
```
                                  (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "Eineinhalb"
    \woodwind-diagram #'flute #'((cc . (one1h))
                                  (lh . ())
                                  (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "Drei Viertel"
    \woodwind-diagram #'flute #'((cc . (one3q))
                                  (lh . ())
                                  (rh . ()))
  }
}

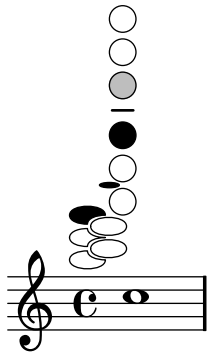
c1^\markup {
  \center-column {
    "Ring"
    \woodwind-diagram #'flute #'((cc . (oneR))
                                  (lh . ())
                                  (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "ganz"
    \woodwind-diagram #'flute #'((cc . (oneF two))
                                  (lh . ())
                                  (rh . ()))
  }
}
```



Triller werden als schattierte Löcher in den Diagrammen angezeigt:

```
c1^\markup {
  \woodwind-diagram #'bass-clarinet
                    #'((cc . (threeT four))
                       (lh . ())
                       (rh . (b fis)))
}
```



Eine Vielzahl von Trillern ist möglich:

```
\textLengthOn
c1^\markup {
  \center-column {
    "ein Viertel zu Ring"
    \woodwind-diagram #'flute #'((cc . (one1qTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

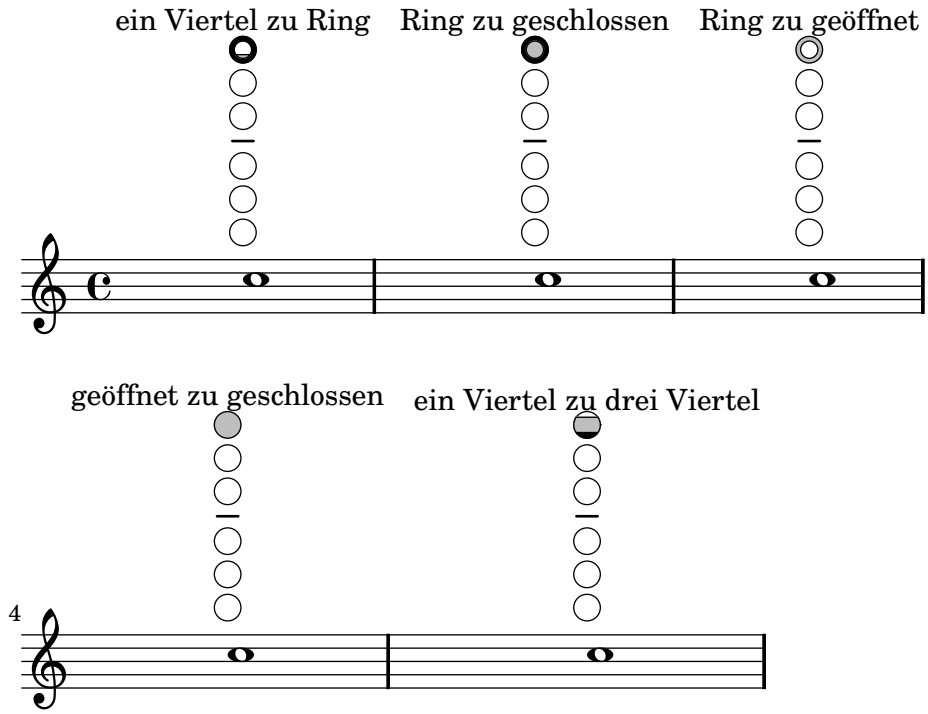
c1^\markup {
  \center-column {
    "Ring zu geschlossen"
    \woodwind-diagram #'flute #'((cc . (oneTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "Ring zu geöffnet"
    \woodwind-diagram #'flute #'((cc . (oneRT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "geöffnet zu geschlossen"
    \woodwind-diagram #'flute #'((cc . (oneT))
                                 (lh . ())
```

```
                                (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "ein Viertel zu drei Viertel"
    \woodwind-diagram #'flute #'((cc . (one1qT3q))
                                 (lh . ())
                                 (rh . ()))
  }
}
```



Die Liste aller möglichen Löcher und Einstellungen eines bestimmten Instruments kann auf der Kommandozeile oder in einer Log-Datei angezeigt werden, auch wenn man sie nicht in der Notenausgabe anzeigen lassen kann:

```
#(print-keys-verbose 'flute)
```

Neue Diagramme können erstellt werden, indem man die Muster in den Datei ‘`scm/define-woodwind-diagrams.scm`’ und ‘`scm/display-woodwind-diagrams.scm`’ befolgt. Das benötigt jedoch Scheme-Fähigkeit und ist nicht für alle Benutzer verständlich.

Ausgewählte Schnipsel

Liste der Holzbläserdiagramme

Folgende Noten zeige alle Holzbläserdiagramme, die für LilyPond definiert sind.

```
\relative c' {
  \textLengthOn
  c1^
  \markup {
```

```
  \center-column {
    'piccolo
    " "
     \woodwind-diagram
                #'piccolo
                #'()
  }
}

c1^
\markup {
  \center-column {
     'flute
     " "
     \woodwind-diagram
        #'flute
        #'()
  }
}
c1^\markup {
  \center-column {
    'oboe
    " "
    \woodwind-diagram
      #'oboe
      #'()
  }
}

c1^\markup {
  \center-column {
    'clarinet
    " "
    \woodwind-diagram
      #'clarinet
      #'()
  }
}

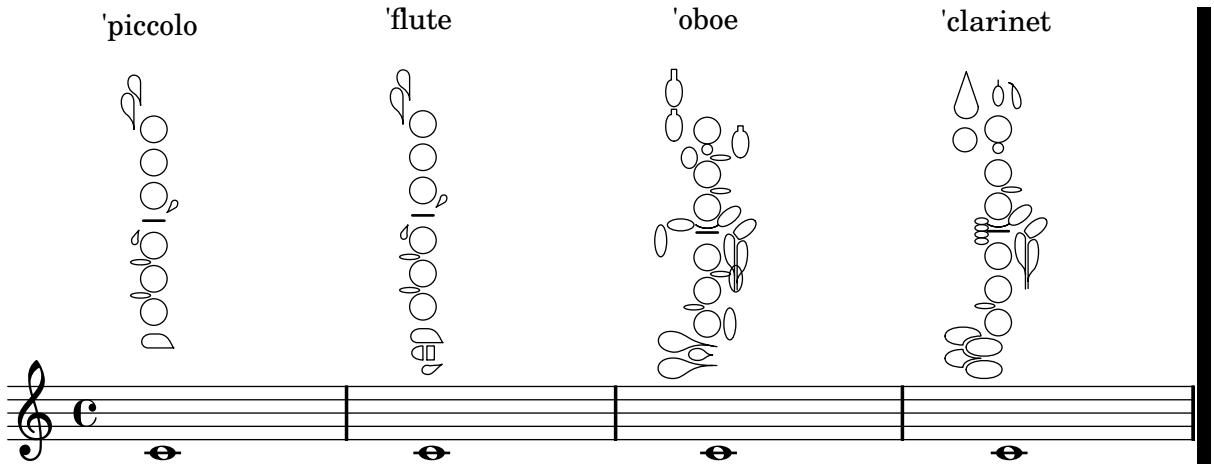
c1^\markup {
  \center-column {
    'bass-clarinet
    " "
    \woodwind-diagram
      #'bass-clarinet
      #'()
  }
}

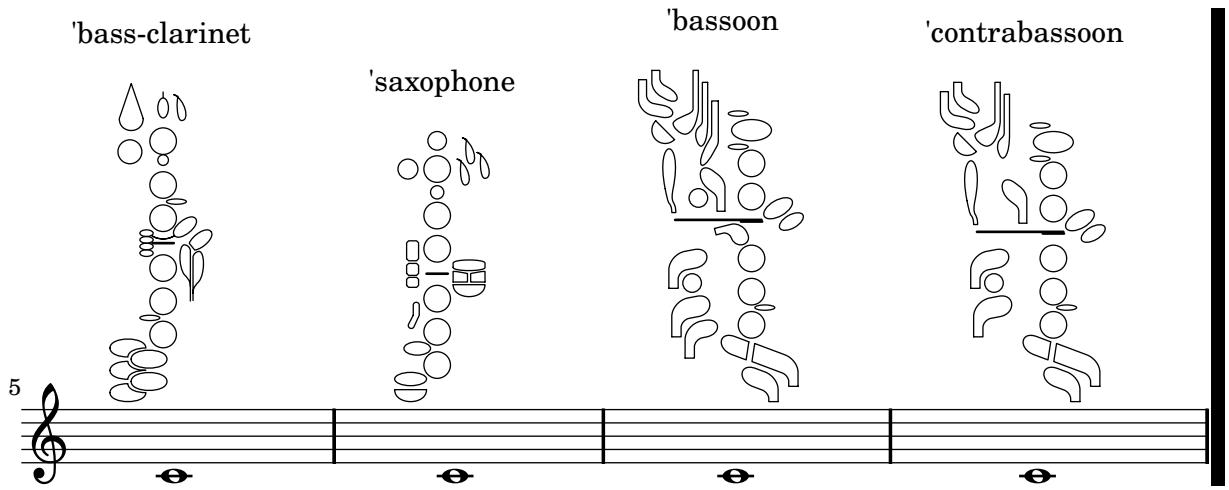
c1^\markup {
  \center-column {
    'saxophone
```

```
      " "
      \woodwind-diagram
        #'saxophone
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'bassoon
      " "
      \woodwind-diagram
        #'bassoon
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'contrabassoon
      " "
      \woodwind-diagram
        #'contrabassoon
        #'()
    }
  }
}
```



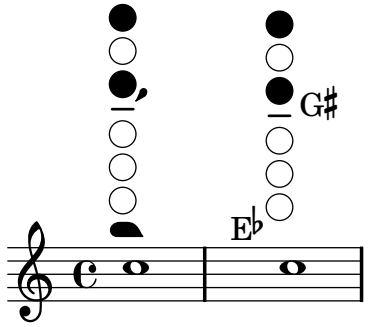


Graphische und Text-Holzbläserdiagramme

In vielen Fällen können die nicht in der mittleren Reihe befindlichen Löcher dargestellt werden, indem man die Lochbezeichnung oder graphische Zeichen benutzt.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
      #'((cc . (one three))
         (lh . (gis))
         (rh . (ees)))

  c^\markup
    \override #'(graphical . #f) {
      \woodwind-diagram
        #'piccolo
        #'((cc . (one three))
           (lh . (gis))
           (rh . (ees)))
    }
}
```



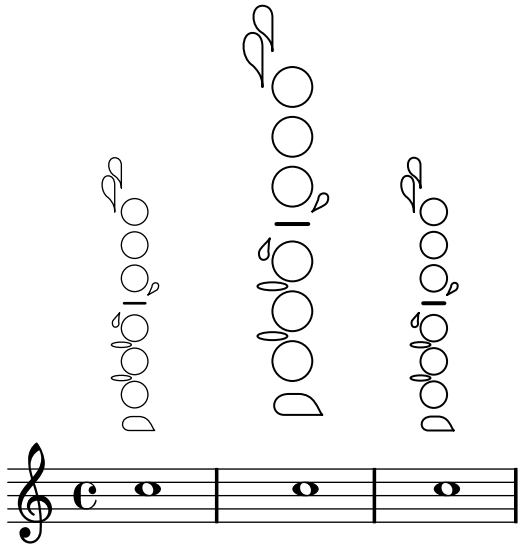
Größe von Holzbläserdiagrammen ändern

Die Größe und Dicke der Holzbläserdiagramme kann geändert werden.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
```

```
      #'()

  c^\markup
    \override #'(size . 1.5) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
  c^\markup
    \override #'(thickness . 0.15) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
}
```



Liste der Löcher für Holzbläserdiagramme

Dieses Schnipsel erzeugt eine Liste aller möglichen Löcher und Locheinstellungen für Holzbläserdiagramme, wie sie in der Datei ‘`scm/define-woodwind-diagrams.scm`’ definiert sind. Die Liste wird auf der Kommandozeile und in der Log-Datei angezeigt, nicht in den Noten.

```
#(print-keys-verbose 'piccolo)
#(print-keys-verbose 'flute)
#(print-keys-verbose 'flute-b-extension)
#(print-keys-verbose 'oboe)
#(print-keys-verbose 'clarinet)
#(print-keys-verbose 'bass-clarinet)
#(print-keys-verbose 'low-bass-clarinet)
#(print-keys-verbose 'saxophone)
#(print-keys-verbose 'baritone-saxophone)
#(print-keys-verbose 'bassoon)
#(print-keys-verbose 'contrabassoon)
```
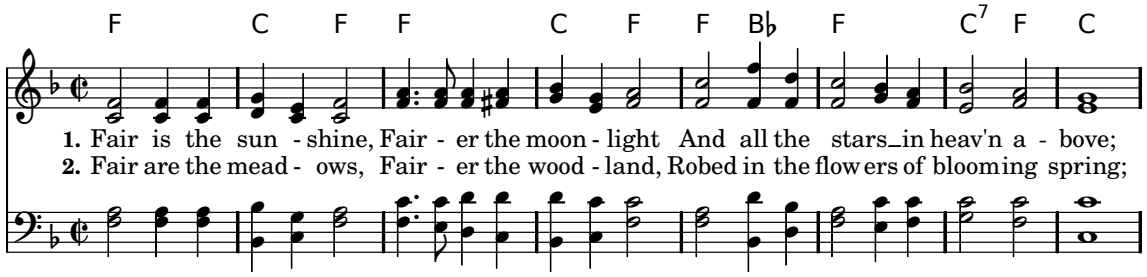
Siehe auch

Installierte Dateien: ‘`scm/define-woodwind-diagrams.scm`’, ‘`scm/display-woodwind-diagrams.scm`’.

Schnipsel: Abschnitt “Winds” in *Schnipsel*.

Referenz der Interna: Abschnitt “TextScript” in *Referenz der Interna*, Abschnitt “instrument-specific-markup-interface” in *Referenz der Interna*.

2.7 Notation von Akkorden



Akkorde können entweder als normale Noten oder im Akkordmodus notiert werden; bei letztere Eingabemethode können unterschiedliche europäische Akkordbezeichnungsstile eingesetzt werden. Akkordbezeichnungen und Generalbass können auch angezeigt werden.

2.7.1 Akkord-Modus
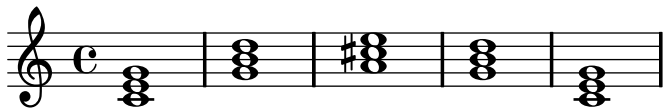
Im Akkordmodus (engl. „chord“) werden Akkorde anhand von einem Symbol der erwünschten Akkordstruktur notiert, anstatt dass die einzelnen Tonhöhen ausgeschrieben werden.

Überblick über den Akkord-Modus

Akkorde können als simultane Noten eingegeben werden, wie gezeigt in [Noten mit Akkorden], Seite 142.

Akkorde können aber auch im Akkordmodus notiert werden. Das ist ein Eingabemodus, der sich an Akkordstrukturen traditioneller europäischer Musik und nicht an bestimmten einzelnen Tonhöhen orientiert. Er bietet sich an, wenn man es gewohnt ist, Akkordsymbole zur Beschreibung von Akkorden zu benutzen. Mehr Information zu unterschiedlichen Eingabemethoden findet sich in Abschnitt 5.4.1 [Eingabe-Modi], Seite 536.

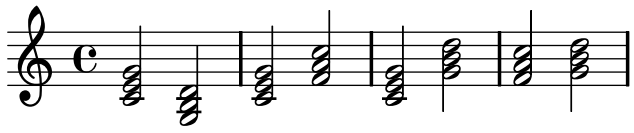
```
\chordmode { c1 g a g c }
```



Akkorde, die im Akkordmodus eingegeben werden, sind musikalische Elemente und können genauso wie Akkorde im Notenmodus transponiert werden. `\chordmode` ist absolut, und deshalb hat `\relative` keine Auswirkung auf die `\chordmode`-Abschnitte. Im Akkord-Modus ist jedoch die absolute Tonhöhe eine Oktave höher als im Notationsmodus.

Akkordmodus und Notenmodus können gemischt verwendet werden:

```
<c e g>2 <g b d>
\chordmode { c2 f }
<c e g>2 <g' b d>
\chordmode { f2 g }
```



Siehe auch

Glossar: Abschnitt “chord” in *Glossar*.

Notationsreferenz: [Noten mit Akkorden], Seite 142, Abschnitt 5.4.1 [Eingabe-Modi], Seite 536.
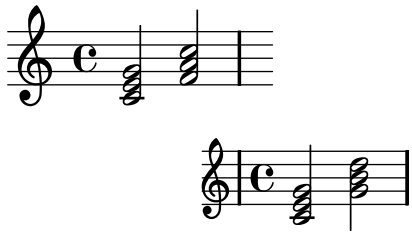
Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Bekannte Probleme und Warnungen

Vordefinierte Abkürzung für Artikulationen und Ornamente können mit Noten im Akkordmodus nicht benutzt werden, siehe auch ⟨undefined⟩ [Artikulationenszeichen und Verzierungen], Seite ⟨undefined⟩.
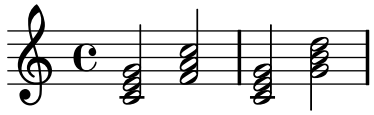
Wenn Akkord- und Notenmodus in linearer Musik abwechseln eingesetzt werden und der Akkordmodus am Anfang steht, erstellt der Notenmodus ein neues Notensystem:

```
\chordmode { c2 f }
<c e g>2 <g' b d>
```



Um dieses Verhalten zu verhindert, muss der `Staff`-Kontext explizit aufgerufen werden:

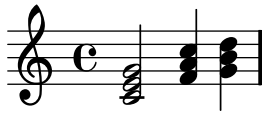
```
\new Staff {
  \chordmode { c2 f }
  <c e g>2  <g' b d>
}
```



Übliche Akkorde
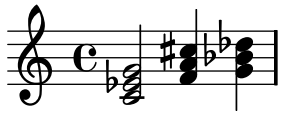
Ein Dreiklang wird mit seinem Grundton mit einer möglichen Dauer dahinter notiert:

```
\chordmode { c2 f4 g }
```



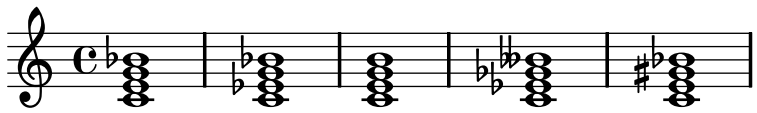
Moll- übermäßige und verminderte Dreiklänge werden notiert, indem `:` und ein Modifikator hinter der Dauer angegeben wird:

```
\chordmode { c2:m f4:aug g:dim }
```

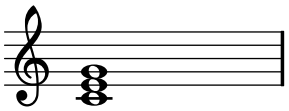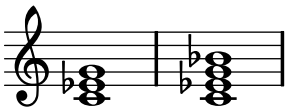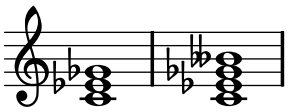
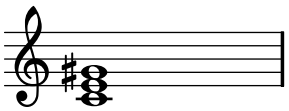
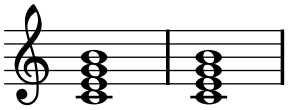


Septakkorde können erstellt werden:

```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



Diese Tabelle zeigt die Funktion der Modifikatoren von Dreiklängen und Septakkorden. Die siebte Stufe wird standardmäßig als kleine Septime realisiert, sodass der Dominantseptakkord die Grundform des Septakkordes darstellt. Alle Alterationen sind relativ zur Dominantsept. Eine vollständigere Tabelle findet sich in Abschnitt A.2 [Übliche Akkord-Variablen], Seite 564.

Modifikator	**Funktion**	**Beispiel**
Kein	Standard: erzeugt einen Durdreiklang.	
m, m7	Mollakkord: Dieser Modifikator erniedrigt die dritte Stufe.	
dim, dim7	Verminderter Akkord: Dieser Modifikator erniedrigt die dritte, fünfte und (wenn vorhanden) die siebte Stufe.	
aug	Übermäßiger Akkord: Dieser Modifikator erhöht die fünfte Stufe.	
maj, maj7	Großer Septakkord: Dieser Modifikator fügt eine erhöhte siebe Stufe hinzu. `7` nach dem `maj` ist optional. NICHT benutzen, um einen Durdreiklang zu notieren.	

Siehe auch

Notationsreferenz: Abschnitt A.2 [Übliche Akkord-Variablen], Seite 564, [Erweiterte und modifizierte Akkorde], Seite 374.

Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Bekannte Probleme und Warnungen

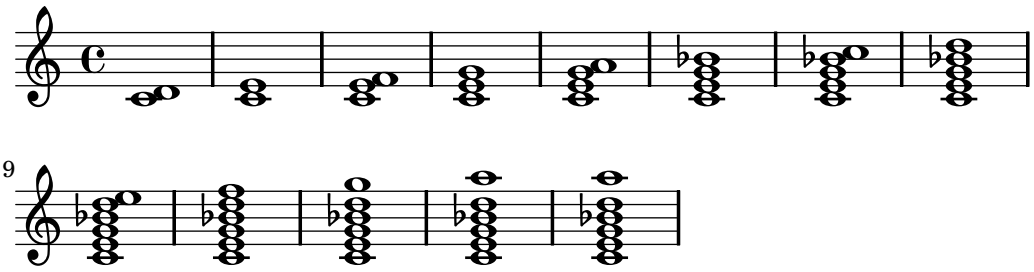
Nur ein Qualitätsmodifikator sollte pro Akkord benutzt werden, meistens für die höchste Stufe des Akkordes. Akkorde mit weiteren Qualitätsmodifikatoren werden ohne Warnung oder Fehlermeldung gelesen, aber das Ergebnis ist nicht vorhersagbar. Akkorde, die nicht mit einem einzigen Qualitätsmodifikator erreicht werden können, sollten mit einzelnen Tonhöhen alteriert werden, wie beschrieben in [Erweiterte und modifizierte Akkorde], Seite 374.

Erweiterte und modifizierte Akkorde

Akkordstrukturen können im Akkordmodus beliebig komplex konstruiert werden. Die Modifikatoren können benutzt werden, um den Akkord zu erweitern, bestimmte Stufen hinzuzufügen oder zu entfernen, Stufen zu erhöhen oder zu erniedrigen und Bassnoten hinzuzufügen bzw. Umkehrungen zu erzeugen.

Die erste Zahl, die auf den Doppelpunkt folgt, wird als „Bereich“ des Akkordes interpretiert: Terzen werden auf dem Grundton gestapelt, bis die angegebene Zahl (=Tonstufe) erreicht ist. Die siebte Stufe, die zu einem Akkord hinzugefügt wird, ist die kleine Septime, nicht die große. Wenn der Bereich keine Terz ist (also etwa 6), dann werden Terzen bis zur höchst möglichen Terz unter dem Bereich gestapelt, und der Endton des Bereichs wird hinzugefügt. Der größtmögliche Wert ist 13. Jeder größere Werte wird als 13 interpretiert.

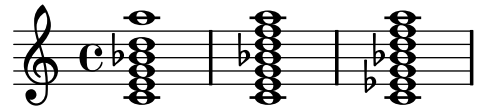
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



Sowohl `c:5` als auch `c` erzeugen einen D-Dur-Dreiklang.
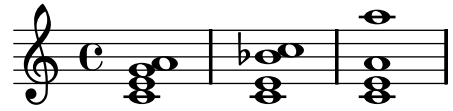
Da eine unveränderte 11 nicht gut klingt, wenn sie mit einer unveränderten 13 zusammenklingt, wird die 11 von einem `:13`-Akkord entfernt (es sei denn sie wird explizit verlangt).

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



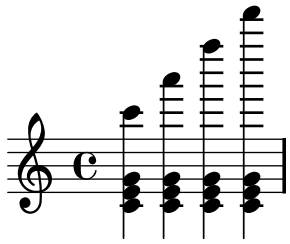
Kompliziertere Akkorde können auch konstruiert werden, indem einzelne Intervalle zu dem Grundton addiert werden. Diese Additionen werden nach dem Bereich notiert und mit Punkten voneinander getrennt. Die normale Septime, die zu einem Akkord hinzugefügt wird, ist die kleine Septime, nicht die große.

```
\chordmode {
  c1:5.6 c:3.7.8 c:3.6.13
}
```
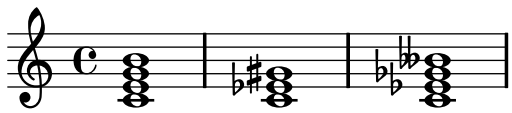


Hinzugefügte Stufen können beliebig groß sein:

```
\chordmode {
  c4:5.15 c:5.20 c:5.25 c:5.30
}
```



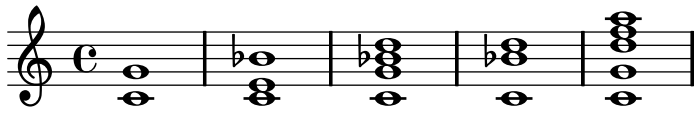
Einzelne Stufen können mit `-` oder `+` vergrößert oder verkleinert werden. Um eine Stufe zu verändert, die automatisch in den Akkord aufgenommen wurde, kann sie in veränderter Form nach dem Bereich hinzugefügt werden.

```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```



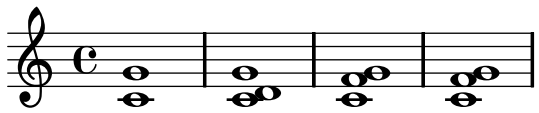
Zu entfernende Töne werden mit der gleichen Methode notiert, allerdings mit einem Dach (`^`) vor der Sequenz, die nicht erscheinen soll. Sie müssen nach den zu addierenden Tönen notiert werden. Die einzelnen zu entfernenden Töne werden mit Punkten getrennt.

```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
}
```



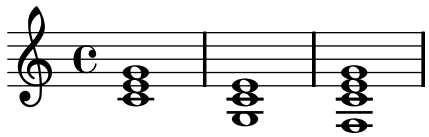
Sekund- und Quartakkorde können mit dem Modifikator `sus` notiert werden. Hiermit wird die dritte Stufe aus dem Akkord entfernt. Mit einer anschließenden `2` wird die zweite, mit einer `4` die vierte Stufe hinzugefügt. `sus` entspricht `^3` und `sus4` ist gleich `.4^3`.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4^3
}
```



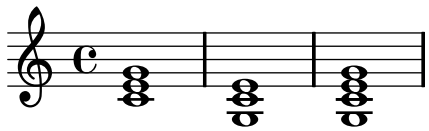
Eine Umkehrung (ein Ton des Akkordes wird unter den Grundton gesetzt) sowie auch zusätzliche Bassnoten können mit dem Schrägstrich (`/`) markiert werden:

```
\chordmode {
  c1 c/g c/f
}
```



Eine Bassnote, die zum Akkord hinzugehört, kann hinzugefügt werden, anstatt dass sie aus dem Akkord entnommen wird, indem noch ein Plus zwischen den Schrägstrich und die Tonhöhe gesetzt wird:

```
\chordmode {
  c1 c/g c/+g
}
```



Akkordmodifikatoren, die benutzt werden können, um eine große Anzahl an Standardakkorden zu erzeugen, werden gezeigt in Abschnitt A.2 [Übliche Akkord-Variablen], Seite 564.

Siehe auch

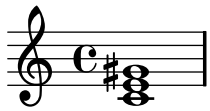
Notationsreferenz: Abschnitt A.2 [Übliche Akkord-Variablen], Seite 564.

Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Bekannte Probleme und Warnungen

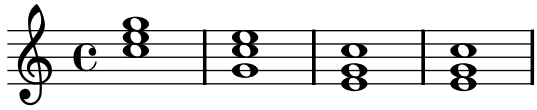
Jede Stufe kann nur einmal in einem Akkord vorkommen. Im folgenden Beispiel wird ein erweiterter Akkord erstellt, weil `5+` zuletzt gelesen wird.

```
\chordmode { c1:5.5-.5+ }
```



Nur die zweite Umkehrung kann erstellt werden, indem eine Bassnote hinzugefügt wird. Die erste Umkehrung erfordert, dass der Grundton des Akkordes geändert wird.

```
\chordmode {
  c'1: c':/g e:6-3-^5 e:m6-^5
}
```



2.7.2 Akkorde anzeigen

Akkorde können zusätzlich zur üblichen Notation als Töne auf einem Notensystem auch mit einem Akkordsymbol gesetzt werden.

Akkordbezeichnungen drucken

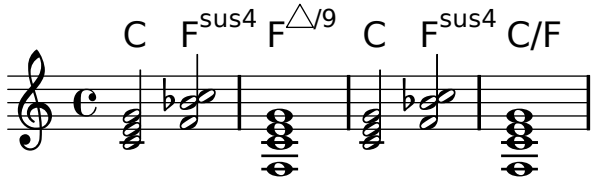
Akkordsymbole anstelle der Noten werde im `ChordNames`-Kontext notiert.

```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```

C F G

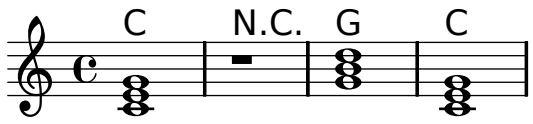
Die Akkorde können entweder als simultane Noten oder unter Einsatz des Akkordmodus (`chordmode`) notiert werden. Der angezeigte Akkord ist der gleiche, es sei denn, Umkehrungen oder zusätzliche Basstöne werden notiert:

```
<<
  \new ChordNames {
    <c e g>2  <f bes c>
    <f c' e g>1
    \chordmode {
      c2 f:sus4 c1:/f
    }
  }
  {
    <c e g>2  <f bes c>
    <f, c' e g>1
    \chordmode {
      c2 f:sus4 c1:/f
    }
  }
>>
```



Pausen, die in einem `ChordNames`-Kontext notiert werden, werden mit der `noChordSymbol`-Beschriftung darstellt.

```
<<
  \new ChordNames \chordmode {
    c1
    r1
    g1
    c1
  }
  \chordmode {
    c1
    r1
    g1
    c1
  }
>>
```



`\chords { ... }` ist eine Kurznotation für die Bezeichnung `\new ChordNames { \chordmode { ... } }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```




```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```

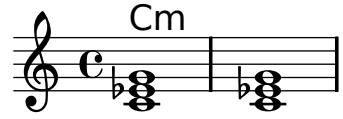


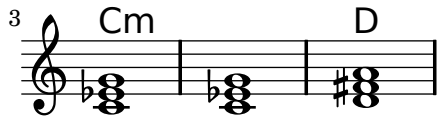
Ausgewählte Schnipsel

Akkordsymbole bei Wechsel anzeigen

Akkordsymbole können so eingestellt werden, dass sie nur zu Beginn der Zeile und bei Akkordwechseln angezeigt werden.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}
<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
  }
  \new Staff {
    \relative c' { \harmonies }
  }
>>
```
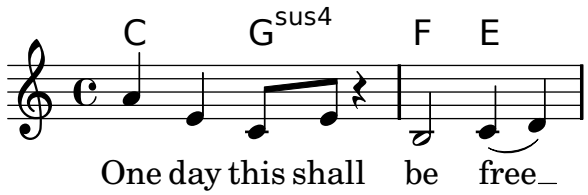




Ein einfaches Liedblatt

Ein Liedblatt besteht aus Akkordbezeichnungen, einer Melodie und dem Liedtext:

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
```

>>



Siehe auch

Glossar: Abschnitt “chord” in *Glossar*.

Notationsreferenz: [Musik parallel notieren], Seite 158.

Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Referenz der Interna: Abschnitt “ChordNames” in *Referenz der Interna*, Abschnitt “ChordName” in *Referenz der Interna*, Abschnitt “Chord_name_engraver” in *Referenz der Interna*, Abschnitt “Volta_engraver” in *Referenz der Interna*, Abschnitt “Bar_engraver” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

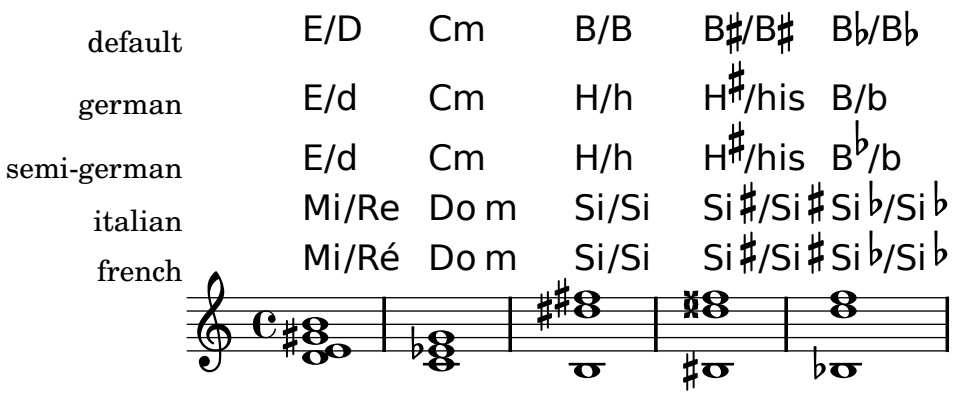
Akkorde, die Umkehrungen oder zusätzliche Basstöne beinhalten, werden nicht richtig bezeichnet, wenn sie im Notenmodus notiert werden.

Akkordbezeichnungen anpassen

Es gibt kein allein gültiges System zur Benennung von Akkorden. Unterschiedliche Musiktraditionen benutzen unterschiedliche Bezeichnungen für die gleichen Akkorde. Es gibt zusätzlich auch unterschiedliche Symbole, die für den gleichen Akkord angezeigt werden können. Die Bezeichnungen und dargestellten Symbole können angepasst werden.

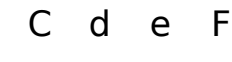
Die Standardeinstellungen für die Symbole entsprechen den Konventionen im Jazz, wie sie von Klaus Ignatzek (siehe Abschnitt “Literatur” in *Aufsatz*). vorgeschlagen wurden. Das Benennungssystem für die Akkorde kann verändert werden, wie weiter unten gezeigt wird. Ein alternatives Notationssystem für Jazzakkorde ist auch erhältlich. Die Ignatzek und die alternative Jazznotation finden sich in der Tabelle in Abschnitt A.1 [Liste der Akkordbezeichnungen], Seite 563.

Zusätzlich zu den unterschiedlichen Bezeichnungssystemen werden unterschiedliche Notenbezeichnungen für die Grundtöne. Die vordefinierten Befehle `\germanChords`, `\semiGermanChords`, `\italianChords` und `\frenchChords` setzen diese Variablen. Die Auswirkungen werden im nächsten Beispiel gezeigt.



Deutsche Liederbücher zeigen Mollakkorde oft durch die Verwendung von Kleinbuchstaben an, ohne die Endung *m*. Dieses Verhalten kann erreicht werden, indem man die `chordNameLowercaseMinor`-Eigenschaft setzt:

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```



Wenn keine der definierten Einstellungen zum gewünschten Ergebnis führt, kann die Anzeige des Akkordsymbols durch die folgenden Eigenschaften verändert werden:

`chordRootNamer`
: Das Akkordsymbol wird normalerweise als Buchstabe des Grundtons mit optionaler Alteration dargestellt. Die Interpretation einer Tonhöhe als Buchstabe wird von der `chordRootNamer`-Funktion übernommen. Besondere Bezeichnungen, wie etwa im Deutschen H für einen H-Dur-Akkord (und nicht „B“ wie im Englischen), können durch Hinzufügen einer neuen Funktion zu dieser Eigenschaft erstellt werden.

`majorSevenSymbol`
: Mit dieser Eigenschaft wird das Aussehen der Notation für die große Septime (7) bestimmt. Vordefiniert sind die Optionen `whiteTriangleMarkup` und `blackTriangleMarkup`.

`chordNoteNamer`
: Wenn das Akkordsymbol zusätzliche Tonhöhen enthält, die nicht den Grundton darstellen (etwa eine zusätzliche Bassnote), wird diese Funktion eingesetzt, um die zusätzliche Tonhöhe auszugeben. In den Standardeinstellungen wird die Tonhöhe mit der `chordRootNamer`-Funktion gesetzt. Die `chordNoteNamer`-Eigenschaft hingegen kann dieses Verhalten verändern und etwa den Basston etwa als Kleinbuchstaben darstellen.

`chordNameSeparator`
: Verschiedene Teile eines Akkordsymboles werden normalerweise durch einen Schrägstrich markiert. Indem `chordNameSeparator` ein anderer Wert zugewiesen wird, kann ein beliebiges Zeichen für den Trenner benutzt werden.

`chordNameExceptions`
: Diese Funktion ist eine Liste mit Paaren. Das erste Objekt eines Paares ist eine Anzahl von Tonhöhen, die die Stufen eines Akkordes definieren. Das zweite Objekt ist eine Beschriftung, die nach `chordRootNamer` ausgegeben wird, um das Akkordsymbol zu erstellen.

`chordPrefixSpacer`
: Das „m“ für Moll-Akkorde wird normalerweise direkt hinter dem Akkordbuchstaben gesetzt. Mit der Eigenschaft `chordPrefixSpacer` kann ein Abstand(halter) zwischen den Buchstaben und das „m“ gesetzt werden. Der Abstandhalter wird nicht verwendet, wenn der Grundton erhöht oder erniedrigt ist.

Vordefinierte Befehle

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

Ausgewählte Schnipsel

Akkordsymbolausnahmen

Die Eigenschaft `chordNameExceptions` kann benutzt werden, um eine Liste an besonderen Notationen für bestimmte Akkorde zu speichern.

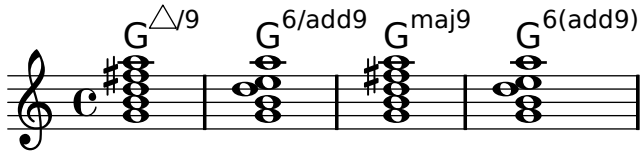
```
% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

% Convert music to list and prepend to existing exceptions.
chExceptions = #( append
  ( sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

<< \context ChordNames \theMusic
   \context Voice \theMusic
>>
```



Akkordbezeichnung maj7

Das Aussehen des großen Septakkords kann mit `majorSevenSymbol` verändert werden.

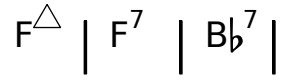
```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```



Tatkstriche in einen ChordNames-Kontext hinzufügen

Um Taktstriche in einem `ChordNames`-Kontext anzeigen zu lassen, muss der `Bar_engraver` hinzugefügt werden.

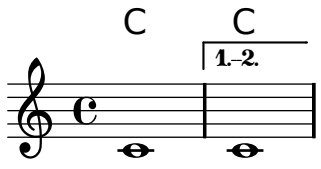
```
\new ChordNames \with {
  \override BarLine #'bar-extent = #'(-2 . 2)
  \consists "Bar_engraver"
}
\chordmode {
  f1:maj7 f:7 bes:7
}
```



Wiederholungs-(Volta-)Klammern unterhalb der Akkordsymbole

Indem man den `Volta_engraver` zu dem entsprechenden Notensystem hinzufügt, können Wiederholungsklammern unterhalb der Akkorde gesetzt werden.

```
\score {
  <<
    \chords {
      c1
      c1
    }
    \new Staff \with {
      \consists "Volta_engraver"
    }
    {
      \repeat volta 2 { c'1 }
      \alternative { c' }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```

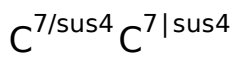


Akkordsymboltrenner verändern

Der Trenner zwischen unterschiedlichen Teilen eines Akkordsymbols kann beliebiger Text sein.

```
\chords {
  c:7sus4
  \set chordNameSeparator
    = \markup { \typewriter | }
  c:7sus4
```

```
}
```



Siehe auch

Notationsreferenz: Abschnitt A.1 [Liste der Akkordbezeichnungen], Seite 563, Abschnitt A.2 [Übliche Akkord-Variablen], Seite 564.

Aufsatz über den automatischen Musiksatz: Abschnitt “Literatur” in *Aufsatz*.

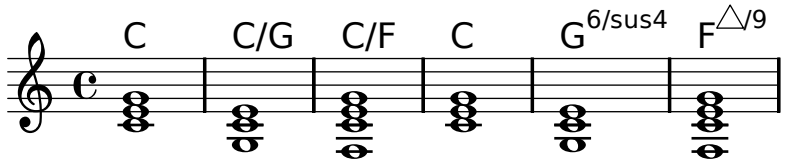
Installierte Dateien: ‘`scm/chords-ignatzek.scm`’, ‘`scm/chord-entry.scm`’, ‘`ly/chord-modifier-init.ly`’.

Schnipsel: Abschnitt “Chords” in *Schnipsel*.
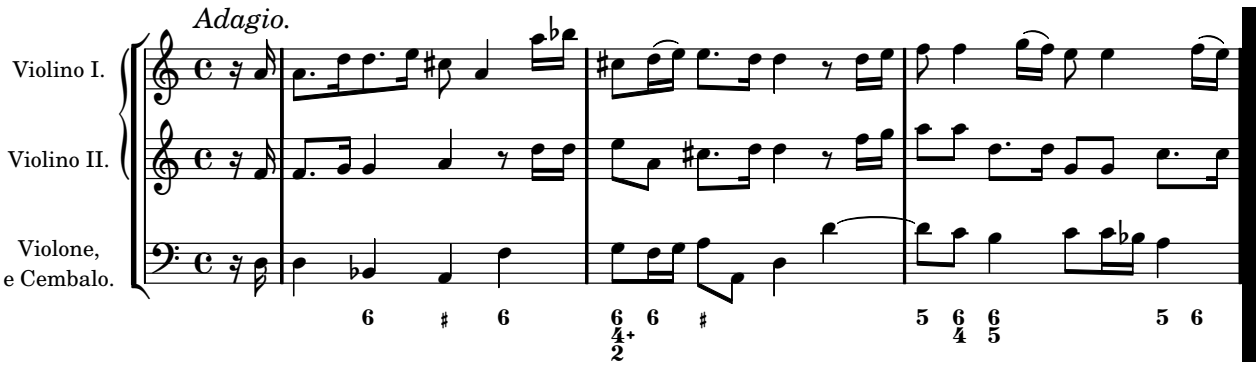
Bekannte Probleme und Warnungen
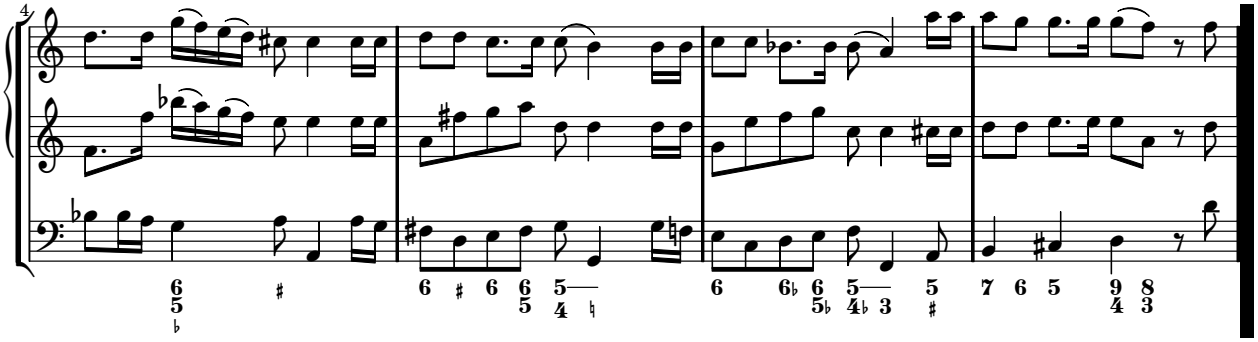
Akkordsymbole werden von den Tonhöhenbezeichnungen innerhalb des Akkordes und der Information über die Akkordstruktur, die innerhalb von `\chordmode` notiert wurde, bestimmt. Wenn der direkte Notenmodus benutzt wird, stammen unerwünschte Bezeichnungen daher, dass Umkehrungen und zusätzliche Bassnoten nicht richtig interpretiert werden.

```
myChords = \relative c' {
  \chordmode { c1 c/g c/f }
  <c e g>1 <g c e> <f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
>>
```



2.7.3 Generalbass
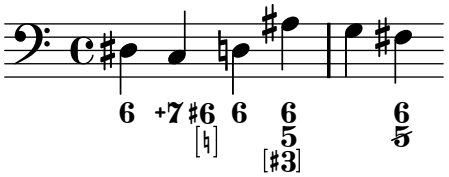




Generalbassnotation kann dargestellt werden.

Grundlagen des Bezifferten Basses

LilyPond stellt Unterstützung für Generalbassnotation, auch als Basso Continuo bezeichnet, zur Verfügung.

```
<<
  \new Voice { \clef bass dis4 c d ais g fis}
  \new FiguredBass {
    \figuremode {
      < 6 >4 < 7\+ >8 < 6+ [_!] >
      < 6 >4 <6 5 [3+] >
      < _ >4 < 6 5/>4
    }
  }
>>
```



Die Unterstützung für Generalbass besteht aus zwei Teilen: Es gibt einen Eingabe-Modus, aktiviert durch den Befehl `\figuremode`, in dem Ziffern für den Bass als Nummern eingegeben werden können, und einen Kontext `FiguredBass`, der dafür sorgt, dass die entsprechenden `BassFigure`-Objekte auch erstellt werden. Generalbass kann auch in einem `Staff`-Kontext dargestellt werden.

`\figures{ ... }` ist eine Kurznotation für `\new FiguredBass { \figuremode { ... } }`.

Auch wenn die Unterstützung für Generalbass auf den ersten Blick wie die Akkordunterstützung ausschauen mag, ist sie sehr viel einfacher. `\figuremode` speichert einfach die Zahlen und der `FiguredBass`-Kontext setzt sie in der Form, wie sie notiert wurden. Sie werden nicht in Tonhöhen umgewandelt.

Siehe auch

Glossar: Abschnitt “figured bass” in *Glossar*.

Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Eingabe des Generalbass’

`\figuremode` (Zahlenmodus) wird benutzt, um den Eingabemodus auf den Zahlenmodus umzustellen. Mehr Information zu unterschiedlichen Eingabemodi findet sich in Abschnitt 5.4.1 [Eingabe-Modi], Seite 536.

Im Zahlenmodus wird eine Gruppe von Bassziffern mit den Zeichen `<` und `>` begrenzt. Die Dauer wird nach dem `>`-Zeichen eingegeben.

```
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
```



Versetzungszeichen (inklusive Auflösungszeichen) können hinzugefügt werden:

```
\figures {
  <7! 6+ 4-> <5++> <3-->
}
```



Übermäßige und verminderte Stufen können dargestellt werden:

```
\figures {
  <6\+ 5/> <7/>
}
```



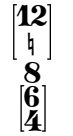
Ein Schrägstrich von links nach rechts (üblicherweise für erhöhte Sexten benutzt) kann erstellt werden:

```
\figures {
  <6> <6\\>
}
```



Vertikaler Platz und Klammern können zu den Zahlen hinzugefügt werden:

```
\figures {
  <[12 _!] 8 [6  4]>
}
```



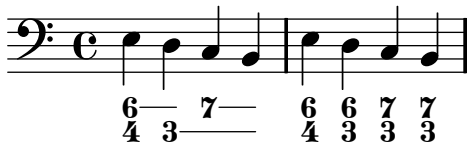
Beliebiger Text kann als Zahl notiert werden:

```
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
```



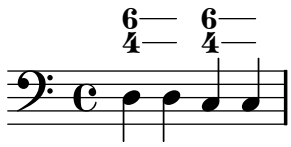
Es ist auch möglich, Fortsetzungslinien für wiederholte Ziffern zu benutzen.

```
<<
  {
    \clef bass
    e4 d c b,
    e4 d c b,
  }
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 3> <7 3> <7 3>
    \bassFigureExtendersOff
    <6 4>4 <6 3> <7 3> <7 3>
  }
>>
```

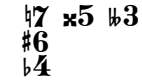


In diesem Fall werden wiederholte Ziffern immer durch eine Linie ersetzt, es sei denn, die Linie wird explizit beendet.

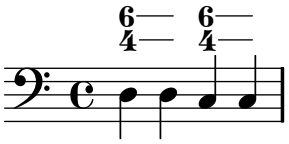
```
<<
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 4> <6\! 4\!> <6 4>
  }
  {
    \clef bass
    d4 d c c
  }
>>
```



Die folgende Tabelle zeigt die vorhandenen Zahlenmodifikatoren:

Modifier	**Purpose**	**Example**
+, -, !	Accidentals	
\+, /	Augmented and diminished steps	
\\	Raised sixth step	

\! End of continuation line



Vordefinierte Befehle

`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

Ausgewählte Schnipsel

Positionen von Generalbass-Alterationszeichen verändern

Versetzungszeichen und Pluszeichen können vor oder nach den Ziffern erscheinen, je nach den Einstellungen der `figuredBassAlterationDirection` und `figuredBassPlusDirection`-Eigenschaften.

```
\figures {
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5+> <6 4-> r
}
```



Siehe auch

Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Referenz der Interna: Abschnitt “BassFigure” in *Referenz der Interna*, Abschnitt “BassFigureAlignment” in *Referenz der Interna*, Abschnitt “BassFigureLine” in *Referenz der Interna*, Abschnitt “BassFigureBracket” in *Referenz der Interna*, Abschnitt “BassFigureContinuation” in *Referenz der Interna*, Abschnitt “FiguredBass” in *Referenz der Interna*.

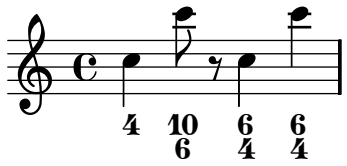
Generalbass anzeigen

Generalbass kann mit dem `FiguredBass`-Kontext, aber auch in den meisten anderen `Staff`-Kontexten dargestellt werden.

Wenn die Ziffern im `FiguredBass`-Kontext dargestellt werden, ist die vertikale Position der Ziffern unabhängig von den Noten des parallelen Systems.

```
<<
  \relative c'' {
    c4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
```
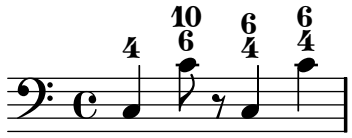
```
  }
>>
```



In diesem Beispiel muss der `FiguredBass`-Kontext explizit erstellt werden, damit kein zusätzliches (leeres) Notensystem erstellt wird.

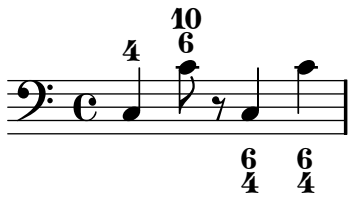
Bassziffern können auch direkt einem Notensystemkontext (`Staff`) hinzugefügt werden. In diesem Fall wird ihre vertikale Position automatisch bestimmt.

```
<<
  \new Staff = myStaff
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = myStaff
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



Wenn Generalbass zu einem vorhandenen System hinzugefügt wird, ist es möglich, die Ziffern über oder unter dem System anzuzeigen:

```
<<
  \new Staff = myStaff
  \figuremode {
    <4>4 <10 6>8 s8
    \bassFigureStaffAlignmentDown
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = myStaff
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```
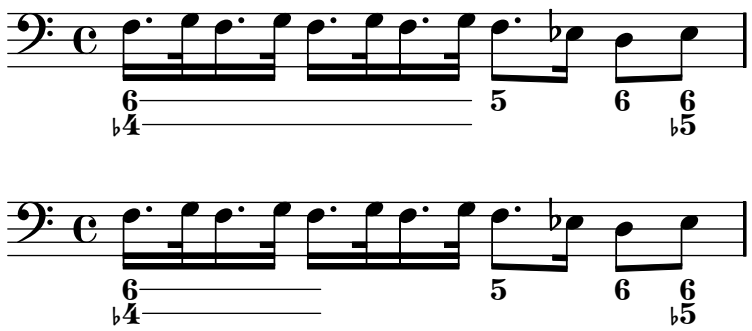


Schnipsel: Abschnitt “Chords” in *Schnipsel*.

Referenz der Interna: Abschnitt “BassFigure” in *Referenz der Interna*, Abschnitt “BassFigureAlignment” in *Referenz der Interna*, Abschnitt “BassFigureLine” in *Referenz der Interna*, Abschnitt “BassFigureBracket” in *Referenz der Interna*, Abschnitt “BassFigureContinuation” in *Referenz der Interna*, Abschnitt “FiguredBass” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

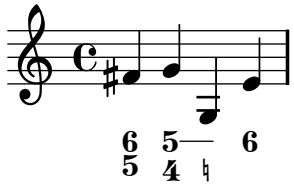
Um sicherzugehen, dass die Fortsetzungslinien funktionieren, sollte der gleiche Rhythmus für die Bassfiguren und die eigentlichen Noten der Bassstimme benutzt werden.

```
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are correct here, with the same rhythm as the bass
    \repeat unfold 4 { <6 4->16. <6 4->32 }
    <5>8. r16 <6>8 <6\! 5->
  }
>>
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are incorrect here, even though the timing is the same
    <6 4->4 <6 4->4
    <5>8. r16 <6>8 <6\! 5->
  }
>>
```



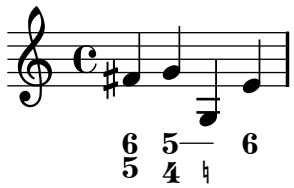
Wenn Fortsetzungslinien eingesetzt werden, können aufeinander folgende Bezifferungen mit der selben Zahl in einer anderen Position dazu führen, dass sich die Reihenfolge der Zahlen umkehrt.

```
<<
  { fis4 g g, e' }
  \figures {
    \bassFigureExtendersOn
    <6 5>4 <5\! 4> < 5 _!> <6>
  }
>>
```



Um dieses Problem zu umgehen, kann die Fortsetzungslinie nach der Bezifferung, mit der die Linie beginnen soll, angeschaltet und am Ende der Linie wieder ausgeschaltet werden.

```
<<
  { fis4 g g, e' }
  \figures {
    <6 5>4 <5 4>
    \bassFigureExtendersOn
    < 5 _!>4 <6>
    \bassFigureExtendersOff
  }
>>
```



2.8 Zeitgenössische Musik

Seit Anfang des 20. Jahrhunderts wurden die kompositorischen Stile und Kompositionstechniken sehr start erweitert. Neue harmonische und rhythmische Entwicklungen, eine Erweiterung der verwendeten Tonhöhen und die Entwicklung eines großen Spektrums neuer instrumentaler Techniken wurden von einer parallelen Evolution der Notationstechnik begleitet. Die Absicht dieses Abschnittes ist es, Informationen und Hintergrundwissen zu bieten, der zur Notation zeitgenössischer Musik benötigt wird.

2.8.1 Tonhöhe und Harmonie in zeitgenössischer Musik

Dieser Abschnitt zeigt Lösungen zur Notation von zeitgenössischen Tonhöhen und Harmonien.

Verweise zu Tonhöhe und Harmonie in zeitgenössischer Musik

- Normale Vierteltonmusik wird behandelt in [Notenbezeichnungen in anderen Sprachen], Seite 7.
- Nicht-Standardvorzeichen werden behandelt in [Tonartbezeichnung], Seite 20.
- Contemporary practises in displaying accidentals are addressed in [Automatische Versetzungszeichen], Seite 24.

Mikrotonale Notation

Zeitgenössische Tonartvorzeichnung und Harmonie

2.8.2 Zeitgenössische Notation von Rhythmen

Dieser Abschnitt erklärt Besonderheiten, die wichtig für die Notation von Rhythmus in zeitgenössischer Musik sind.

Verweise für zeitgenössische Benutzung von Rhythmus

- Zusammengesetzte Taktarten werden erklärt in #[Taktangabe], Seite 58.
- Grundlegende polymetrische Notation ist erklärt in [Polymetrische Notation], Seite 68.
- Gespreizte Balken sind erklärt in [Gespreizte Balken], Seite 86.
- Mensurstrich-Taktstriche (zwischen den Systemen) finden sich erklärt in ⟨undefined⟩ [Grouping staves], Seite ⟨undefined⟩.

N-tolen in zeitgenössischer Musik

Zeitgenössische Taktarten

Erweiterte polymetrische Notation

Balken in zeitgenössischer Musik

Taktstriche in zeitgenössischer Musik

2.8.3 Graphische Notation

2.8.4 Zeitgenössische Partiturtechniken

2.8.5 Neue Instrumententechniken

2.8.6 Leseliste und interessante Referenzpartituren
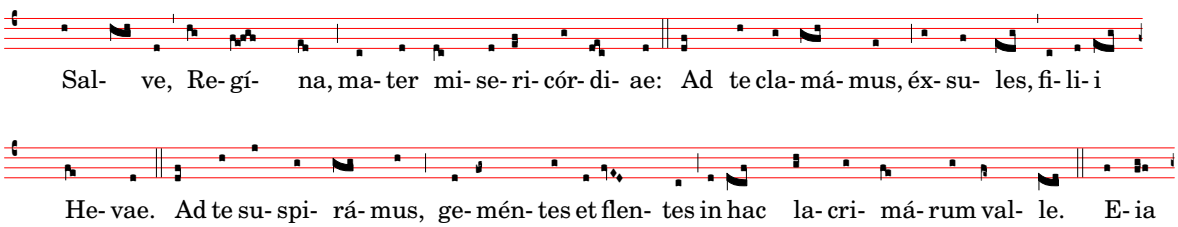
Dieser Abschnitt weist auf einige Bücher, Musikbeispiele und andere Resourcen hin, die relevant für die Notation zeitgenössischer Musik sind.

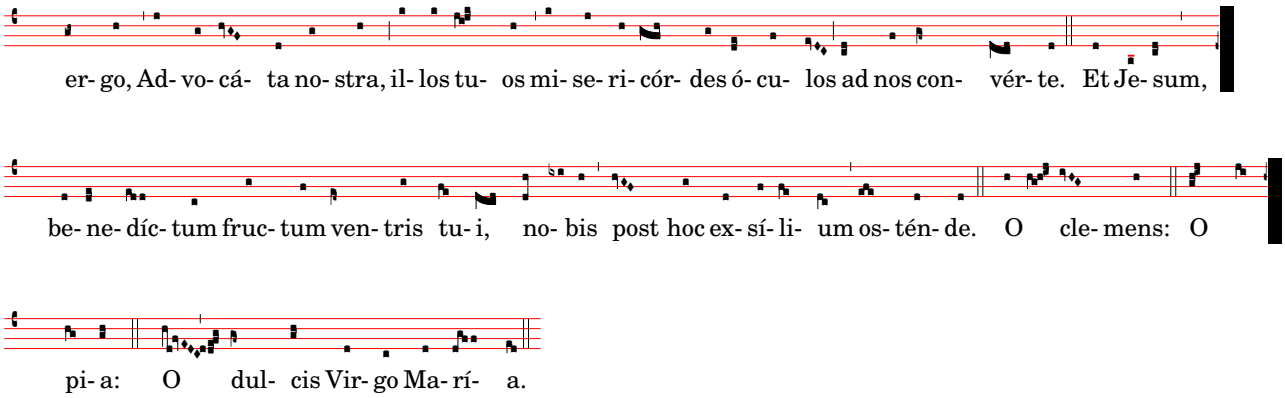
Bücher und Artikel über zeitgenössische Notation

- *Music Notation in the Twentieth Century: A Practical Guidebook* von Kurt Stone [W. W. Norton, 1980]
- *Music Notation: A Manual of Modern Practice* von Gardner Read [Taplinger, 1979]
- *Instrumentation and Orchestration* von Alfred Blatter [Schirmer, 2nd ed. 1997]

Partituren und Musikbeispiele

2.9 Notation von alter Musik





Unterstützung für Notation der Alten Musik enthält einige Eigenheiten der Mensuralnotation und der Notation des gregorianischen Chorals. Diese Eigenheiten können eingestellt werden, indem man Stileigenschaften von graphischen Objekten wie Notenköpfen und Pausen verändert, oder indem man vordefinierte fertige Kontexte für mensurale oder Choralnotation einsetzt.

Viele graphische Objekte, wie Notenköpfe, Fähnchen, Versetzungszeichen, Taktarten und Pausen haben eine `style`-Eigenschaft, die verändert werden kann, um verschiedene Stile Alter Notation nachzuahmen. Siehe auch:

- [Mensurale Notenköpfe], Seite 398,
- [Mensurale Versetzungszeichen und Tonartbezeichnung], Seite 400,
- [Mensurale Pausen], Seite 400,
- [Mensurale Schlüssel], Seite 396,
- [Gregorianische Schlüssel], Seite 404,
- [Mensurale Fähnchen], Seite 399,
- [Mensurale Taktartenbezeichnungen], Seite 397.

Ein paar notationelle Konzepte sind inbesondere für die Notation Alter Musik eingeführt worden:

- [Custodes], Seite 394,
- [Divisiones], Seite 405,
- [Ligaturen], Seite 394.

Siehe auch

Glossar: Abschnitt “custos” in *Glossar*, Abschnitt “ligature” in *Glossar*, Abschnitt “mensural notation” in *Glossar*.

Notationsreferenz: [Mensurale Notenköpfe], Seite 398, [Mensurale Versetzungszeichen und Tonartbezeichnung], Seite 400, [Mensurale Pausen], Seite 400, [Mensurale Schlüssel], Seite 396, [Mensurale Fähnchen], Seite 399, [Mensurale Taktartenbezeichnungen], Seite 397, [Gregorianische Schlüssel], Seite 404, [Custodes], Seite 394, [Divisiones], Seite 405, [Ligaturen], Seite 394.

2.9.1 Überblick über die unterstützten Stile

Drei Stile sind vorhanden, um den gregorianischen Choral zu setzen:

- *Editio Vaticana* ist ein vollständiger Stil für den gregorianischen Choral, der stilistisch den Choralausgaben von Solsemes folgt. Hierbei handelt es sich um die offizielle Choralausgabe des Vatikans seit 1904. LilyPond unterstützt alle Notationszeichen, die in diesem Stil benutzt werden, inklusive Ligaturen, custodes und besondere Zeichen wie die Quilisma und den Oriscus.

- Der *Editio Medicaea*-Stil stellt bestimmte Eigenschaften zur Verfügung, die in den Medicaea (oder Ratisbona)-Editionen benutzt wurden. Dieser Stil war vor den Solesmes-Editionen in Benutzung. Der größte Unterschied von dem *Vaticana*-Stil sind die Schlüssel, die nach unten gerichtete Striche haben, und die Notenköpfe, die hier quadratisch und ebenmäßig geformt sind.
- Der *Hufnagel-* oder *gotische* Stil ahmt den Stil der Schreiber bestimmter Manuskripte aus dem Deutschland und Mitteleuropa des Mittelalters nach. Er ist nach der Form der wichtigsten Note (der *Virga*) benannt, die wie ein kleiner Nagel aussieht.

Drei Stile ahmen die Erscheinung von Renaissancehandschriften und -drucken der Mensuralmusik nach:

- Der *Mensural*-Stil versucht, den Stil von Handschriften nachzuahmen und hat recht kleine, rhombenförmige Notenköpfe und wie handgeschriebene Pausenzeichen.
- Der *Neomensural*-Stil ist eine modernisierte und stilisierte Version des erstens: Die Notenköpfe sind etwas breiter und die Pausen bestehen aus graden Linien. Dieser Stil ist besonders gut geeignet, um moderne Editionen der Mensuralmusik mit einem Incipit zu versehen.
- Der *Petrucci*-Stil ist nach Ottaviano Petrucci (1466-1539) benannt, dem ersten Drucker, der bewegliche Stempel benutzt hat, um musikalische Notation zu drucken (in seinem Buch *Harmonice musices odhecaton*, 1501). Dieser Stil setzt größere Notenköpfe ein als die anderen mensuralen Stile.

Baroque (Barockstil) und *Classical* (klassischer Stil) sind keine vollständigen Stile, sondern unterscheiden sich vom Standard nur in einigen Details: der Barockstil verändert bestimmte Notenköpfe, der klassische Stil die Form der Viertelpause.

Nur der Mensuralstil hat für alle Aspekte der Notation eine alternative Form. Die anderen Stile sind nur teilweise ausgeführt: die gregorianischen Stile haben keine Pausen oder Fähnchen, weil diese Zeichen im Choral nicht vorkommen, und der Petrucci-Stil hat keine eigenen Fähnchen und Versetzungszeichen.

Jedes Notationselement kann unabhängig von den anderen verändert werden, sodass man gut mensurale Fähnchen, Petrucci-Notenköpfe, klassische Pausen und Vatikana-Schlüssel nebeneinander benutzen kann, wenn das gewünscht ist.

Siehe auch

Glossary: Abschnitt “mensural notation” in *Glossar*, Abschnitt “flag” in *Glossar*.

2.9.2 Alte Notation – Allgemeines

Vordefinierte Umgebungen

Für den gregorianischen Choral und die Mensuralnotation gibt es vordefinierte Stimm- und Systemkontexte, die all die Notationszeichen auf Werte setzen, die diesem Stil angemessen sind. Wenn man mit den Werten zufrieden ist, kann man sofort mit der Notation beginnen, ohne sich um die Einzelheiten von tiefergreifenden Kontextanpassungen kümmern zu müssen. Die definierten Kontexte sind: `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice` und `MensuralStaff`.

Siehe auch

- [Gregorianische Gesangs-Kontexte], Seite 403,
- [Mensural-Kontexte], Seite 395.

Siehe auch

Glossar: Abschnitt “mensural notation” in *Glossar*.
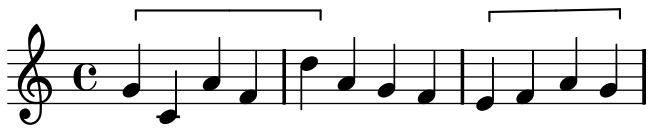
Notationsreferenz: [Gregorianische Gesangs-Kontexte], Seite 403, [Mensural-Kontexte], Seite 395.

Ligaturen

Eine Ligatur ist ein graphisches Symbol das wenigstens zwei unterschiedliche Noten darstellt. Ligaturen treten ursprünglich in Manuskripten des Gregorianischen Chorals auf, um auf- oder absteigende Notensequenzen zu notieren.

Ligaturen werden in LilyPond notiert, indem die dazugehörigen Noten zwischen `\[` und `\]` eingeschlossen werden. Einige Ligaturstile benötigen zusätzliche Syntax für eine bestimmte Ligatur. In der Standardeinstellung setzt der `LigatureBracket`-Engraver ganz einfach eckige Klammern über die Noten der Ligatur.

```
\transpose c c' {
  \[ g c a f d' \]
  a g f
  \[ e f a g \]
}
```



Es gibt zwei weitere Ligaturstile: Vaticana für den gregorianischen Choral und mensural für Mensuralnotation (wobei hier nur weiße Ligaturen unterstützt sind, und auch sie nur beschränkt). Um einen gestimmten Ligaturstil auszuwählen, muss der `Ligature_bracket_engraver` mit einem entsprechenden Ligatur-Engraver im Stimmenkontext ausgetauscht werden, wie erklärt in [Weiße Mensuralligaturen], Seite 402 und [Ligaturen der gregorianischen Quadratnotation], Seite 407.

Siehe auch

Glossar: Abschnitt “ligature” in *Glossar*.

Notationsreferenz: [Weiße Mensuralligaturen], Seite 402, [Ligaturen der gregorianischen Quadratnotation], Seite 407.

Bekannte Probleme und Warnungen

Ligaturen benötigen eine Platzaufteilung, die sich von der klassischen Notation deutlich unterscheidet. Das ist bisher sehr schlecht verwirklicht, sodass fast immer zu viel Platz zwischen Ligaturen ist und Zeilenumbrüche unbefriedigend ausfallen. Text lässt sich auch nicht richtig an Ligaturen ausrichten.

Versetzungszeichen dürfen nicht innerhalb von einer Ligatur gedruckt werden, sondern müssen gesammelt und vor der Ligatur ausgegeben werden.

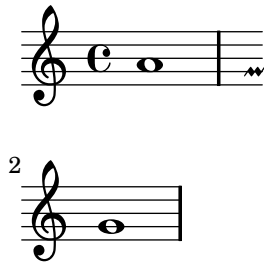
Die Syntax verwendet immer noch den verworfenen Infix-Stil (`\[ musik. Ausdr. \]`). Aus Gründen der Konsistenz soll dies geändert werden in den Postfix-Stil (`Note\[ ... Note\]`).

Custodes

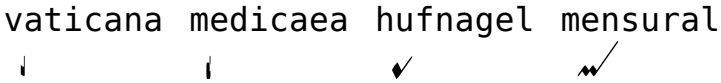
Ein *Custos* (Plural: *Custodes*; Lateinisch: „Weiser“) ist ein Symbol, das am Ende jedes Notensystems erscheint. Es nimmt die Tonhöhe der ersten Note der nächsten Zeile vorweg und hilft damit dem Vortragenden, die Zeilenwechsel während der Vorführung zu bewältigen.

Custodes wurden bis zum 17. Jahrhundert sehr häufig in der Musiknotation eingesetzt. Heute finden sie sich nur noch in einigen bestimmten Notationsformen, etwa modernen Editionen des Gregorianischen Chorals wie der *editio vaticana*. LilyPond stellt unterschiedliche Custos-Symbole für die unterschiedlichen Notationsstile zur Verfügung.

Damit Custodes angezeigt werden, muss ein `Custos_engraver` im `Staff`-Kontext gefordert werden. Der Aufruf folgt im Rahmen des Layout-Kontextes, wie das folgende Beispiel zeigt. Der Stil des Custos wird mit dem `override`-Befehl eingestellt, wie in dem folgenden Beispiel gezeigt:



Das Custos-Zeichen wird mit der `style`-Eigenschaft ausgewählt. Die unterstützten Stile sind: `vaticana`, `medicaea`, `hufnagel` und `mensural`. Sie werden im folgenden Fragment demonstriert.



Siehe auch

Music Glossary: Abschnitt “custos” in *Glossar*.

Referenz der Interna: Abschnitt “Custos” in *Referenz der Interna*.

Schnipsel: Abschnitt “Ancient notation” in *Schnipsel*.

Unterstützung für Generalbass

Es gibt beschränkte Unterstützung für Generalbassziffern aus der Barockzeit. Siehe hierzu Abschnitt 2.7.3 [Generalbass], Seite 383.

Siehe auch

Glossar: Abschnitt “figured bass” in *Glossar*.

Notationsreferenz: Abschnitt 2.7.3 [Generalbass], Seite 383.

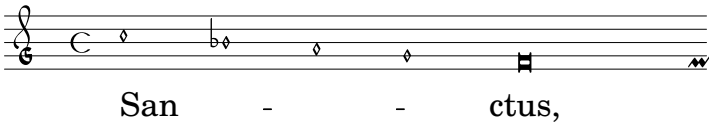
2.9.3 Mesurale Musik setzen

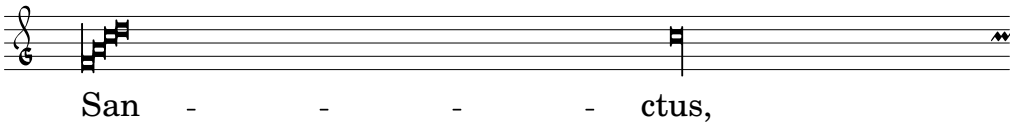
Mensural-Kontexte

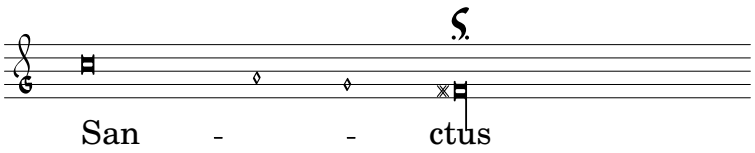
Die vordefinierten Kontexte `MensuralVoice` und `MensuralStaff` können eingesetzt werden, um ein Stück in Mensuralnotation zu schreiben. Die Kontexte initialisieren alle relevanten Eigenschaften und graphischen Objekte, so dass unmittelbar mit der Notation begonnen werden kann. Siehe das folgende Beispiel:

```
\score {
  <<
    \new MensuralVoice = "discantus" \transpose c c' {
      \override Score.BarNumber #'transparent = ##t {
        c'1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c'\breve d'\melismaEnd \]
```

```
        c'\longa
        c'\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```
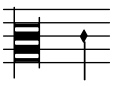






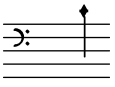
Siehe auch

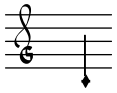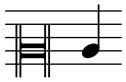
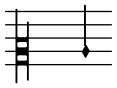
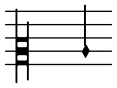
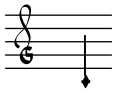
Glossar: Abschnitt “mensural notation” in *Glossar*.

Mensurale Schlüssel

In der Tabelle unten werden alle Mensuralschlüssel gezeigt, die mit dem `\clef`-Befehl erreicht werden. Manche Schlüssel benutzen dasselbe Zeichen, unterscheiden sich aber in der Notenlinie, auf der der Schlüssel notiert wird. In diesem Fällen ist eine Nummer im Schlüsselnamen eingefügt, nummeriert von unten nach oben. Man kann aber trotzdem eine beliebige Nummer erzwingen, wie es im Abschnitt [Notenschlüssel], Seite 16 beschrieben wird. Die Note, die rechts von jedem Schlüssel gesetzt ist, zeigt das `c'` in Bezug zu dem jeweiligen Schlüssel.

Petrucci hat C-Schlüssel benutzt, die unterschiedlich ausbalancierte vertikale Balken auf der linken Seite hatten, je nachdem, auf welcher Notenlinie er sich befand.

Beschreibung	**Unterstützte Schlüssel**	**Beispiel**
Mensuraler C-Schlüssel im historischen Stil	`mensural-c1`, `mensural-c2`, `mensural-c3`, `mensural-c4`	
Mensuraler F-Schlüssel im historischen Stil	`mensural-f`	

Mensuraler G-Schlüssel im historischen Stil	`mensural-g`	
Mensuraler C-Schlüssel im modernen Stil	`neomensural-c1`, `neomensural-c2`, `neomensural-c3`, `neomensural-c4`	
Mensuraler C-Schlüssel im Petrucci-Stil, zur Benutzung auf verschiedenen Notenlinien (im Beispiel den Schlüssel auf der zweiten Linie)	`petrucci-c1`, `petrucci-c2`, `petrucci-c3`, `petrucci-c4`, `petrucci-c5`	
Mensuraler F-Schlüssel im Petrucci-Stil, kann auf verschiedenen Notenlinien benutzt werden (im Beispiel auf der dritten Linie)	`petrucci-c1`, `petrucci-c2`, `petrucci-c3`, `petrucci-c4`, `petrucci-c5`	
Mensuraler G-Schlüssel im Petrucci-Stil	`petrucci-g`	

Siehe auch

Glossar: Abschnitt “mensural notation” in *Glossar*, Abschnitt “clef” in *Glossar*.

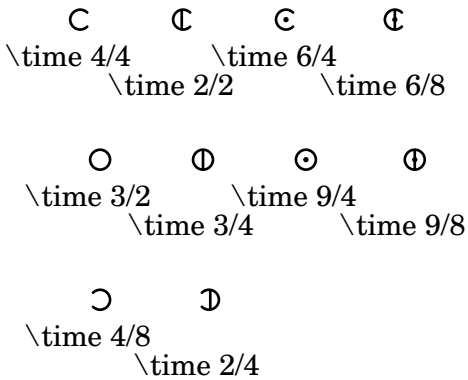
Notationsreferenz: [Notenschlüssel], Seite 16.

Bekannte Probleme und Warnungen

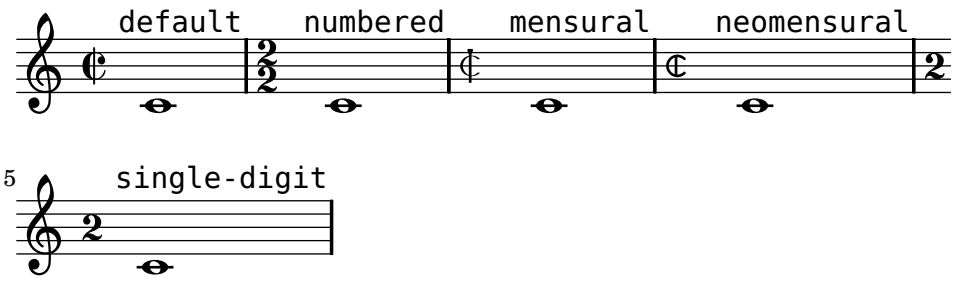
Der mensurale G-Schlüssel ist als Petrucci-G-Schlüssel deklariert.

Mensurale Taktartenbezeichnungen

LilyPond besitzt beschränkte Unterstützung für Mensurzeichen (die den heutigen Taktarten ähneln, aber doch einige Eigenheiten haben). Die Symbole sind starr verknüpft mit bestimmten Brüchen. Darum müssen die Werte `n` und `m` der folgenden Tabelle in den Befehl `\time n/m` eingesetzt werden, um die entsprechenden Symbole zu erhalten.



Mit der `style`-Eigenschaft des Objektes `TimeSignature` können die Taktarten angewählt werden. Unterstützte Stile sind: `neomensural` und `mensural`. In der Tabelle oben wurde der neomensurale Stil verwendet. Im folgenden Beispiel sind die unterschiedlichen Stile dargestellt.



Siehe auch

Glossary: Abschnitt “mensural notation” in *Glossar*.

Notationsreferenz: [Taktangabe], Seite 58.

Bekannte Probleme und Warnungen

Die Verhältnisse der Notenwerte können nicht bei Mensurwechsel geändert werden, weil sie nicht konstant sind. Zum Beispiel kann das Verhältnis 1 brevis = 3 semibrevis (tempus perfectum) manuell erstellt werden, indem folgende Variable erstellt wird:

```
breveTP = #(ly:make-duration -1 0 3 2)
...
{ c\breveTP f1 }
```

Hiermit wird die Variable `breveTP` auf den Wert „3/2 mal 2 = 3 mal eine Ganze“ gesetzt.

Die Symbole `mensural68alt` und `neomensural68alt` (alternative Symbole für 6/8) können nicht mit dem `\time`-Befehl. Anstelle dess muss `\markup {\musicglyph #"timesig.mensural68alt" }` benutzt werden.

Mensurale Notenköpfe

Für die Mensuralnotation kann ein Notenkopfstil ausgewählt werden, der sich vom Standard (`default`) unterscheidet. Dies wird erreicht, indem die `style`-Eigenschaft des Notenkopf-(`NoteHead`)-Objekts auf einen der Werte `baroque`, `neomensural`, `mensural`, `petrucci`, `blackpetrucci` oder `semipetrucci` gesetzt wird.

Der barocke (`baroque`) Stil unterscheidet sich vom Standard (`default`) folgendermaßen:

- Er stellt einen `maxima`-Notenkopf zur Verfügung und
- setzt eine eckige Form für die Brevis (`\breve`) ein.

Die Stile `neomensural`, `mensural` und `petrucci` unterscheiden sich vom barocken Stil folgendermaßen:

- Für Semibrevis und kleinere Notenwerte werden rhombenförmige Notenköpfe eingesetzt und
- die Hälse werden über den Kopf zentriert.
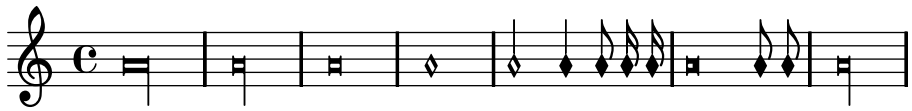
Der `blackpetrucci`-Stil erstellt Notenköpfe zur Benutzung für die schwarze Mensuralnotation oder *coloratio*-Abschnitten in der weißen Menusralnotation. Weil der Notenkopfstil nicht die Anzahl der Fähnchen beeinflusst, muss eine Semiminia in diesem Stil als `a8*2` notiert werden, nicht als `a4`, weil sie sonst wie eine Minima aussehen würde. Der Faktor, mit dem der Notenwert multipliziert wird, kann sich ändern, wenn *coloratio* etwa zur Notation von Triolen eingesetzt wurde.

Mit dem `semipetrucci`-Stil können halb-schwarze Notenköpfe notiert werden (Brevis, Longa und Maxima).

Das folgende Beispiel zeigt den Petrucci-Stil:

```
\set Score.skipBars = ##t
\autoBeamOff
\override NoteHead #'style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
```

```
\override NoteHead #'style = #'semipetrucci
a'\breve*5/6
\override NoteHead #'style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead #'style = #'petrucci
a'\longa
```



Siehe auch

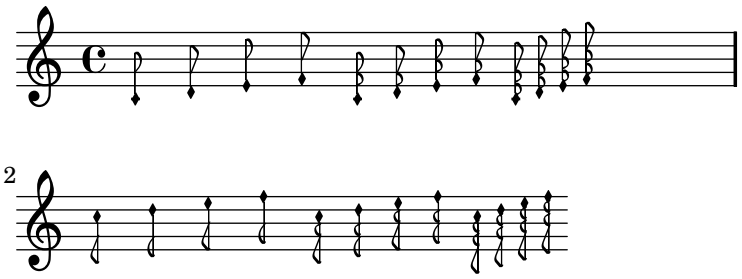
Glossar: Abschnitt “mensural notation” in *Glossar*, Abschnitt “note head” in *Glossar*.

Notationsreferenz: Abschnitt A.8 [Notenkopfstile], Seite 597.

Mensurale Fähnchen

Mit der Fähnchen-(`flag-style`)-Eigenschaft der graphischen Objekte „Hals“ (`Stem`) können auch Mensuralfähnchen gesetzt werden. Neben dem Standardstil (`default`) ist nur (`mensural`) unterstützt.

```
\override Stem #'flag-style = #'mensural
\override Stem #'thickness = #1.0
\override NoteHead #'style = #'mensural
\autoBeamOff
c'8 d'8 e'8 f'8 c'16 d'16 e'16 f'16 c'32 d'32 e'32 f'32 s8
c''8 d''8 e''8 f''8 c''16 d''16 e''16 f''16 c''32 d''32 e''32 f''32
```



Dabei ist die innerste Fahne immer vertikal auf eine Notenlinie ausgerichtet.

Es gibt keinen eigenen Stil für den neomensuralen oder Petrucci-Stil. Für die Notation des Gregorianischen Chorals gibt es keine Fähnchen.

Siehe auch

Glossar: Abschnitt “mensural notation” in *Glossar*, Abschnitt “flag” in *Glossar*.

Bekannte Probleme und Warnungen

Die Positionierung der Fähnchen an den Hälsen ist leicht verschoben.

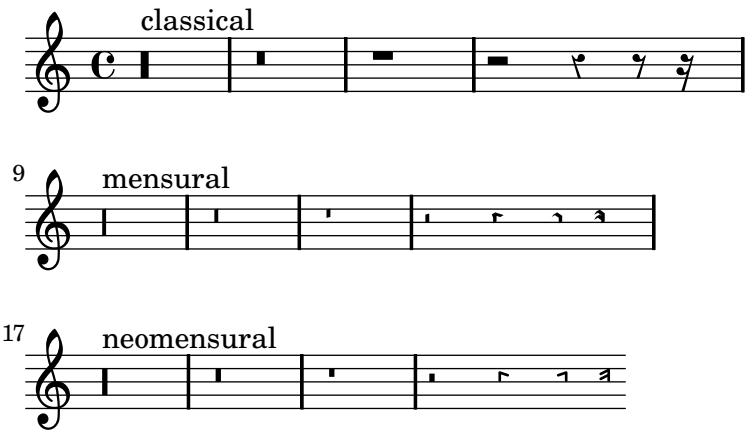
Vertikale Ausrichtung der Fähnchen an einer Notenlinie geht von der Annahme aus, dass der Hals entweder genau auf einer Notenlinie oder genau zwischen zwei Notenlinien endet. Das ist aber nicht unbedingt immer der Fall, weil LilyPond komplizierte Methoden zur Ermittlung des besten Layouts verwendet. Diese Methoden sollten aber eigentlich nicht zur Notation von mensuraler Musik eingesetzt werden.

Mensurale Pausen

Besondere Pausensymbole für die Notation der Alten Musik können mit der `style`-Eigenschaft des graphischen Objektes (grob) „Pause“ (`Rest`) angewählt werden. Unterstützte Stile sind klassisch (`classical`), `neomensural` und `mensural`. Der klassische (`classical`) Stil unterscheidet sich vom Standardstil (`default`) nur darin, dass die Viertelpause wie eine gespiegelte Achtelpause aussieht. Der mensurale und neomensurale Stil ahmt die Form von Pausen nach, wie man sie in Drucken bis zum 16. Jahrhundert finden kann.

Das folgende Beispiel demonstriert den mensuralen und den neomensuralen Stil:

```
\set Score.skipBars = ##t
\override Rest #'style = #'classical
r\longa^"classical" r\breve r1 r2 r4 r8 r16 s \break
\override Rest #'style = #'mensural
r\longa^"mensural" r\breve r1 r2 r4 r8 r16 s \break
\override Rest #'style = #'neomensural
r\longa^"neomensural" r\breve r1 r2 r4 r8 r16
```



Es gibt keine 32-stel- und 64-stel-Pausen für den mensuralen oder neomensuralen Stil. Anstatt dessen werden die Pausenformen des Standardstiles verwendet.

Eine Liste aller Pausen findet sich in Abschnitt “Ancient notation” in *Schnipsel*.

Siehe auch

Notationsreferenz: [Pausen], Seite 51.

Schnipsel: Abschnitt “Ancient notation” in *Schnipsel*.

Bekannte Probleme und Warnungen

Das Zeichen für die Maxima-Pause im mensuralen Stil ist eigentlich eine perfekte Longa-Pause: zwei (oder drei) Longa-Pausen müssen benutzt werden, um eine Maxima-Pause zu setzen. Longa-Pausen werden nicht automatisch gruppiert, sodass man das manuell vornehmen muss, indem man Pausen mit Tonhöhe einsetzt.

Mensurale Versetzungszeichen und Tonartbezeichnung

Der `mensural`-Stil stellt ein Kreuz und ein B zur Verfügung, die sich vom Standardstil unterscheiden. Wenn das Auflösungszeichen notiert wird, wird es aus dem `vaticana`-Stil gesetzt.

mensural



Der Stil für Versetzungszeichen und Vorzeichen wird durch die `glyph-name-alist`-Eigenschaft der Grobs `Accidental` und `KeySignature` bestimmt, also etwa folgendermaßen:

```
\override Staff.Accidental #'glyph-name-alist = #alteration-mensural-glyph-name-alist
```

Siehe auch

Glossar: Abschnitt “mensural notation” in *Glossar*, Abschnitt “Pitch names” in *Glossar*, Abschnitt “accidental” in *Glossar*, Abschnitt “key signature” in *Glossar*.

Notationsreferenz: Abschnitt 1.1 [Tonhöhen], Seite 1, [Versetzungszeichen], Seite 5, [Automatische Versetzungszeichen], Seite 24, [Tonartbezeichnung], Seite 20.
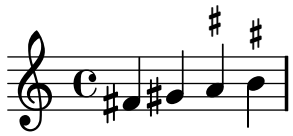
Referenz der Interna: Abschnitt “KeySignature” in *Referenz der Interna*.

Vorgeschlagene Versetzungszeichen (*musica ficta*)

In der europäischen Notation bis etwa 1600 wurde von Sängern erwartet, dass sie eigenständig Noten nach bestimmten Regeln chromatisch veränderten. Das wird als *musica ficta* bezeichnet. In modernen Transkriptionen werden diese Versetzungszeichen üblicherweise über die Note notiert.
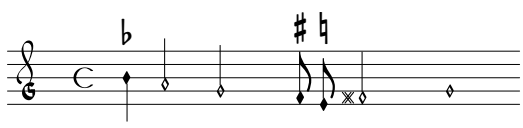
Es ist möglich, derartige Versetzungszeichen zu notieren, und die Anzeige kann zwischen normaler Satzweise und musica ficta hin- und hergewechselt werden. Hierzu muss `suggestAccidentals` auf wahr gesetzt werden:

```
fis gis
\set suggestAccidentals = ##t
ais bis
```



Damit wird *jedes* folgende Versetzungszeichen über dem System gesetzt werden, bis die Eigenschaft mit `\set suggestAccidentals = ##f` wieder zum Standardverhalten verändert wurde. Eine praktischere Lösung ist es, `\once \set suggestAccidentals = ##t` zu benutzen, was man als Variable definieren kann:

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative c''
  \new MensuralVoice  {
        \once \set suggestAccidentals = ##t
  bes4 a2 g2 \ficta fis8 \ficta e! fis2 g1
  }
}
```



Siehe auch

Referenz der Interna: Abschnitt “Accidental_engraver” in *Referenz der Interna*, Abschnitt “AccidentalSuggestion” in *Referenz der Interna*.

Weiße Mensuralligaturen

Begrenzte Unterstützung für Ligaturen der weißen Mensuralnotation ist vorhanden.

Um weiße Mensuralligaturen zu benutzen, muss innerhalb des Layout-Blocks im `Voice`-Kontext der `Mensural_ligature_engraver` aktiviert werden und gleichzeitig der `Ligature_bracket_engraver` (der die Klammern über den Noten setzt) entfernt werden, wie im Beispiel.

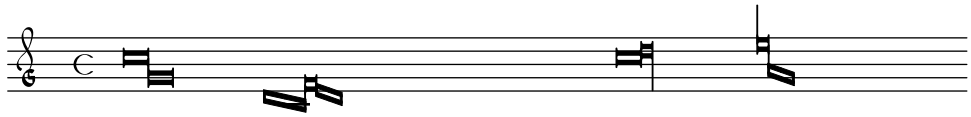
```
\layout {
  \context {
    \Voice
    \remove Ligature_bracket_engraver
    \consists Mensural_ligature_engraver
  }
}
```

Zusätzlich zu diesen Einstellungen gibt es keine eigenen Befehle, die die Form einer Ligatur bestimmen. Die Form wird vielmehr aus Tonhöhen und Tondauern der in Klammern gesetzten Noten geschlossen. Diese Herangehensweise erfordert einige Eingewöhnung, hat aber den großen Vorteil, dass der musikalische Inhalt der Ligatur dem Programm bekannt ist. Das ist nicht nur notwendig für korrekte MIDI-Ausgabe, sondern erlaubt es auch, automatische Transkriptionen von Ligaturen anzufertigen.
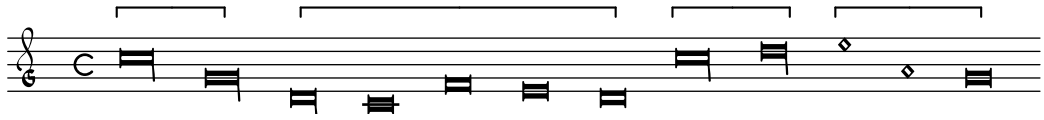
An bestimmten Stellen können zwei aufeinanderfolgende Noten entweder als zwei Quadrate oder eine Obliqua (Flexa) dargestellt werden. In derartigen Fällen ist die Quadratform der Standard, aber die Obliqua kann verlangt werden, indem man die `ligature-flexa`-Eigenschaft des *zweiten* Notenkopfes setzt. Die Länge der Obliqua kann durch die Notenkopfeigenschaft `flexa-width` definiert werden.

Eine Datei kann zum Beispiel so aussehen:

```
\score {
  \transpose c c' {
    \set Score.timing = ##f
    \set Score.defaultBarType = "empty"
    \override NoteHead #'style = #'petrucci
    \override Staff.TimeSignature #'style = #'mensural
    \clef "petrucci-g"
    \[ c'\maxima g \]
    \[ d\longa
       \override NoteHead #'ligature-flexa = ##t
       \once \override NoteHead #'flexa-width = #3.2
       c\breve f e d \]
    \[ c'\maxima d'\longa \]
    \[ e'1 a g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove Ligature_bracket_engraver
      \consists Mensural_ligature_engraver
    }
  }
}
```



Wenn der `Ligature_bracket_engraver` nicht durch den `Mensural_ligature_engraver` ersetzt wird, werden die Noten wie folgt ausgegeben:



Siehe auch

Glossar: Abschnitt “ligature” in *Glossar*.

Notationreferenz: [Ligaturen der gregorianischen Quadratnotation], Seite 407, [Ligaturen], Seite 394.

Bekannte Probleme und Warnungen

Die horizontale Positionierung ist sehr schlecht.

Versetzungszeichen können mit vorhergehenden Noten kollidieren.

2.9.4 Gregorianischen Choral setzen

Wenn ein gregorianischer Choral notiert wird, wählt der `Vaticana_ligature_engraver` automatisch die richtigen Notenköpfe aus, so dass man den Notenkopfstil nicht explizit setzen muss. Der Stil kann dennoch gesetzt werden, etwa auf `vaticana_punctum` um punctum-Neumen zu erzeugen. Ähnlich funktioniert auch der `Mensural_ligature_engraver`, der Mensuralligaturen setzt.

Siehe auch

Glossar: Abschnitt “ligature” in *Glossar*.

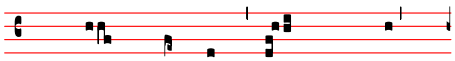
Notationreferenz: [Weiße Mensuralligaturen], Seite 402, [Ligaturen], Seite 394.

Gregorianische Gesangs-Kontexte

Die vordefinierten Kontexte `VaticanaVoice` (für eine gregorianische Stimme) und `VaticanaStaff` (für ein gregorianisches Notensystem) können eingesetzt werden, um Gregorianischen Choral im Stil der Editio Vaticana zu setzen. Diese Kontexte initialisieren alle relevanten Eigenschaften für das Notensystem und die graphischen Objekte, so dass unmittelbar mit der Notation begonnen werden kann. Siehe das folgende Beispiel:

```
\include "gregorian.ly"
\score {
  <<
    \new VaticanaVoice = "cantus" {
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new Lyrics \lyricsto "cantus" {
      San- ctus, San- ctus, San- ctus
    }
```

```
  >>
}
```



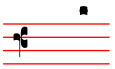
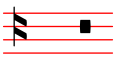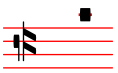
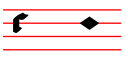
San- ctus, San- ctus,



San- ctus

Gregorianische Schlüssel

Die folgende Tabelle zeigt alle Schlüssel für den gregorianischen Choral, die mit dem `\clef`-Befehl unterstützt sind. Einige Schlüssel benutzen das selbe Zeichen, unterscheiden sich aber in der Notenlinie, auf der der Schlüssel gesetzt wird. In diesem Fall wird eine Nummer benutzt, die die Notenlinie von unten nach oben kennzeichnet. Man kann die Schlüssel aber auch manuell auf eine bestimmte Notenlinie zwingen, wie gezeigt in [Notenschlüssel], Seite 16. Die Note, die rechts von den Schlüsseln im Beispiel gezeigt wird, ist ein `c'` in Bezug auf den aktuellen Schlüssel.

Beschreibung	**unterstützter Schlüssel**	**Beispiel**
Do-Schlüssel der Editio Vaticana	`vaticana-do1`, `vaticana-do2`, `vaticana-do3`	
Fa-Schlüssel der Editio Vaticana	`vaticana-fa1`, `vaticana-fa2`	
Do-Schlüssel der Editio Medicaea	`medicaea-do1`, `medicaea-do2`, `medicaea-do3`	
Fa-Schlüssel der Editio Medicaea	`medicaea-fa1`, `medicaea-fa2`	
Hufnagel Do-Schlüssel für den historischen Stil	`hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3`	
Hufnagel Fa-Schlüssel für den historischen Stil	`hufnagel-fa1`, `hufnagel-fa2`	
Kombinierter Do/Fa-Hufnagelschlüssel für den historischen Stil	`hufnagel-do-fa`	

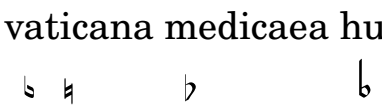
Siehe auch

Glossar: Abschnitt “clef” in *Glossar*.

Notationsreferenz: [Notenschlüssel], Seite 16.

Gregorianische Versetzungszeichen und Tonartbezeichnung

Es gibt Versetzungszeichen in drei unterschiedlichen Stilen für die Notation des gregorianischen Chorals:



Wie zu sehen ist, werden nicht alle Versetzungszeichen von jedem Stil unterstützt. Wenn versucht wird, ein Versetzungszeichen zu notieren, das von einem bestimmten Stil nicht unterstützt wird, wechselt LilyPond zu einem anderen Stil.

Der Stil für Versetzungs- und Vorzeichen wird von der `glyph-name-alist`-Eigenschaft der Grobs `Accidental` und `KeySignature` kontrolliert, beispielsweise:

```
\override Staff.Accidental #'glyph-name-alist = #alteration-mensural-glyph-name-alist
```

Siehe auch

Glossar: Abschnitt “accidental” in *Glossar*, Abschnitt “key signature” in *Glossar*.

Notationsreferenz: Abschnitt 1.1 [Tonhöhen], Seite 1, [Versetzungszeichen], Seite 5, [Automatische Versetzungszeichen], Seite 24, [Tonartbezeichnung], Seite 20.

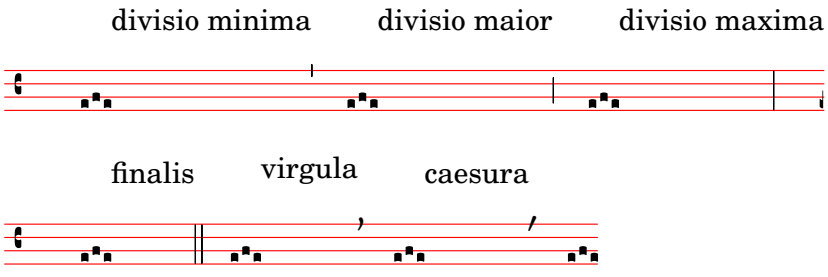
Referenz der Interna: Abschnitt “KeySignature” in *Referenz der Interna*.

Divisiones

Die Notation des gregorianischen Chorals benutzt keine Pausen, anstatt dessen werden *Divisiones* eingesetzt.

Eine *divisio* (Plural: *divisiones*; Latein: „Teilung“) ist ein Symbol des Notensystemkontextes, das benutzt wird, um Phrasierung und Abschnitte im Gregorianischen Choral anzuzeigen. Die musikalische Bedeutung von *divisio minima*, *divisio maior* und *divisio maxima* kann beschrieben werden als kurze, mittlere und lange Pause, ungefähr wie die Atemzeichen aus dem Abschnitt [Atemzeichen], Seite 120. Das *finalis*-Zeichen bezeichnet nicht nur das Ende eines Chorals, sondern wird auch oft innerhalb eines Antiphons/Responsoriums benutzt, um das Ende eines Abschnitts anzuzeigen.

Divisiones können benutzt werden, indem die Datei ‘`gregorian.ly`’ in die Quelldatei eingefügt wird. Hier sind die entsprechenden Definitionen schon abgelegt, so dass es genügt, die Befehle `\divisioMinima`, `\divisioMaior`, `\divisioMaxima` und `\finalis` an den entsprechenden Stellen zu schreiben. Einige Editionen verwenden eine *virgula* oder *caesura* anstelle der divisio minima. Darum findet sich in der Datei ‘`gregorian.ly`’ auch eine Definition für `\virgula` und `\caesura`.



Vordefinierte Befehle

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

Siehe auch

Glossary: Abschnitt “caesura” in *Glossar*, Abschnitt “divisio” in *Glossar*.

Notationsreferenz: [Atemzeichen], Seite 120.

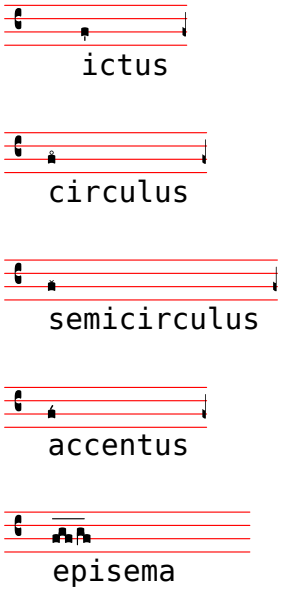
Installierte Dateien: ‘`gregorian.ly`’.

Referenz der Interna: Abschnitt “BreathingSign” in *Referenz der Interna*.

Artikulationszeichen des Gregorianischen Chorals

Zusätzlich zu den Standardartikulationszeichen, wie sie im Abschnitt [Artikulationszeichen und Verzierungen], Seite 106 beschrieben werden, werden auch Artikulationszeichen für die Notation des Editio Vaticana-Stils zur Verfügung gestellt.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \override TextScript #'font-family = #'typewriter
    \override TextScript #'font-shape = #'upright
    \override Script #'padding = #-0.1
    a\ictus_"ictus " \bar "" \break
    a\circulus_"circulus " \bar "" \break
    a\semicirculus_"semicirculus " \bar "" \break
    a\accentus_"accentus " \bar "" \break
    \[ a_"episema" \episemInitium \pes b \flexa a b \episemFinis \flexa a \]
  }
}
```



Siehe auch

Notationreferenz: [Artikulationszeichen und Verzierungen], Seite 106.

Schnipsel: Abschnitt “Ancient notation” in *Schnipsel*.

Referenz der Interna: Abschnitt “Episema” in *Referenz der Interna*, Abschnitt “EpisemaEvent” in *Referenz der Interna*, Abschnitt “Episema_engraver” in *Referenz der Interna*, Abschnitt “Script” in *Referenz der Interna*, Abschnitt “ScriptEvent” in *Referenz der Interna*, Abschnitt “Script_engraver” in *Referenz der Interna*.
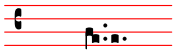
Bekannte Probleme und Warnungen

Einige Artikulationszeichen sind vertikal zu dich an den entsprechenden Notenköpfen gesetzt.

Augmentationspunkte (*morae*)

Verlängerungspunkte, auch als *morae* bezeichnet, werden mit der Musikfunktion `\augmentum` hinzugefügt. Es handelt sich um eine eigenständige Funktion und nicht um einen Präfix, der zu einer Note gehört. Die Funktion wirkt sich nur auf den direkt vorhergehenden musik. Ausdruck aus. Das heißt, dass `\augmentum \virga c` keine sichtbare Wirkung hat. Anstelle dessen sollte geschrieben werden: `\virga \augmentum c` oder `\augmentum {\virga c}`. Man kann `\augmentum {a g}` als Kurznotation für `\augmentum a \augmentum g` schreiben.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



Siehe auch

Notationsreferenz: [Atemzeichen], Seite 120.

Referenz der Interna: Abschnitt “BreathingSign” in *Referenz der Interna*.

Schnipsel: Abschnitt “Ancient notation” in *Schnipsel*.

Ligaturen der gregorianischen Quadratnotation

Beschränkte Unterstützung für gregorianische Quadratneumen-Ligaturen (nach dem Stil der Editio Vaticana) ist vorhanden. Die wichtigsten Ligaturen können schon gesetzt werden, aber wichtige Eigenschaften anspruchsvoller Typographie wie horizontale Ausrichtung von mehreren Ligaturen, korrekte Silbenpositionierung und richtiger Umgang mit Versetzungszeichen fehlen noch.

Die Unterstützung für gregorianische Neumen wird aktiviert, indem man mit `\include` die Datei ‘`gregorian.ly`’ am Anfang der Quelldatei aktiviert. Damit werden zusätzliche Befehl zur Verfügung gestellt, mit denen man die Neumensymbole des Chorals produzieren kann.

Notenköpfe können verändert und/bzw. verbunden werden.

- Die Form des Notenkopf kann verändert werden, indem man *vor* die Noten folgende Befehle schreibt: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\ascendens`, `\descendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Eigentliche Ligaturen (also Noten, die miteinander verbunden sind), werden erstellt, indem man einen der verbindenden Befehle, `\pes` oder `\flexa` für Aufwärts- bzw. Abwärtsbewegung, zwischen die zu verbindenden Noten setzt.

Eine Notenbezeichnung ohne jeglichen Modifikator produziert ein *punctum*. Alle anderen Neumen, auch einzelne Noten-Neumen mit einer anderen Form als der *Virga* werden generell als Ligaturen betrachtet und deshalb von den Zeichen `\[...\]` eingeklammert werden.

Einzelne Noten-Neumen:

- Das *punctum* ist die grundlegende Notenform (im *Vaticana*-Stil: ein Quadrat mit gebogenen Ober- und Unterkanten). Zusätzlich gibt es auch noch das oblique *punctum inclinatum*, das mit dem Präfix `\inclinatum` erstellt wird. Das normale *punctum* kann durch `\cavum`

verändert werden, wodurch eine hohle Note erstellt wird, und durch `\linea`, wodurch vertikale Linien zu den Seiten der Note gezogen werden.

- Die *virga* hat einen absteigenden Hals auf der rechten Seite. Sie wird durch den Modifikator `\virga` erstellt.

Ligaturen

Anders als in anderen Neumennotationssystemen, wird das typographische Aussehen einer Ligatur nicht durch Eingabebefehle direkt vorgegeben, sondern richtet sich nach bestimmten Darstellungsregeln, die durch die musikalische Bedeutung bestimmt werden. Eine Ligatur mit drei Noten beispielsweise, mit der Form tief-hoch-tief, wie etwa `\[ a \pes b \flexa g \]`, ergibt einen Torculus, der aus drei Punctum-Köpfen besteht, während die Form hoch-tief-hoch, wie etwa `\[ a \flexa g \pes b \]`, einen Porrectus mit einer gebogenen Flexa und nur einem Punctum-Kopf ergibt. Es gibt keinen Befehl, mit dem explizit eine gebogene Flexa gesetzt werden können; die Entscheidung, wann eine derartige Form im Notenbild vorkommen soll, wird durch die musikalische Bedeutung der Noten vorgegeben. Die Idee hinter dieser Art der Eingabe ist es, dass der musikalische Inhalt von der graphischen Ausgabe getrennt wird. Dadurch wird es möglich, die gleiche Quelldatei zu benutzen, um beispielsweise die Noten in einem anderen Stil darzustellen.

Liquescente Neumen

Eine weitere Hauptkategorie der Notation von gregorianischem Choral sind die sogenannten liquescenten Neumen. Sie werden unter bestimmten Umständen am Ende einer Silbe eingesetzt, die auf einen „liquescenten“ Buchstaben endet (das sind die Konsonanten, die eine Tonhöhe haben können, also die Nasale, l, r, v, j und ihre diphtongalen Entsprechungen). Liquescente Neumen werden also nie alleine eingesetzt (auch wenn sie isoliert produziert werden können) und treten immer am Ende einer Silbe auf.

Liquescente Neumen werden graphisch auf zwei Arten dargestellt: mit einer kleineren Note oder indem die Hauptnote nach oben bzw. unten „gedreht“ wird. Die erste Darstellungsweise erreicht man, indem einen normalen `pes` oder `flexa` schreibt und dann die Form der zweiten Note verändert: `\[ a \pes \deminutum b \]` . Die zweite Darstellungsweise erreicht man, indem die Form einer einzelnen Neume mit `\auctum` und einem der Richtungsanzeiger `\descendens` bzw. `\ascendens` versieht: `\[ \auctum \descendens a \]` .

Spezielle Zeichen

Eine dritte Kategorie besteht aus einer kleinen Anzahl an Zeichen mit einer besonderen Bedeutung: die *quilisma*, der *oriscus* und der *strophicus*. Sie werden notiert, indem man vor die entsprechende Note den Modifikator `\quilisma`, `\oriscus` oder `\stropha` schreibt.

Im Grunde kann innerhalb der Ligaturbegrenzer `\[` und `\]` eine beliebige Anzahl ans Notenköpfen eingefügt werden und Präfixe wie `\pes`, `\flexa`, `\virga`, `\inclinatum` usw. können beliebig untereinander kombiniert werden. Der Einsatz der Regeln, mit denen die Ligaturen konstruiert werden, wird entsprechend angepasst. Auf diese Art kann eine unendliche Anzahl an Ligaturen erstellt werden.

Die Benutzung der Notationszeichen folgt allerdings bestimmten Regeln, die nicht von LilyPond überprüft werden. Die *quilisma* beispielsweise findet sich immer als mittlere Note einer aufsteigenden Ligatur und fällt üblicherweise auf einen Halbtonschritt, aber es ist durchaus möglich, wenn auch nicht *richtig*, eine Quilisma bestehend aus einer Note zu notieren.

Neben den Notenformen definiert die Datei ‘`gregorian.ly`’ auch die Befehle `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ` und `\IIJ`, mit denen die entsprechenden Zeichen, etwa für den Text oder als Abschnittsmarkierung erstellt werden können. Diese Befehl benutzen bestimmte Unicode-Zeichen und funktionieren nur, wenn eine Schriftart vorhanden ist, die diese Zeichen unterstützt.

In der folgenden Tabelle wird eine begrenzte, aber dennoch repräsentative Anzahl an Ligaturen der Neumennotation dargestellt, denen Fragmente beigefügt sind, die die Notation in LilyPond zeigen. Die Tabelle basiert auf der erweiterten Neumentabelle des zweiten Bands des Antiphonale Romanum (*Liber Hymnarius*), 1983 von den Mönchen von Solsemes herausgegeben. Die erste Spalte zeigt die Bezeichnungen der Ligaturen, fett für die Normalform, kursiv für die liquescente Form. Die dritte Spalte zeigt Code-Schnipsel, mit denen die Ligatur notiert werden kann, wobei die Noten `g`, `a` und `b` als Tonhöhen eingesetzt werden.

Neumen aus einzelnen Noten

Grundform und *liquescente Form*	**Ausgabe**	**LilyPond-Code**
Punctum		`\[ b \]`
		`\[ \cavum b \]`
		`\[ \linea b \]`
Punctum Auctum Ascendens		`\[ \auctum \ascendens b \]`
Punctum Auctum Descendens		`\[ \auctum \descendens b \]`
Punctum inclinatum		`\[ \inclinatum b \]`
Punctum Inclinatum Auctum		`\[ \inclinatum \auctum b \]`
Punctum Inclinatum Parvum		`\[ \inclinatum \deminutum b \]`

Virga



Ligaturen aus zwei Noten

Clivis vel Flexa

```
\[ b \flexa g \]
```



Clivis Aucta Descendens

```
\[ b \flexa \auctum \descendens
g \]
```



Clivis Aucta Ascendens

```
\[ b \flexa \auctum \ascendens
g \]
```



Cephalicus

```
\[ b \flexa \deminutum g \]
```



Podatus/Pes

```
\[ g \pes b \]
```



Pes Auctus Descendens

```
\[ g \pes \auctum \descendens b
\]
```



Pes Auctus Ascendens

```
\[ g \pes \auctum \ascendens b
\]
```



Epiphonus

```
\[ g \pes \deminutum b \]
```



Pes Initio Debilis

```
\[ \deminutum g \pes b \]
```



Pes Auctus Descendens Initio Debilis

```
\[ \deminutum g \pes \auctum
\descendens b \]
```



Ligaturen mit mehr als zwei Noten

Torculus

```
\[ a \pes b \flexa g \]
```



Torculus Auctus Descendens

```
\[ a \pes b \flexa \auctum
\descendens g \]
```



Torculus Deminutus

```
\[ a \pes b \flexa \deminutum g
\]
```



Torculus Initio Debilis

```
\[ \deminutum a \pes b \flexa g
\]
```



Torculus Auctus Descendens Initio Debilis

```
\[ \deminutum a \pes b \flexa
\auctum \descendens g \]
```



Torculus Deminutus Initio Debilis

```
\[ \deminutum a \pes b \flexa
\deminutum g \]
```



Porrectus

```
\[ a \flexa g \pes b \]
```



Porrectus Auctus Descendens

```
\[ a \flexa g \pes \auctum
\descendens b \]
```



Porrectus Deminutus

```
\[ a \flexa g \pes \deminutum b
\]
```



Climacus

```
\[ \virga b \inclinatum a
\inclinatum g \]
```



Climacus Auctus

```
\[ \virga b \inclinatum a
\inclinatum \auctum g \]
```



Climacus Deminutus

```
\[ \virga b \inclinatum a
\inclinatum \deminutum g \]
```



Scandicus

```
\[ g \pes a \virga b \]
```



Scandicus Auctus Descendens

```
\[ g \pes a \pes \auctum
\descendens b \]
```



Scandicus Deminutus

```
\[ g \pes a \pes \deminutum b \]
```



Special Signs

Quilisma

```
\[ g \pes \quilisma a \pes b \]
```



Quilisma Pes Auctus Descendens

```
\[ \quilisma g \pes \auctum
\descendens b \]
```



Oriscus

```
\[ \oriscus b \]
```


Pes Quassus

```
\[ \oriscus g \pes \virga b \]
```



Pes Quassus Auctus Descendens

```
\[ \oriscus g \pes \auctum
\descendens b \]
```



Salicus

```
\[ g \oriscus a \pes \virga b \]
```



Salicus Auctus Descendens

```
\[ g \oriscus a \pes \auctum
\descendens b \]
```



(Apo)stropha

```
\[ \stropha b \]
```



Stropha Aucta

```
\[ \stropha \auctum b \]
```



Bistropha

```
\[ \stropha b \stropha b \]
```



Tristropha

```
\[ \stropha b \stropha b
\stropha b \]
```



Trigonus

```
\[ \stropha b \stropha b
\stropha a \]
```



Vordefinierte Befehle

Folgende Notenpräfixe sind unterstützt: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Präfixe können kombiniert werden, wenn es hier auch Begrenzungen gibt. Zum Beispiel können die Präfixe `\descendens` oder `\ascendens` vor einer Note geschrieben werden, aber nicht beide für die selbe Note.

Zwei benachbarte Noten können mit den `\pes` und `\flexa`-Infixen verbunden werden, um eine steigende bwz. fallende Melodielinie zu notieren.

Die musikalische Funktion `\augmentum` muss benutzt werden, um augmentum-Punkte hinzuzufügen.

Siehe auch

Glossar: Abschnitt “ligature” in *Glossar*.

Notationreferenz: ⟨undefined⟩ [Ligaturen der gregorianischen Quadratnotation], Seite ⟨undefined⟩, [Weiße Mensuralligaturen], Seite 402, [Ligaturen], Seite 394.

Bekannte Probleme und Warnungen

Wenn ein `\augmentum`-Punkt am Ende des letzten Systems innerhalb einer Ligatur gesetzt wird, ist er vertikal etwas falsch positioniert. Als Abhilfe kann eine unsichtbare Note (z. B. `s8`) als letzte Note im System eingegeben werden.

`\augmentum` sollte als Präfix implementiert sein, nicht als eigene musikalische Funktion, so dass `\augmentum` mit den anderen Präfixen in arbiträrer Reihenfolge notiert werden kann.

2.9.5 Musiksatz Alter Musik in der Praxis – Szenarien und Lösungen

Wenn man mit Alter Notation zu tun hat, fallen oft Aufgaben an, die in der modernen Notation nicht vorkommen, für welche LilyPond geschaffen wurde. In diesem Abschnitt sollen darum einige praktische Problemstellungen und Lösungsvorschläge dargestellt werden. Dabei handelt es sich um:

- wie man Incipite in modernen Editionen von Mensuralnotation notieren kann (d.h. ein kleiner Abschnitt vor der eigentlichen Partitur, der die Originalnotenformen darstellt),
- wie man *Mensurstriche* einstellt, mit denen oft moderne Transkriptionen polyphoner Musik notiert werden,
- wie man den gregorianischen Choral mit moderner Notation darstellt und
- wie man sowohl ein Mensuralnotationsbild als auch eine moderne Edition aus der selben Quelle erstellt.

Incipite

In Arbeit.

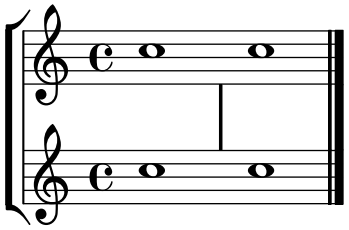
Mensurstriche

Als *Mensurstriche* wird ein Notenlayout bezeichnet, in dem die Taktlinien nicht auf den Systemen, sondern nur zwischen Systemen gezogen werden. Damit soll signalisiert werden, dass das Original keine Takteinteilung besessen hat und etwa Synkopen nicht über Taktlinien hinweg

aufgeteilt werden müssen, während man sich dennoch an den Taktlinien rhythmisch orientieren kann.

Das Mensurstiche-Layout, in welchem die Taktlinien nicht auf den Systemen, sondern zwischen den Systemen gesetzt werden, kann mit einer `StaffGroup` anstelle von `ChoirStaff` erreicht werden. Die Taktlinien auf den Systemen werden mit der `transparent`-Eigenschaft ausgelöscht.

```
global = {
  \override Staff.BarLine #'transparent = ##t
  s1 s
  % the final bar line is not interrupted
  \revert Staff.BarLine #'transparent
  \bar "|."
}
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Gregorianischen Choral transkribieren

Gregorianischer Choral kann mit einigen einfachen Einstellungen in moderner Notation notiert werden.

Hälse. Hälse können meistens weggelassen werden, was geschieht, indem man den `Stem_engraver` aus dem Stimmenkontext entfernt:

```
\layout {
  ...
  \context {
    \Voice
      \remove "Stem_engraver"
  }
}
```

In einigen Transkriptionsstilen werden jedoch teilweise Hälse eingesetzt, um etwa den Übergang von einem Einton-Rezitativ zu einer melodischen Geste anzuzeigen. In diesem Fall können Hälse entweder mit `\override Stem #'transparent = ##t` unsichtbar gemacht werden oder mit `\override Stem #'length = #0` auf die Länge von 0 reduziert werden. Die Hälse müssen dann wieder an den entsprechenden Stellen mit `\once \override Stem #'transparent = ##f` sichtbar gemacht werden (siehe auch Beispiel unten).

Takt. Für Gesang ohne Metrum gibt es einige Alternativen.

Der `Time_signature_engraver` kann aus dem `Staff`-Kontext entfernt werden, ohne dass es negative Seiteneffekte gäbe. Alternativ kann er durchsichtig gemacht werden, dabei entsteht aber ein leerer Platz zu Beginn der Noten an der Stelle, wo normalerweise die Taktangabe stehen würde.

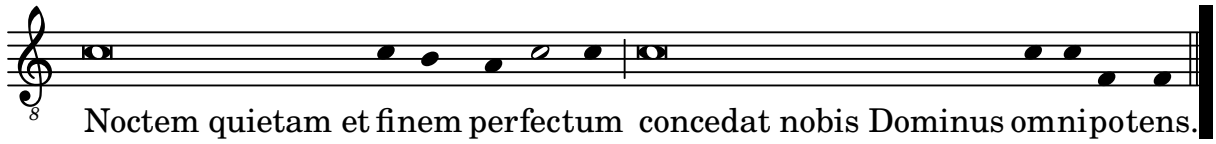
In vielen Fällen ergibt `\set Score.timing = ##f` gute Ergebnisse. Eine andere Möglichkeit ist es, `\CadenzaOn` und `\CadenzaOff` zu benutzen.

Um Taktstriche zu entfernen, kann man radikal den `Bar_engraver` aus dem `Staff`-Kontext entfernen. Wenn man ab und zu einen Taktstrich braucht, sollten die Striche nur mit `\override BarLine #'transparent = ##t` unsichtbar gemacht werden.

Oft werden Rezitativtöne mit einer Brevis angezeigt. Der Text für die Rezitativnote kann auf zwei Arten notiert werden: entweder als einzelne, links ausgerichtete Silbe:

```
\include "gregorian.ly"
chant = \relative c' {
  \clef "G_8"
  c\breve c4 b4 a c2 c4  \divisioMaior
  c\breve c4 c f, f \finalis
}

verba = \lyricmode {
  \once \override LyricText #'self-alignment-X = #-1
  "Noctem quietam et" fi -- nem per -- fec -- tum
  \once \override LyricText #'self-alignment-X = #-1
  "concedat nobis Dominus" om -- ni -- po -- tens.
}
\score {
  \new Staff <<
  \new Voice = "melody" \chant
  \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \override Stem #'transparent = ##t
    }
  }
}
```
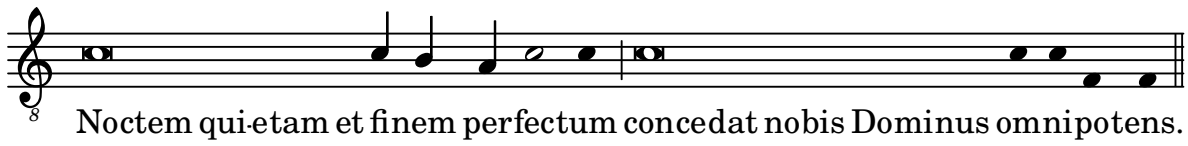


Das funktioniert gut, solange der Text nicht über einen Zeilenumbruch reicht. In diesem Fall kann man etwa die Noten der Silben verstecken (hier werden auch die Hälse unsichtbar gemacht):

```
\include "gregorian.ly"
chant = \relative c' {
  \clef "G_8"
  \set Score.timing = ##f
  c\breve \override NoteHead #'transparent = ##t  c c c c c
  \revert NoteHead #'transparent
  \override Stem #'transparent = ##f \stemUp c4 b4 a
  \override Stem #'transparent = ##t c2 c4  \divisioMaior
  c\breve \override NoteHead #'transparent = ##t c c c c c c c
```

```
  \revert NoteHead #'transparent c4 c f, f \finalis
}

verba = \lyricmode {
  No -- ctem qui -- e -- tam et fi -- nem per -- fec -- tum
  con -- ce -- dat no -- bis Do -- mi -- nus om -- ni -- po -- tens.
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics \lyricsto "melody" \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \override BarLine #'transparent = ##t
      \override Stem #'transparent = ##t
    }
  }
}
```



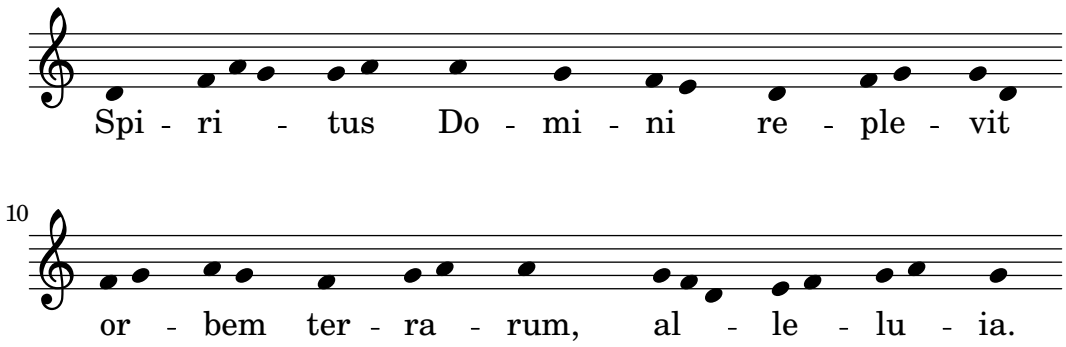
Eine andere übliche Situation ist die Transkription von neumatischem oder melismatischem Gesang, d.h. Gesang, der eine unterschiedliche Anzahl von Noten pro Silbe hat. In diesem Fall sollen die Silbengruppen üblicherweise deutlich voneinander getrennt gesetzt werden, oft auch die Untergruppen eines längeren Melismas. Eine Möglichkeit, das zu erreichen, ist es, eine feste Taktart, etwa 1/4, zu benutzen und dann jeder Silbe oder Notengruppe einen ganzen Takt zuzuweisen, u.U. mit Hilfe von Triolen und kleinen Notenwerten. Wenn die Taktstriche und alle anderen rhythmischen Anweisungen unsichtbar gemacht werden, und der Platz um die Taktstriche vergrößert wird, ergibt sich eine recht gute Repräsentation der Originalnotation.

Damit Silben mit unterschiedlicher Länge (etwa „-ri“ und „-rum“) die Silbengruppen nicht ungleichmäßig aufweiten, kann die `#'X-extent`-Eigenschaft des `LyricText`-Objekts auf einen festen Wert gesetzt werden. Eine andere Möglichkeit wäre es, die Silben als Textbeschriftung einzufügen. Wenn weitere horizontale Anpassungen nötig sind, können sie mit unsichtbaren (`s`)-Noten vorgenommen werden.

```
spiritus = \relative c' {
  \time 1/4
  \override Lyrics.LyricText #'X-extent  = #'(0 . 3)
  d4 \times 2/3 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4  s
  \times 2/3 { g8 f d } e f g a g4
}

spirLyr = \lyricmode {
  Spi -- ri -- _ _ tus  _ Do -- mi -- ni  _ re -- ple -- _ vit _
  or -- _ bem _  ter -- ra -- _ rum, al -- _ _ le -- _ lu
```

```
  -- _ ia.
}
\score {
  \new Staff <<
    \new Voice = "chant" \spiritus
    \new Lyrics = "one" \lyricsto "chant" \spirLyr
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \override BarLine #'X-extent = #'(-1 . 1)
      \override Stem #'transparent = ##t
      \override Beam #'transparent = ##t
      \override BarLine #'transparent = ##t
      \override TupletNumber #'transparent = ##t
    }
  }
}
```



Alte und moderne Edition aus einer Quelldatei

In Arbeit.

Herausgeberische Anmerkungen

In Arbeit.

2.10 Weltmusik

Dieser Abschnitt soll Besonderheiten der Notation aufzeigen, die insbesondere relevant sind, um Musik nicht-westlicher Tradition zu notieren.

2.10.1 Übliche Notation für nichteuropäische Musik

Dieser Abschnitt zeigt, wie man Partituren erstellt, die nicht der europäischen klassischen Musiktradition angehören.

Erweiterung von Notation und Stimmungssystemen

Die klassische Standardnotation wird üblicherweise zur Notation verschiedenster Musikarten benutzt und ist nicht auf die „klassische Musik“ beschränkt. Diese Notation wird behandelt in Abschnitt 1.1.1 [Tonhöhen setzen], Seite 1, und die unterschiedlichen Notenbezeichnungen, die eingesetzt werden können, finden sich in [Notenbezeichnungen in anderen Sprachen], Seite 7.

Viele nicht-europäische Musik (und auch manche europäische Volksmusik) benutzt jedoch alternative oder erweiterte Skalen (Tonleitern), die man nicht mit der normalen westlichen Notation notieren kann.

In einigen Fällen wird die klassische Notation dennoch benutzt, wobei man die Tonhöhenunterschiede implizit mitliest. Beispielsweise arabische Musik wird mit normalen Halb- und Vierteltonversetzungszeichen notiert und die exakte Tonhöhe (die etwas von der notierten abeweichen kann) dann aus dem Kontext erschlossen. Italienische Notenbezeichnungen werden normalerweise benutzt, und die Init-Datei '`arabic.ly`' stellt eine Anzahl an Makros zur Verfügung, die die Standardnoation erweitern. Siehe auch Abschnitt 2.10.2 [Arabische Musik], Seite 419.

Andere Musik brauchen erweiterte oder ganz einzigartige Notation. Die klassische Musik der Türkei, oder ottomanische Musik, benutzt melodische Formen, die als *makamlar* bekannt sind und deren Intervalle auf 1/9-Bruchteilen des Ganztones beruhen. Die normale europäische Notation wird trotzdem auf dem System mit normalen Noten benutzt mit speziellen türkischen Versetzungszeichen. Diese Versetzungszeichen sind in der Datei '`makam.ly`' definiert. Zu weiterer Information über die klassische türkische Musik und Makamlar, siehe Abschnitt 2.10.3 [Türkische klassische Musik], Seite 425.

Um Dateien wie '`arabic.ly`' oder '`makam.ly`' zu finden, siehe Abschnitt "Mehr Information" in *Handbuch zum Lernen*.
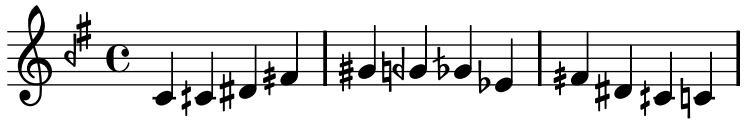
Ausgewählte Schnipsel

Makam-Beispiel

Makam ist eine türkische Melodie, in der 1/9-Tonabstände eingesetzt werden. Sehen Sie sich die Initialisierungsdatei '`makam.ly`' für weiter Information zu Tonhöhenbezeichnungen und Alterationen an (siehe Handbuch zum Lernen 2.14.1, 4.6.3 Weitere Information zu Hinweisen, wo diese Datei gespeichert ist).

```
% Initialize makam settings
\include "makam.ly"

\relative c' {
  \set Staff.keySignature = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```



Siehe auch

Glossar: Abschnitt "Common Practice Period" in *Glossar*, Abschnitt "makamlar" in *Glossar*.

Handbuch zum Lernen: Abschnitt "Mehr Information" in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 1.1.1 [Tonhöhen setzen], Seite 1, [Notenbezeichnungen in anderen Sprachen], Seite 7, Abschnitt 2.10.2 [Arabische Musik], Seite 419, Abschnitt 2.10.3 [Türkische klassische Musik], Seite 425.

2.10.2 Arabische Musik

Dieser Abschnitt zeigt Möglichkeiten, wie arabische Musik notiert werden kann.

Referenz für arabische Musik

Arabische Musik wurde bisher vor allem mündlich tradiert. Wenn Musik transkribiert wird, handelt es sich meistens um ein Gerüst, auf dem der Musiker eigene Improvisationen ausführt. Mehr und mehr wird die westliche Notation mit einigen Veränderungen benutzt, um die arabische Musiktradition weiterzugeben und zu konservieren.

Einige Elemente der westlichen Notation wie etwa die Transkription von Akkorden oder eigenständige Stimmen werden für die traditionelleren arabischen Noten nicht benötigt. Es gibt allerdings einige andere Probleme, wie etwa die Notwendigkeit, Zwischenintervalle zu notieren, die sich irgendwo zwischen einem Halbton und einem Ganzton befinden. Daneben werden auch die westlichen Halb- und Ganztöne eingesetzt. Es muss auch möglich sein, eine große Anzahl an maqam (Modi) der arabischen Musik zu bezeichnen und zu gruppieren.

Üblicherweise müssen Mikrotöne in der arabischen Musik nicht präzise notiert werden.

Einige Bereiche, die für die arabische Notation wichtig sind, sind an anderer Stelle behandelt:

- Notenbezeichnungen und Versetzungszeichen (inklusive Vierteltöne) können angepasst werden, wie behandelt in Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.
- Zusätzliche Taktarten können erstellt werden, siehe [Tonartbezeichnung], Seite 20.
- Komplexe Taktarten erfordern evtl., dass Noten manuell gruppiert werden, wie gezeigt in [Manuelle Balken], Seite 84.
- *Takasim*, rhythmisch freie Improvisationen, können ohne Taktlinien notiert werden, siehe hierzu [Musik ohne Metrum], Seite 67.

Siehe auch

Notationsreferenz: Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418, [Tonartbezeichnung], Seite 20, [Manuelle Balken], Seite 84.
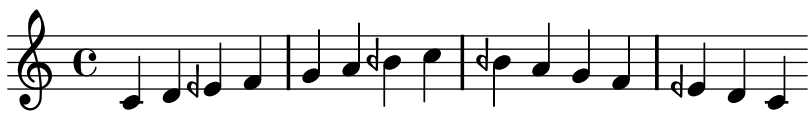
Schnipsel: Abschnitt “World music” in *Schnipsel*.

Arabische Notenbezeichnungen

An der arabischen Tradition orientierte Notenbezeichnungen können sehr land sein und eignen sich daher nicht gut für die Notation von Musik. Sie werden nicht benutzt. Englische Notenbezeichnungen hingegen sind in der arabischen Musikerziehung recht unbekannt, weshalb italienische Notenbezeichnungen (`do, re, mi, fa, sol, la, si`) eingesetzt werden. Modifikatoren (Versetzungszeichen) können auch benutzt werden. Italienische Notenbezeichnungen finden sich erklärt in [Notenbezeichnungen in anderen Sprachen], Seite 7, die Benutzung der normalen europäischen Notation für nichteuropäische Musik findet sich erklärt in Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.
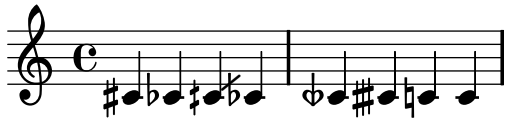
Hier ein Beispiel der arabischen *rast*-Tonleiter:

```
\include "arabic.ly"
\relative do' {
  do re misb fa sol la sisb do sisb la sol fa misb re do
}
```



Das Symbol für das Halb-B sieht anders aus als das Symbol, was üblicherweise in arabischer Notation benutzt wird. Das `\dwn`-Symbol, das in der Datei ‘`arabic.ly`’ definiert ist, kann als ein Workaround eingesetzt werden, wenn es notwendig ist, das arabische Symbol zu benutzen. Das Aussehen des Halb-Bs in den Vorzeichen kann mit dieser Methode nicht verändert werden.

```
\include "arabic.ly"
\relative do' {
  \set Staff.extraNatural = ##f
  dod dob dosd \dwn dob dobsb dodsd do do
}
```



Siehe auch

Notationsreferenz: [Notenbezeichnungen in anderen Sprachen], Seite 7, Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.

Schnipsel: Abschnitt “World music” in *Schnipsel*.

Arabische Tonarten

Neben den westlichen Dur- und Moll-Tonarten sind folgende Tonarten in ‘`arabic.ly`’ definiert: *bayati*, *rast*, *sikah*, *iraq* und *kurd*. Diese Tonarten definieren eine kleine Gruppe von Maqams, die weitverbreitet sind.

Ein Maqam kann die Tonart der Gruppe benutzen, zu der er gehört, oder die einer benachbarten Gruppe. Zusätzlich können verschiedene Versetzungszeichen in den Noten markiert werden.

Um also etwa die Tonart des Maqams „muhayer“ folgendermaßen notiert:

```
\key re \bayati
```

re ist die Tonhöhe für den „muhayer“-Maqam und *bayati* ist die Bezeichnung des Basismaqams der Gruppe.

Während die Vorzeichen eine Gruppe anzeigen, wird meistens der eigentliche Maqam im Titel definiert. In diesem Beispiel müsste also der „muhayer“-Maqam im Titel erscheinen.

Andere Maqams derselben Bayati-Gruppe, wie in der Tabelle unten gezeigt ((bayati, hussaini, saba und ushaq) können auf die gleiche Weise notiert werden. Sie sind alle Variationen des Grundmaqams Bayati. Sie unterscheiden sich üblicherweise vom grundlegenden Maqam in ihrem oberen Tetrachord oder in bestimmten Einzelheiten, die aber nicht ihre eigentliche Qualität verändern.

Der andere Maqam der gleichen Gruppe (Nawa) ist mit bayati durch eine Modulation verwandt, deren Grundton in der Tabelle angezeigt wird, wenn es sich um einen Maqam handelt, der eine Modulation eines anderen Maqams darstellt. Nawa kann folgenderweise notiert werden:

```
\key sol \bayati
```

In der arabischen Musik ist ein Begriff wie bayati, der eine Maqam-Gruppe bezeichnet, gleichzeitig auch selber ein Maqam, meistens der häufigste dieser Gruppe.

Hier ist eine Möglichkeit, Maqams zu gruppieren, womit die häufigsten Maqams bestimmten Vorzeichen zugeordnet werden:

Maqam-Gruppe	**Vorzeichen (`\key`)**	**Finalis**	**Andere Maqams der Gruppe (Finalis)**
ajam	major	sib	jaharka (fa)
bayati	bayati	re	hussaini, muhayer, saba, ushaq, nawa (sol)
hijaz	kurd	re	shahnaz, shad arban (sol), hijazkar (do)
iraq	iraq	sisb	-

kurd	kurd	re	hijazkar kurd (do)
nahawand	minor	do	busalik (re), farah faza (sol)
nakriz	minor	do	nawa athar, hisar (re)
rast	rast	do	mahur, yakah (sol)
sikah	sikah	misb	huzam

Ausgewählte Schnipsel

Untypische Tonarten
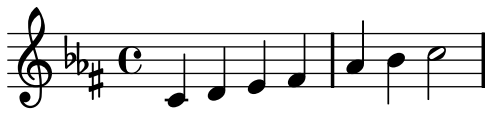
Der üblicherweise benutzte `\key`-Befehl setzt die `keySignature`-Eigenschaft im `Staff`-Kontext.

Um untypische Tonartenvorzeichen zu erstellen, muss man diese Eigenschaft direkt setzen. Das Format für den Befehl ist eine Liste: `\set Staff.keySignature = #`(((Oktave . Schritt) . Alteration) ((Oktave . Schritt) . Alteration) ...)` wobei für jedes Element in der Liste `Oktave` die Oktave angibt (0 ist die Oktave vom eingestrichenen C bis zum eingestrichenen H), `Schritt` gibt die Note innerhalb der Oktave an (0 heißt C und 6 heißt H), und `Alteration` ist `,SHARP ,FLAT ,DOUBLE-SHARP` usw. (Beachte das beginnende Komma.)

Alternativ kann auch jedes Element der Liste mit dem allgemeineren Format `(Schritt . Alteration)` gesetzt werden, wobei dann die Einstellungen für alle Oktaven gelten.

Hier ein Beispiel einer möglichen Tonart für eine Ganztonleiter:

```
\relative c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
                              ((0 . 5) . ,FLAT)
                              ((0 . 3) . ,SHARP))
  c4 d e fis
  aes4 bes c2
}
```



Siehe auch

Glossar: Abschnitt “maqam” in *Glossar*, Abschnitt “bayati” in *Glossar*, Abschnitt “rast” in *Glossar*, Abschnitt “sikah” in *Glossar*, Abschnitt “iraq” in *Glossar*, Abschnitt “kurd” in *Glossar*.

Notationsreferenz: [Tonartbezeichnung], Seite 20.

Handbuch zum Lernen: Abschnitt “Versetzungszeichen und Tonartbezeichnung (Vorzeichen)” in *Handbuch zum Lernen*.

Referenz der Interna: Abschnitt “KeySignature” in *Referenz der Interna*.

Schnipsel: Abschnitt “World music” in *Schnipsel*, Abschnitt “Pitches” in *Schnipsel*.

Arabische Taktarten

Einige klassische Formen der arabischen und türkischen Musik wie etwa *Semai* haben ungewöhnliche Taktarten wie etwa 10/8. Das kann dazu führen, dass die automatische Bebalkung der Noten nicht zu dem Ergebnis kommt, welches in der üblichen Notation dieser Musik eingesetzt wird. Die Noten werden nicht anhand einer Taktzeit, sondern anhand von Kriterien gruppiert, die man schwer mit einer automatischen Balkenfunktion erfassen kann. Das kann umgangen werden, indem die automatische Bebalkung ausgeschaltet wird und die Balken explizit gesetzt werden. Auch wenn es nicht darauf ankommen sollte, eine schon notierte

Musik nachzuahmen, ist es in vielen Fällen dennoch erforderlich, die Bebalkung anzupassen und/oder zusammengesetzte Taktarten zu benutzen.
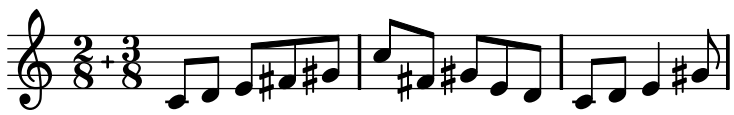
Ausgewählte Schnipsel

Zusammengesetzte Taktarten

Ungerade Taktarten werden (wie etwa "5/8") werden oft als zusammengesetzte Taktarten interpretiert (bspw. "3/8 + 2/8"), in welchen zwei oder mehr Teiltakte unterschieden werden. LilyPond kann derartige Noten produzieren, indem entsprechende Taktarten gesetzt werden und die automatische Bebalkung angepasst wird.

```
#(define ((compound-time one two num) grob)
   (grob-interpret-markup grob
                          (markup #:override '(baseline-skip . 0) #:number
                                  (#:line ((#:column (one num))
                                           #:vcenter "+"
                                           (#:column (two num)))))))

\relative c' {
  \override Staff.TimeSignature #'stencil = #(compound-time "2" "3" "8")
  \time 5/8
  \set Staff.beatStructure = #'(2 3)
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```

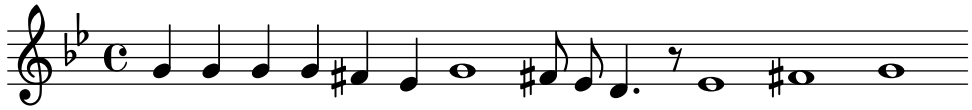


Arabische Improvisation

Bei Improvisation oder *taqasim*, die zeitlich frei gespielt werden, kann die Taktart ausgelassen werden und `\cadenzaOn` kann eingesetzt werden. Es kann nötig sein, den Versetzungszeichenstil anzupassen, weil sonst die Versetzungszeichen nur einmal ausgegeben werden, da keine Taktlinien gesetzt sind. Hier ein Beispiel, wie der Beginn einer *hijaz*-Improvisation aussehen könnte:

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  #(set-accidental-style 'forget)
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



Siehe auch

Glossar: Abschnitt “semai” in *Glossar*, Abschnitt “taqasim” in *Glossar*.

Notationsreferenz: [Manuelle Balken], Seite 84, [Automatische Balken], Seite 74, [Musik ohne Metrum], Seite 67, [Automatische Versetzungszeichen], Seite 24, [Einstellung von automatischen Balken], Seite 77, [Taktangabe], Seite 58.
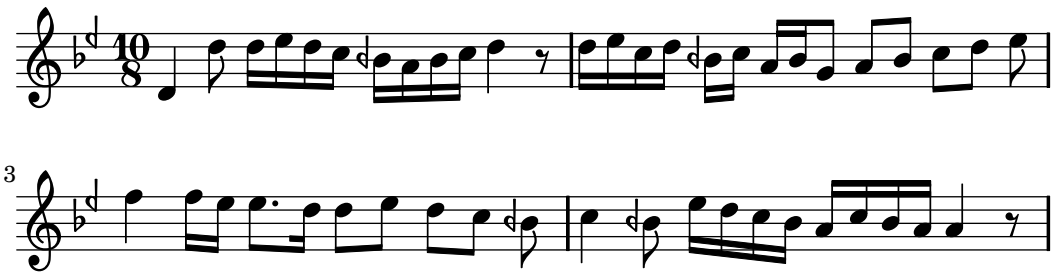
Schnipsel: Abschnitt “World music” in *Schnipsel*.

Arabische Notenbeispiele

Hier eine Vorlage, welche den Beginn eines türkischen *Semai* benutzt, der in der arabischen Musikerziehung oft herangezogen wird, um Besonderheiten der arabischen Musiknotation, wie etwa Zwischenintervalle und ungewöhnliche Modi, zu illustrieren.

```
\include "arabic.ly"
\score {
  \relative re' {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
    re16 [misb do re] sisb [do] la [sisb sol8] la [sisb] do [re] misb
    fa4 fa16 [misb] misb8. [re16] re8 [misb] re  [do] sisb
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
  }
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
}
```



Siehe auch

Schnipsel: Abschnitt “World music” in *Schnipsel*.

Weitere Literatur zur arabischen Musik

1. *The Music of the Arabs* von Habib Hassan Touma (Amadeus Press, 1996) enthält eine Beschreibung von Maqams und Methoden zu ihrer Gruppierung.

   Es gibt auch einige Internetseiten, die Maqams erklären und teilweise auch Klangdateien zur Verfügung stellen:

   - http://www.maqamworld.com/
   - http://www.turath.org/

   Die Maqam-Gruppierungen unterscheiden sich in einigen Details, auch wenn die allgemeinen Kriterien weithin anerkannt sind: gemeinsame untere Tetrachorde sowie Modulation.

2. Es gibt keine Übereinstimmung darüber, wie die Vorzeichen für bestimmte Maqams angegeben werden sollen. Oft wird eine Vorzeichenart für eine ganze Maqam-Gruppe verwendet, anstatt dass jeder Maqam eigene Vorzeichen hätte.

   Lehrbücher für *Oud*, die arabische Laute, folgender Autoren enthalten Beispiele vor allem türkischer und arabischer Kompositionen:

   - Charbel Rouhana
   - George Farah
   - Ibrahim Ali Darwish Al-masri

2.10.3 Türkische klassische Musik

Dieser Abschnitt zeigt Probleme, die bei der Notation von klassischer türkischer Musik auftreten können.

Verweise für türkische klassische Musik

Türkische klassische Musik wurde im Osmanischen Reich während einer Periode entwickelt, die ungefähr zur gleichen Zeit der westlichen klassischen Musik stattfand. Diese lebendige und starke Tradition wird bis heute mit ihren eigenen kompositorischen Formen, Musiktheorie und Aufführungsstilen weitergeführt. Unter den Eigenheiten dieser Tradition befinden sich die Benutzung von Mikrointervallen basierend auf „Kommas“ von 1/9-Tönen, aus denen melodische Formen konstruiert werden, welche man als *makam* (Pl. *makamlar*) bezeichnet.

Einige Probleme der Notation türkischer klassischer Musik sind woanders behandelt:

- Notenbezeichnungen und Versetzungszeichen finden sich in Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.

Türkische Notenbezeichnungen

Tonhöhen der türkischen klassischen Musik haben traditionell einmalige Bezeichnungen, und weil die Tonhöhen auf 1/9-Tönen basieren, unterscheiden sich die Intervalle von makamlar deutlich von den Intervallen westlicher klassischer Musik: *koma* (1/9 eines Ganztons), *eksik bakiye* (3/9), *bakiye* (4/9), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanîni* (ein Ganzton) und *artık ikili* (12/9 oder 13/9 eines Ganztons).

Es bietet sich an, die normalen westlichen Noten auf dem Notensystem zu benutzen (also c, d, e . . .) angrereichert mit besonderen Versetzungszeichen, die die Noten um 1/9, 4/9, 5/9 und 8/9 eines Ganztons erhöhen oder erniedrigen. Diese Versetzungszeichen sind definiert in der Datei ‘`makam.ly`’.

Die folgende Tabelle zeigt

- die Bezeichnung dieser besonderen Versetzungszeichen
- die Endung, die an die Note gehängt werden muss und
- die Tonhöhenveränderung als Bruch eines Ganztones.

Versetzungszeichen	**Endung**	**Tonhöhenveränderung**
büyük mücenneb (Kreuz)	-bm	+8/9
kücük mücenneb (Kreuz)	-k	+5/9
bakiye (Kreuz)	-b	+4/9
koma (Kreuz)	-c	+1/9
koma (B)	-fc	-1/9
bakiye (B)	-fb	-4/9
kücük mücenneb (B)	-fk	-5/9

büyük mücenneb (B) -fbm -8/9

Eine weitergehende Erklärung der Notation nichteuropäischer Musik findet sich in Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.

Siehe auch

Glossar: Abschnitt “makam” in *Glossar*, Abschnitt “makamlar” in *Glossar*.

Notationsreferenz: Abschnitt 2.10.1 [Übliche Notation für nichteuropäische Musik], Seite 418.

3 Allgemeine Eingabe und Ausgabe

Dieses Kapitel erklärt allgemeine Fragen zur Eingabe und Ausgabe von Notation mit LilyPond und weniger direkte Fragen der Notation.

3.1 Eingabestruktur

Das hauptsächliche Eingabeformat von LilyPond sind Textdateien. Üblicherweise werden diese Dateien mit der Endung '`.ly`' versehen.

3.1.1 Struktur einer Partitur

Eine `\score`-Umgebung muss einen einzelnen musikalischen Ausdruck beinhalten, der durch geschweifte Klammern begrenzt wird:
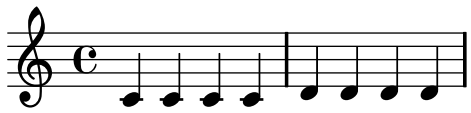
```
\score {
...
}
```

Achtung: Es darf **nur ein** äußerer musikalischer Ausdruck in der `\score`-Umgebung geschrieben werden, und er **muss** von geschweiften Klammern umgeben sein.

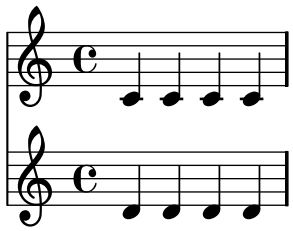
Dieser einzelne musikalische Ausdruck kann beliebige Größe annehmen und andere musikalische Ausdrücke von beliebiger Komplexität beinhalten. Alle diese Beispiele sind musikalische Ausdrücke:

```
{ c'4 c' c' c' }
```

```
{
  { c'4 c' c' c' }
  { d'4 d' d' d' }
}
```



```
<<
  \new Staff { c'4 c' c' c' }
  \new Staff { d'4 d' d' d' }
>>
```



```
{
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { \Flöte }
      \new Staff { \Oboe }
    >>
    \new StaffGroup <<
```

```
      \new Staff { \GeigeI }
      \new Staff { \GeigeII }
    >>
  >>
}
```

Kommentare bilden eine Ausnahme dieser Regel. (Andere Ausnahmen siehe Abschnitt 3.1.5 [Die Dateistruktur], Seite 431.) Sowohl einzeilige als auch Blockkommentare (eingegrenzt durch `%{ .. %}`) können an beliebiger Stelle einer Eingabedatei geschrieben werden. Sie können innerhalb oder außerhalb der `\score`-Umgebung vorkommen, und innerhalb oder außerhalb des einzelnen musikalischen Ausdrucks innerhalb der `\score`-Umgebung.

Denken Sie daran, dass auch eine Datei, die nur eine `\score`-Umgebung enhält, implizit in eine `\book`-Umgebung eingeschlossen wird. Eine `\book`-Umgebung in einer Eingabdatei produziert wenigstens eine Ausgabedatei, und standardmäßig wird der Name der Ausagabedatei aus dem Namen der Eingabedatei abgeleitet. ‘`fandangoforelephants.ly`’ produziert also ‘`fandangoforelephants.pdf`’.

Zu weiteren Einzelheiten zu `\book`-Umgebungen siehe Abschnitt 3.1.2 [Mehrere Partituren in einem Buch], Seite 428, Abschnitt 3.1.3 [Mehrere Ausgabedateien aus einer Eingabedatei], Seite 429 und ⟨undefined⟩ [Dateistruktur], Seite ⟨undefined⟩.

Siehe auch

Handbuch zum Lernen: Abschnitt “Arbeiten an Eingabe-Dateien” in *Handbuch zum Lernen*, Abschnitt “Musikalische Ausdrücke erklärt” in *Handbuch zum Lernen*, Abschnitt “Score ist ein (einziger) zusammengesetzter musikalischer Ausdruck” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 3.1.2 [Mehrere Partituren in einem Buch], Seite 428, Abschnitt 3.1.3 [Mehrere Ausgabedateien aus einer Eingabedatei], Seite 429 und ⟨undefined⟩ [Dateistruktur], Seite ⟨undefined⟩.

3.1.2 Mehrere Partituren in einem Buch

Eine Partitur kann mehrere musikalische Stücke und verschiedene Texte beinhalten. Beispiele hierzu sind etwa eine Etüdensammlung oder ein Orchesterstück mit mehreren Sätzen. Jeder Satz wird in einer eigenen `\score`-Umgebung notiert:

```
\score {
  ..Noten..
}
```

und Texte werden mit einer `\markup`-Umgebung geschrieben:

```
\markup {
  ..Text..
}
```

Alle Sätze und Texte, die in derselben ‘`.ly`’-Datei vorkommen, werden normalerweise in eine einzige Ausgabedatei gesetzt.

```
\score {
  ..
}
\markup {
  ..
}
\score {
  ..
}
```

Eine wichtige Ausnahme stellen Dokumente dar, die mit lilypond-book erstellt werden, für die Sie explizit `\book`-Umgebungen notieren müssen, weil sonst nur die erste `\score`- bzw. `\markup`-Umgebung angezeigt wird.

Der Kopfbereich für jedes Musikstück kann innerhalb der `\score`-Umgebung definiert werden. Die `piece`-(Stück)-Bezeichnung aus dieser `\header`-Umgebung wird vor jedem Satz ausgegeben. Die Überschrift für ein ganzes Buch kann innerhalb von `\book` notiert werden, aber wenn diese Umgebung fehlt, wird die `\header`-Umgebung genommen, die auf erster Ebene der Datei notiert ist.

```
\header {
  title = "Acht Miniaturen"
  composer = "Igor Stravinsky"
}
\score {
  ...
  \header { piece = "Romanze" }
}
\markup {
   ..Text der zweiten Strophe..
}
\markup {
   ..Text der dritten Strophe..
}
\score {
  ...
  \header { piece = "Menuetto" }
}
```

Stücke können innerhalb eines Buches mit `\bookpart` gruppiert werden. Derartige Buchabschnitte werden durch einen Seitenumbruch voneinander getrennt und können wie auch das ganze Buch selber mit einem Titel innerhalb einer `\header`-Umgebung beginnen.

```
\bookpart {
  \header {
    title = "Buchtitel"
    subtitle = "Erster Teil"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Zweiter Teil"
  }
  \score { ... }
  ...
}
```

3.1.3 Mehrere Ausgabedateien aus einer Eingabedatei

Wenn Sie mehrere Ausgabedateien aus derselben '`.ly`'-Datei haben wollen, können Sie mehrere `\book`-Umgebungen hinzufügen, wobei jede Umgebung eine neue Ausgabedatei produziert. Wenn Sie keine `\book`-Umgebung in der Eingabedatei angeben, wird die Datei von LilyPond implizit als eine große `\book`-Umgebung behandelt, siehe auch ⟨undefined⟩ [Dateistruktur], Seite ⟨undefined⟩.

Wenn man mehrere Dateien aus einer einzigen Eingabedatei erstellt, stellt LilyPond sicher, dass keine der Ausgabedateien der vorhandenen `\book`-Umgebungen eine andere Ausgabedatei, etwa von der vorherigen `\book`-Umgebung, überschreibt.

Dies geschieht, indem ein Suffix an den Ausgabenamen für jede `\book`-Umgebung gehängt wird, die den Dateinamen der Eingabdatei als Grundlage nimmt.

Das Standardverhalten ist es, einen Zahlen-Suffix für die Namen hinzuzufügen, die in Konflikt stehen. Der Code

```
\book {
  \score { ... }
  \layout { ... }
}
\book {
  \score { ... }
  \layout { ... }
}
\book {
  \score { ... }
  \layout { ... }
}
```

produziert also

- ‘`eightminiatures.pdf`’,
- ‘`eightminiatures-1.pdf`’ and
- ‘`eightminiatures-2.pdf`’.

3.1.4 Dateinamen der Ausgabedateien

LilyPond stellt die Möglichkeit zur Verfügung zu kontrollieren, welche Dateinamen für welche Back-ends benutzt werden sollen, wenn die Ausgabedateien erstellt werden.

Im vorhergehenden Abschnitt wurde gezeigt, wir LilyPond gleichnamige Ausgabedateien verhindert, wenn mehrere Ausgabedateien aus derselben Eingabedatei erstellt werden. Es gibt auch die Möglichkeit, eigene Suffixe für jeden `\book`-Abschnitt zu definieren, sodass man etwa Dateinamen wie ‘`eightminiatures-Romanze.pdf`’, ‘`eightminiatures-Menuetto.pdf`’ und ‘`eightminiatures-Nocturne.pdf`’ produzieren kann, indem man eine `\bookOutputSuffix`-Angabe in jede `\book`-Umgebung einfügt.

```
\book {
  \bookOutputSuffix "Romanze"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputSuffix "Menuetto"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputSuffix "Nocturne"
  \score { ... }
  \layout { ... }
}
```

Man kann auch einen anderen Dateinamen für die Ausgabedatei einer `\book`-Umgebung erstellen, indem man `\bookOutputName`-Angabe macht:

```
\book {
  \bookOutputName "Romanze"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputName "Menuetto"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputName "Nocturne"
  \score { ... }
  \layout { ... }
}
```

Die obige Datei produziert folgene Ausgabedateien:

- ‘`Romanze.pdf`’,
- ‘`Menuetto.pdf`’ and
- ‘`Nocturne.pdf`’.

3.1.5 Die Dateistruktur

Eine ‘`.ly`’-Datei kann eine beliebige Anzahl an Ausdrücken auf der obersten Ebene beinhalten, wobei ein Ausdruck der obersten Ebene einer der folgenden sein kann:

- Eine Ausgabedefinition, wie `\paper`, `\midi` und `\layout`. Derartige Definitionen auf oberster Ebene verändern die globalen Einstellungen für das ganze „Buch“. Wenn mehr als eine derartige Definition desselben Typs angegeben wird, hat die spätere Vorrang.
- Ein direkter Scheme-Ausdruck, wie etwa `#(set-default-paper-size "a7" 'landscape)` oder `#(ly:set-option 'point-and-click #f)`.
- Eine `\header`-Umgebung. Damit wird die globale Titelei eingestellt. Das ist die Umgebung, in der sich Definition für das ganze Buch befinden, wie Komponist, Titel usw.
- Eine `\score`-Umgebung. Die in ihr enthaltene Partitur wird zusammen mit anderen vorkommenden `\score`-Umgebungen gesammelt und in ein `\book` zusammengefasst. Dieses Verhalten kann verändert werden, indem die Variable `toplevel-score-handler` auf höchster Ebene gesetzt wird. Die Definition des Standards findet sich in der Datei ‘`../scm/lily.scm`’.
- Eine `\book`-Umgebung fasst mehrere Sätze (d. h. mehrere `\score`-Umgebungen) logisch in ein Dokument zusammen. Wenn mehrere `\score`-Partituren vorkommen, wird für jede `\book`-Umgebung eine eigene Ausgabedatei erstellt, in der alle in der Umgebung enthaltenen Partituren zusammengefasst sind. Der einzige Grund, explizit eine `\book`-Umgebung zu setzen, ist, wenn mehrere Ausgabedateien aus einer einzigen Quelldatei erstellt werden sollen. Eine Ausnahme sind lilypond-book-Dokumente, in denen eine `\book`-Umgebung explizit hinzugefügt werden muss, wenn mehr als eine `\score`- oder `\markup`-Umgebung im gleichen Beispiel angezeigt werden soll. Dieses Verhalten kann verändert werden, indem die Variable `toplevel-book-handler` auf höchster Ebene gesetzt wird. Die Definition des Standards findet sich in der Datei ‘`../scm/lily.scm`’.
- Eine `\bookpart`-Umgebung. Ein Buch (`\book`) kann in mehrere Teile untergliedert sein, indem `\bookpart`-Umgebungen eingesetzt werden. Jeder Buchabschnitt beginnt auf einer neuen Seite und kann eigene Papierdefinitionen in einer `\paper`-Umgebung haben.
- Ein zusammengesetzter musikalischer Ausdruck wie etwa

```
{ c'4 d' e'2 }
```

Dieses Beispiel wird von LilyPond automatisch in einer `\score`-Umgebung in einem Buch interpretiert und mit anderen `\score`-Umgebungen und musikalischen Ausdrücken auf der höchsten Ebene zusammen ausgegeben. Anders gesagt: eine Datei, die nur das obige Beispiel beinhaltet, wird übersetzt zu

```
\book {
  \score {
    \new Staff {
      \new Voice {
        { c'4 d' e'2 }
      }
    }
  }
\layout { }
\header { }
}
```

Dieses Verhalten kann verändert werden, indem die Variable `toplevel-music-handler` auf der obersten Ebene gesetzt wird. Die Definition des Standards findet sich in der Datei '`../scm/lily.scm`'.

- Eine Textbeschriftung, eine Strophe etwa:

  ```
  \markup {
     2.  Die erste Zeile der zweiten Strophe.
  }
  ```

  Textbeschriftungen werden über, zwischen oder unter musikalischen Ausdrücken gesetzt, so wie sie notiert werde.

- Eine Variable, wie

  ```
  foo = { c4 d e d }
  ```

  Sie kann dann später in der Datei eingesetzt werden, indem `\foo` geschrieben wird. Die Bezeichnung der Variable darf nur aus alphabetischen Zeichen bestehen, keine Zahlen, Unter- oder Bindestriche.

Das folgende Beispiel zeigt drei Dinge, die auf der obersten Ebene notiert werden können:

```
\layout {
  % Zeilen rechtsbündig setzen
  ragged-right = ##t
}

\header {
   title = "Do-re-mi"
}

{ c'4 d' e2 }
```

An einer beliebigen Stelle der Datei kann jede der folgenden lexikalen Anweisungen notiert werden:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- Ein einzeiliger Kommentar, beginnend mit `%`.

- Ein mehrzeiliger Kommentar, umgeben von `%{ .. %}`.

Leerzeichen zwischen Einheiten in der Eingabe werden generell ignoriert und können nach Belieben weggelassen werden oder hinzugefügt werden, um die Lesbarkeit des Codes zu verbessern. Mindestens ein Leerzeichen sollte jedoch unter folgenden Umständen immer eingesetzt werden, um Fehler zu vermeiden:

- Vor und hinter jeder schließenden oder öffnenden Klammer,
- nach jedem Befehl oder jeder Variable, also jeder Einheit, die mit `\` beginnt,
- nach jeder Einheit, die als Scheme-Ausdruck interpretiert werden, also alle Einheiten, die mit `#` beginnen.
- Alle Einheiten von Scheme-Ausdrücken müssen mit Leerzeichen getrennt werden,
- in Gesangstextabschnitten (`lyricmode`) müssen Leerzeichen zwischen alle Ausdrücke in `\override`- und `\set`-Befehlen gesetzt werden. Insbesondere müssen um Punkte und Gleichzeitszeichen in Befehlen wie `\override Score . LyricTex #'font-size = #5)` und vor dem gesamten Befehl geschrieben werden.

Siehe auch

Handbuch zum Lernen: Abschnitt “Wie eine LilyPond-Eingabe-Datei funktioniert” in *Handbuch zum Lernen*.

3.2 Titel

Fast alle gedruckten Noten beinhalten einen Titel und den Namen des Komponisten, teilweise wird auch noch sehr viel mehr Information zur Verfügung gestellt.

3.2.1 Titel erstellen

Überschriften können für jede `\score`-Umgebung erstellt werden, sowohl für die gesamte Datei (oder eine `\book`-Umgebung) als auch für einzelne Buchabschnitte (innerhalb einer `\bookpart`-Umgebung).

Der Inhalt der Titelei wird aus der `\header`-Umgebung übernommen. Die `\header`-Umgebung eines Buches unterstützt folgende Felder:

`dedication`
: Die Widmung der Noten, wird auf oben auf der ersten Seite gesetzt.

`title`
: Die Überschrift der Noten, wird unter der Widmung zentriert gesetzt.

`subtitle`
: Untertitel, zentriert unter der Überschrift.

`subsubtitle`
: Unteruntertitel, zentriert unter dem Untertitel.

`poet`
: Name des Dichters, linksbündig unter dem Unteruntertitel.

`instrument`
: Bezeichnung des Instruments, zentriert unter dem Unteruntertitel. Auch oben auf der Seite zentriert (andere als erste Seite).

`composer`
: Name des Komponisten, rechtsbündig unter dem Unteruntertitel.

`meter`
: Metrum, linksbündig unter dem Dichter.

`arranger`
: Name des Bearbeiters/Arrangeurs, rechtsbündig unter dem Komponisten.

`piece`
: Bezeichnung des Stückes, linksbündig unter dem Metrum.

`opus`
: Bezeichnung des Opus, rechtsbündig unter dem Bearbeiter.

`breakbefore`
: Hiermit beginnt der Titel auf einer neuen Seite. (Kann die Werte ##t (wahr) oder ##f (falsch) haben.)

`copyright`
: Anzeige eines Copyright, zentriert unten auf der ersten Seite. Um das Copyright-Symbol zu notieren, siehe Abschnitt 3.3.3 [Zeichenkodierung], Seite 447.

`tagline`
: Zentriert unten auf der letzten Seite. Enthält standardmäßig: „Music engraving by LilyPond (*version*)—www.lilypond.org“

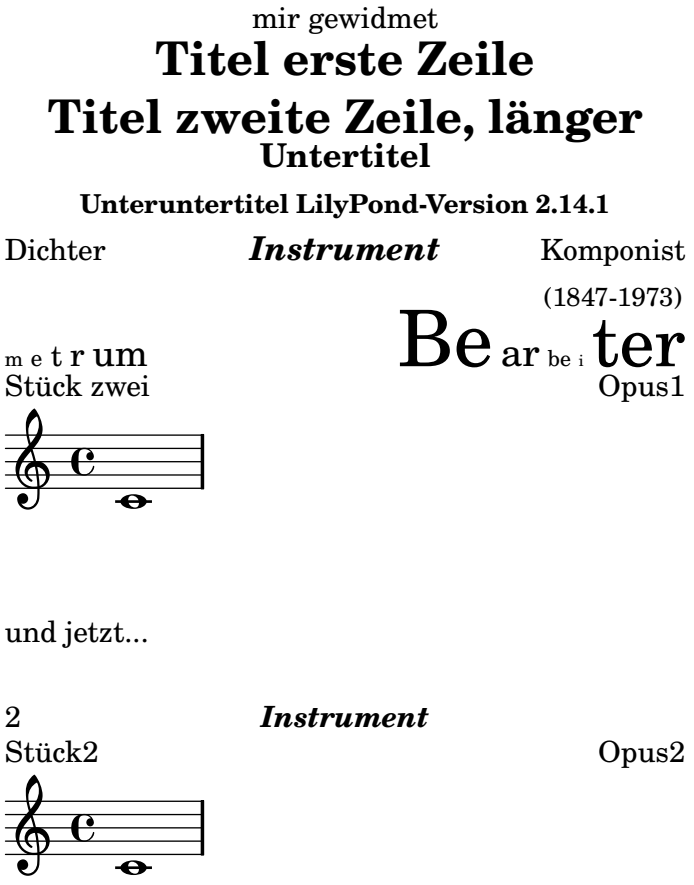
Hier eine Demonstration der möglichen Felder. Beliebige Formatierungsbefehle für Textbeschriftung können in der Titelei eingesetzt werden. Siehe hierzu auch Abschnitt 1.8.2 [Text formatieren], Seite 210.

```
\paper {
  line-width = 9.0\cm
  paper-height = 10.0\cm
}

\book {
  \header {
    dedication = "mir gewidmet"
    title = \markup \center-column { "Titel erste Zeile" "Titel zweite Zeile, länger" }
    subtitle = "Untertitel"
    subsubtitle = #(string-append "Unteruntertitel LilyPond-Version "
(lilypond-version))
    poet = "Dichter"
    composer =  \markup \center-column { "Komponist" \small "(1847-1973)" }
    texttranslator = "Übersetzer"
    meter = \markup { \teeny "m" \tiny "e" \normalsize "t" \large "r" \huge
"um" }
    arranger = \markup { \fontsize #8.5 "Be" \fontsize #2.5 "ar" \fontsize
#-2.5 "be" \fontsize #-5.3 "i" \fontsize #7.5 "ter" }
    instrument = \markup \bold \italic "Instrument"
    piece = "Stück"
  }

  \score {
    { c'1 }
    \header {
      piece = "Stück zwei"
      opus = "Opus1"
    }
  }
  \markup {
      und jetzt...
  }
  \score {
    { c'1 }
    \header {
      piece = "Stück2"
      opus = "Opus2"
    }
```

```
  }
}
```



Wie schon oben gezeigt, können mehrfache `\header`-Umgebungen eingesetzt werden. Wenn das gleiche Feld in mehreren Umgebungen, wird die letzte vorkommende Version benutzt. Hier ein kurzes Beispiel:
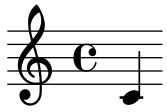
```
\header {
  composer = "Komponist"
}
\header {
  piece = "Stück"
```

```
}
\score {
  \new Staff { c'4 }
  \header {
    piece = "Neues Stück"  % überschreibt die die vorige Definition
  }
}
```

Wenn `\header` innerhalb der `\score`-Umgebung definiert wird, wird normalerweise nur die Information von `piece` und `opus` ausgegeben. Musikalische Ausdrücke innerhalb von `\score` müssen vor `\header` gesetzt werden.

```
\score {
  { c'4 }
  \header {
    title = "title"  % not printed
    piece = "piece"
    opus = "opus"
  }
}
```

piece opus



Dieses Verhalten kann verändert werden (sodass alle Angaben aus der Überschrift gesetzt werden, wenn sich `\header` innerhalb von `\score` befindet), indem man schreibt:

```
\paper{
  print-all-headers = ##t
}
```

Die Standardfußzeile ist leer mit Ausnahme der ersten Seite, auf der das `copyright`-Feld aus der `\header`-Umgebung eingefügt wird, und die letzte Seite, auf der das `tagline`-Feld eingefügt wird. Der Standardinhalt von `tagline` ist „Music engraving by LilyPond (*version*)—www.lilypond.org“. Gut gesetzte Noten werben sehr effektiv für LilyPond, darum bitten wir darum, diese Zeile stehen zu lassen, wenn es möglich ist.

Ein Titelfeld kann vollständig entfernt werden, indem es auf falsch gesetzt wird:

```
\header {
  tagline = ##f
  composer = ##f
}
```

3.2.2 Eigene Kopf- und Fußzeilen sowie Titel

Kompliziertere Anpassungen können vorgenommen werden, indem die folgenden Variablen innerhalb der `\paper`-Umgebung geändert werden. Die Init-Datei ‘`../ly/titling-init.ly`’ enthält das Standardverhalten.

`bookTitleMarkup`
: Das ist die Überschrift, die für das gesamte Dokument gilt. Üblicherweise wird hier der Komponist und die Überschrift des Werkes genannt.

`scoreTitleMarkup`
: Das ist die Überschrift, die vor jede `\score`-Umgebung gesetzt wird. Üblicherweise wird hier etwa die Bezeichnung eines Satzes notiert (im `piece`-Feld).

`oddHeaderMarkup`
: Das ist der Seitenkopf für ungerade Seiten.

`evenHeaderMarkup`
: Das ist der Seitenkopf für gerade Seiten. Wenn undefiniert, wird der ungerade Seitenkopf eingesetzt.

  Standardmäßig werden die Kopfzeilen so definiert, dass die Seitennummer sich außen befindet und das Instrument zentriert gesetzt wird.

`oddFooterMarkup`
: Das ist die Fußzeile für ungerade Seiten.

`evenFooterMarkup`
: Das ist die Fußzeile für gerade Seiten. Wenn undefiniert, wird die ungerade Fußzeile eingesetzt.

  Standardmäßig wird in der Fußzeile auf der ersten Seite das Copyright und auf der letzten Seite die Tag-Zeile gesetzt.

Die folgende Definition setzt die Überschrift linksbündig und den Komponisten rechtsbündig auf einer einzelnen Zeile:

```
\paper {
  bookTitleMarkup = \markup {
   \fill-line {
     \fromproperty #'header:title
     \fromproperty #'header:composer
   }
  }
}
```

Kopf- und Fußzeile werden mit den Funktionen `make-header` und `make-footer` erstellt, welche in `\paper` definiert werden. Die Standardimplementationen finden sich in ‘`ly/paper-defaults-init.ly`’ und ‘`ly/titling-init.ly`’.

Dieses Beispiel zentriert die Seitenzahlen unten auf jeder Seite:

```
\paper {
  print-page-number = ##t
  print-first-page-number = ##t
  oddHeaderMarkup = \markup \fill-line { " " }
  evenHeaderMarkup = \markup \fill-line { " " }
  oddFooterMarkup = \markup {
    \fill-line {
      \bold \fontsize #3
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
    }
  }
  evenFooterMarkup = \markup {
    \fill-line {
      \bold \fontsize #3
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
```

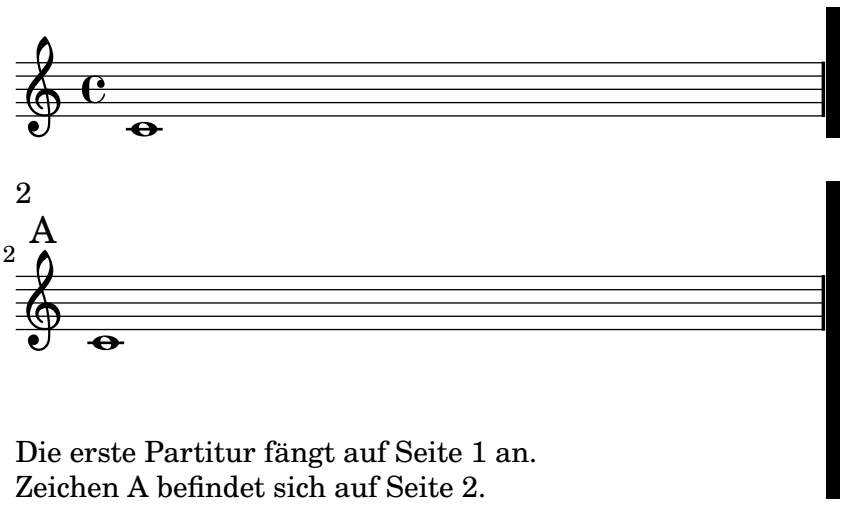
```
    }
  }
}
```

3.2.3 Verweis auf die Seitenzahlen

Eine bestimmte Stelle der Partitur kann mit einem `\label`-Befehl markiert werden, sowohl auf oberster Ebene als auch innerhalb eines musikalischen Ausdrucks. Auf diese Marke kann dann verwiesen werden, um die Seitenzahl zu erhalten, auf der die Marke vorkommt. Der Verweis wird mit dem Befehl `\page-ref` gefordert (innerhalb von `\markup`).

```
\header { tagline = ##f }
\book {
  \label #'ErstePartitur
  \score {
    {
      c'1
      \pageBreak \mark A \label #'ZeichenA
      c'1
    }
  }

  \markup { Die erste Partitur fängt auf
            Seite \page-ref #'ErstePartitur "0" "?" an.}
  \markup { Zeichen A befindet sich auf Seite
            \concat { \page-ref #'ZeichenA "0" "?" . } }
}
```



Die erste Partitur fängt auf Seite 1 an.
Zeichen A befindet sich auf Seite 2.

Der `\page-ref`-Textbeschriftungsbefehl braucht drei Argumente:

1. die Marke, ein Scheme-Symbol, etwa `#'ErstePartitur`,
2. eine Beschriftung, die als Platzhalter benutzt wird, um die Breite des Verweisen zu schätzen,
3. eine Beschriftung, die anstelle der Seitenzahl gesetzt wird, wenn die Marke unbekannt ist.

Der Grund, warum ein Platzhalter benötigt wird, ist dass zu dem Zeitpunkt, an dem die Textbeschriftungen ausgewertet werden, noch keine Seitenumbrüche vorgenommen wurden und die Seitenzahlen deshalb noch nicht bekannt sind. Um hier ein Problem zu vermeiden, wird die eigentliche Auswertung der Textbeschriftung erst später ausgeführt, die Größe des Textes muss aber schon vorher bekannt sein. Die Größe wird mithilfe des Platzhalters bestimmt. Wenn eine Partitur zwischen 10 und 99 Seiten hat, kann man `"00"` schreiben, also eine zweistellige Zahl.

`\label \page-ref`

Vordefinierte Befehle

3.2.4 Inhaltsverzeichnis

Ein Inhaltsverzeichnis kann eingefügt werden mit dem Befehl `\markuplines \table-of-contents`. Die Elemente, die im Inhaltsverzeichnis aufgelistet werden sollen, werden mit dem `\tocItem`-Befehl markiert, welches sowohl auf höchster Ebene als auch in einem musikalischen Ausdruck verwendet werden kann.

```
\markuplines \table-of-contents
\pageBreak

\tocItem \markup "Erste Partitur"
\score {
  {
    c'4  % ...
    \tocItem \markup "Ein bestimmter Punkt innerhalb der ersten Partitur"
    d'4  % ...
  }
}

\tocItem \markup "zweite Partitur"
\score {
  {
    e'4 % ...
  }
}
```

Die Beschriftungen, die benutzt werden um das Inhaltsverzeichnis zu formatieren, sind in der `\paper`-Umgebung definiert. Die Standardformatierungselemente sind `tocTitleMarkup` um die Überschrift zu formatieren und `tocItemMarkup` um die einzelnen Inhaltselemente zu formatieren, bestehend aus dem Titelelement und einer Seitenzahl. Die Variablen können durch den Benutzer geändert werden:

```
\paper {
  %% Übersetzung der Inhaltsverzeichnisüberschrift nach französisch:
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
  %% hier größere Schriftarten
  tocItemMarkup = \markup \large \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
}
```

Die Inhaltsverzeichniselemente Text und Seitenzahl werden in der Definition von `tocItemMarkup` aufgerufen mit `#'toc:text` und `#'toc:page`.

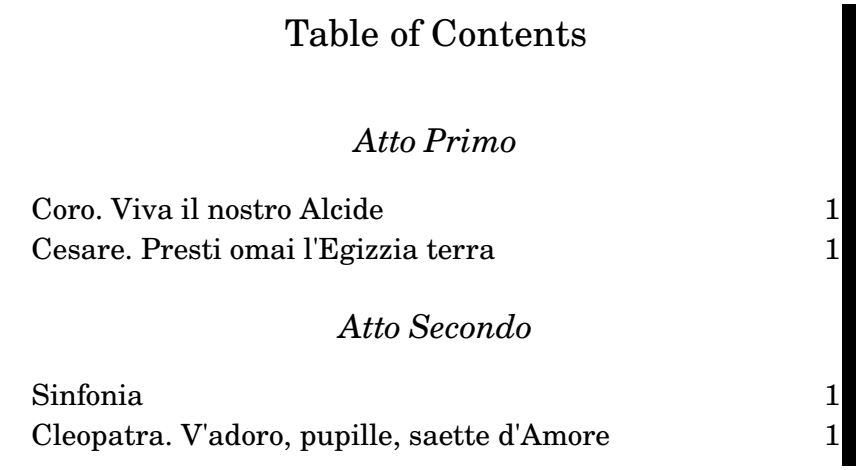
Neue Befehle und Beschriftungen können auch definiert werden, um eigene Inhaltsverzeichnisse zu gestalten:

- zuerst muss eine neue Beschriftungsvariable in der `\paper`-Umgebung definiert werden
- dann muss die musikalische Funktion definiert werden, die ein Element zum Inhaltsverzeichnis hinzufügt, indem die neue Variable benutzt wird.

Das folgende Beispiel definiert einen neuen Stil um Akt-Bezeichnungen einer Oper in das Inhaltsverzeichnis aufzunehmen:

```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}

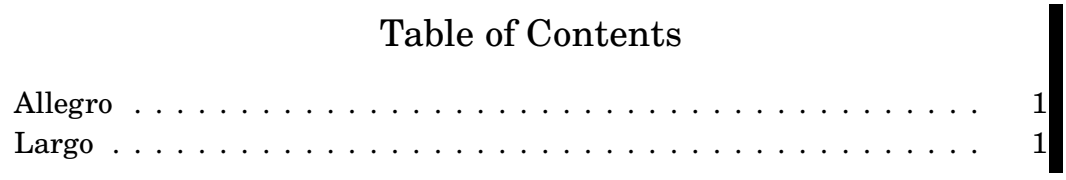
tocAct =
#(define-music-function (parser location text) (markup?)
   (add-toc-item! 'tocActMarkup text))
```



Die Zeile zwischen dem Inhalt und der Seitenzahl kann mit einer gepunkteten Linie gefüllt werden:

```
\header { tagline = ##f }
\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}

\book {
  \markuplines \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}
```



Siehe auch

Installierte Dateien: '`../ly/toc-init.ly`'.

Vordefinierte Befehle

`\table-of-contents`, `\tocItem`.

3.3 Arbeiten an Eingabe-Dateien

3.3.1 LilyPond-Dateien einfügen

Ein größeres Projekt kann in einzelne Dateien aufgeteilt werden. Um eine andere Datei einzubinden, kann der Befehl

```
\include "andereDatei.ly"
```

benutzt werden.

Die Zeile `\include "andereDatei.ly"` benimmt sich genauso, also ob der Inhalt der Datei '`andereDatei.ly`' komplett in die Hauptdatei eingefügt werden würde. So kann man für ein größeres Projekt die einzelnen Stimmen der Instrumente getrennt notieren und sie dann in einer Partitur-Datei benutzen. Meistens werden in den eingefügten Dateien einige Variablen definiert, die dann auch in der Hauptdatei eingesetzt werden können. Mit Marken (Tags) gekennzeichnete Abschnitte können eingesetzt werden, um die entsprechenden Noten etc. an verschiedenen Stellen in der Datei zur Verfügung zu stellen. Siehe auch Abschnitt 3.3.2 [Verschiedene Editionen aus einer Quelldatei], Seite 442.

Auf Dateien im aktuellen Verzeichnis kann einfach mit dem Dateinamen nach dem `\include`-Befehl verwiesen werden. Dateien an anderen Stellen können eingebunden werden, indem entweder ein vollständiger Pfad oder ein relativer Pfad zu der Datei angegeben wird. Hierbei sollten die für UNIX typischen Schrägstriche (/) und nicht die rückwärtsgeneigten von Windows (\) verwendet werden, um die Verzeichnisse zu trennen. Wenn etwa die Datei '`kram.ly`' ein Verzeichnis höher liegt als das aktuelle Verzeichnis, sollte der Befehl so aussehen:

```
\include "../kram.ly"
```

Wenn die Orchesterstimmen andererseits in einem Unterordner mit der Bezeichnung `stimmen` liegen, sieht er folgendermaßen aus:

```
\include "stimmen/VI.ly"
\include "stimmen/VII.ly"
... etc
```

Dateien, die eingebunden werden sollen, können selber auch wiederum ein `\include` enthalten. Diese Einbindung zweiter Ebene werden erst interpretiert, wenn sie sich in der Hauptdatei befinden, sodass die Pfadangaben hier nicht relativ zur eingebundenen Datei, sondern relativ zur Hauptdatei gesetzt werden müssen. Dieses Verhalten kann jedoch auch verändert werden, indem man lilypond die Option `-drelative-includes` auf der Kommandozeile zuweist (oder indem man den Befehl `#(ly:set-option 'relative-includes #t)` an den Beginn der Quelldatei schreibt). Mit `relative-includes` wird der Pfad jedes `\include`-Befehls als relativ zu der Datei angenommen, in der sich der Befehl befindet. Dieses Verhalten wird empfohlen und wird in zukünftigen Versionen von LilyPond den Standard darstellen.

Dateien können auch aus einem Verzeichnis eingebunden werden, dass im Suchpfad von LilyPond liegt. Hierzu muss auf der Kommandozeile das entsprechende Verzeichnis angegeben werden und die Dateien, die eingebunden werden, müssen nur mit ihrem Namen notiert sein. Wenn etwa die Datei '`Haupt.ly`' kompiliert werden soll, die Dateien aus dem Unterverzeichnis '`stimmen`' einbindet, müssen sie sich im Verzeichnis von '`Haupt.ly`' befinden und dann LilyPond folgendermaßen aufrufen:

```
lilypond --include=stimmen Haupt.ly
```

In '`Haupt.ly`' steht:

```
\include "VI.ly"
```

```
\include "VII.ly"
... usw.
```

Dateien, die in vielen Partituren verwendet werden sollen, können im LilyPond-Verzeichnis '`../ly`' gespeichert werden. (Die Stelle, an der dieses Verzeichnis sich befindet, hängt vom Betriebssystem ab, siehe hierzu Abschnitt "Mehr Information" in *Handbuch zum Lernen*). Dateien in diesem Verzeichnis können einfach mit ihrem Namen eingefügt werden. So werden auch die Sprachdateien wie etwa '`deutsch.ly`' eingefügt.

LilyPond lädt eine Anzahl an Dateien, wenn das Programm aufgerufen wird. Diese Dateien sind für den Benutzer nicht ersichtlich, aber die Dateien können identifiziert werden, indem LilyPond auf der Kommandozeile mit Option aufgerufen wird: `lilypond --verbose`. Hiermit wird neben anderer Information auch eine Liste von Pfaden und Dateien aufgeführt, die LilyPond benutzt. Die wichtigeren Dateien werden im Abschnitt Abschnitt "Mehr Information" in *Handbuch zum Lernen* besprochen. Diese Dateien können verändert werden, aber Änderungen gehen verloren, wenn eine neue LilyPond-Version installiert wird.

Eine einfache Beispiele, die die Benutzung von `\include` demonstrieren, sind dargestellt in Abschnitt "Partituren und Stimmen" in *Handbuch zum Lernen*.

Siehe auch

Handbuch zum Lernen: Abschnitt "Mehr Information" in *Handbuch zum Lernen*, Abschnitt "Partituren und Stimmen" in *Handbuch zum Lernen*.

Bekannte Probleme und Warnungen

Wenn eine Datei eingebunden wird, deren Name einer Datei aus dem Installationsverzeichnis von LilyPond entspricht, wird die installierte Datei anstelle der eigenen verwendet.

3.3.2 Verschiedene Editionen aus einer Quelldatei
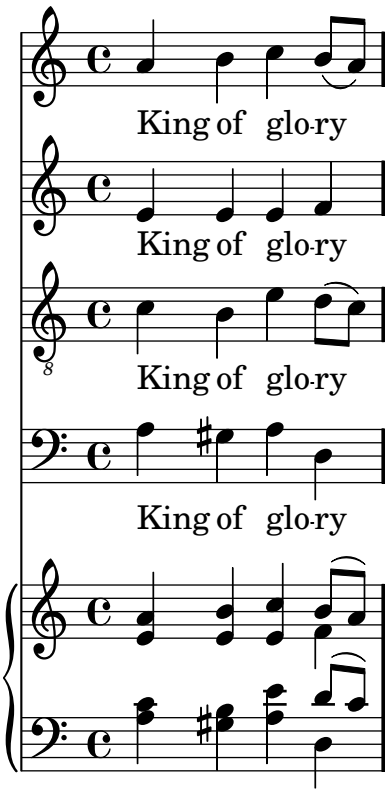
Es gibt verschiedene Funktionen, die es möglich machen, unterschiedliche Versionen einer Partitur aus der gleichen Quelldatei zu produzieren. Variablen werden am besten eingesetzt, wenn es darum geht, längere Notenpassagen und/oder Anmerkungen/Textmarken miteinander auf verschiedene Weise zu kombinieren. Tag-Marken dagegen werden am besten eingesetzt, wenn eine von mehreren kurzen alternativen Notenabschnitten ausgewählt werden soll. Egal welche Methode am Ende eingesetzt wird: Es erleichtert die Arbeit in jedem Fall, wenn die eigentlichen Noten und die Struktur der Partitur voneinander getrennt notiert werden – so kann die Struktur geändert werden, ohne dass man Änderungen an den Noten vornehmen muss.

Variablen benutzen

Wenn Notenabschnitt in Variablen definiert werden, können sie an unterschiedlichen Stellen in der Partitur eingesetzt werden, siehe auch Abschnitt "Stücke durch Bezeichner organisieren" in *Handbuch zum Lernen*. Zum Beispiel enthält eine Vokalpartitur für ein *a cappella* Stück oft einen Klavierauszug, der das Einüben einfacher macht. Der Klavierauszug enthält die gleichen Noten, sodass man sie nur einmal notieren muss. Noten aus zwei Variablen können auf einem System kombiniert werden, siehe [Automatische Kombination von Stimmen], Seite 154. Hier ein Beispiel:

```
sopranoMusic = \relative c'' { a4 b c b8( a) }
altoMusic = \relative g' { e4 e e f }
tenorMusic = \relative c' { c4 b e d8( c) }
bassMusic = \relative c' { a4 gis a d, }
allLyrics = \lyricmode {King of glo -- ry }
<<
  \new Staff = "Soprano" \sopranoMusic
```

```
  \new Lyrics \allLyrics
  \new Staff = "Alto" \altoMusic
  \new Lyrics \allLyrics
  \new Staff = "Tenor" {
    \clef "treble_8"
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Bass" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "RH" {
      \set Staff.printPartCombineTexts = ##f
      \partcombine
      \sopranoMusic
      \altoMusic
    }
    \new Staff = "LH" {
      \set Staff.printPartCombineTexts = ##f
      \clef "bass"
      \partcombine
      \tenorMusic
      \bassMusic
    }
  >>
>>
```



Unterschiedliche Partituren, die entweder nur den Chor oder das Klavier zeigen, können produziert werden, indem die Struktur verändert wird; die Noten müssen dazu nicht verändert werden.

Für längere Partituren können Variablen in eigene Dateien notiert werden, die dann eingebunden werden, siehe Abschnitt 3.3.1 [LilyPond-Dateien einfügen], Seite 441.

Marken benutzen

Der `\tag #'`*`TeilA`*-Befehl markiert einen musikalischen Ausdruck mit der Bezeichnung *TeilA*. Ausdrücke, die auf diese Weise markiert werden, können mit ihrer Bezeichnung später ausgewählt bzw. ausgefiltert werden. Das geschieht mit den Befehlen `\keepWithTag #'`*`Bezeichnung`* bzw. `\removeWithTag #'`*`Bezeichnung`*. Die Wirkung dieser Filter auf die markierten Notenabschnitte ist wie folgt:

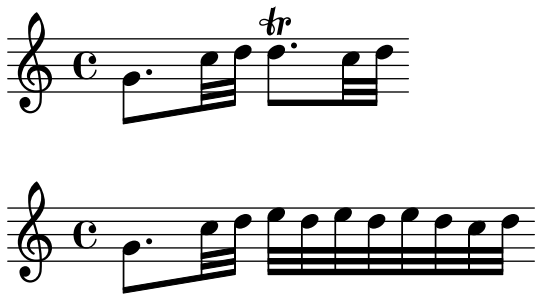
Filter	**Resultat**
Markierte Noten mit vorgesetztem `\keepWithTag #'`*`Bezeichnung`*	Unmarkierte Noten und Noten mit der Marke *Bezeichnung* werden gesetzt, Noten mit einer anderen Marke werden nicht angezeigt.
Markierte Noten mit vorgesetztem `\removeWithTag #'`*`Bezeichnung`*	Unmarkierte Noten und Noten mit einer anderen Marke als *Bezeichnung* wird angezeigt, Noten markiert mit *Bezeichnung* werden nicht angezeigt.
Markierte Noten, weder mit vorgesetztem `\keepWithTag` noch `\removeWithTag`	Alle markierten und unmarkierten Noten werden angezeigt.

Die Argumente der Befehle `\tag`, `\keepWithTag` und `\removeWithTag` sollten ein Symbol sein (wie etwa `#'score` oder `#'part`), gefolgt von einem musikalischen Ausdruck.

Im folgenden Beispiel erscheinen zwei Versionen der Noten, eine zeigt Triller in normaler Notation, die andere zeigt sie ausgeschrieben:

```
music = \relative g' {
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand { \repeat unfold 3 { e32 d } }
  c32 d
 }

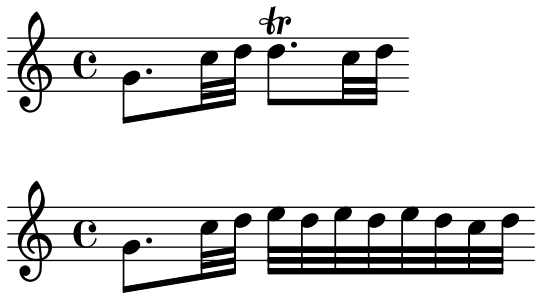
\score {
  \keepWithTag #'trills \music
}
\score {
  \keepWithTag #'expand \music
}
```



Entsprechend können auch Abschnitte ausgeschlossen werden; das erfordert manchmal weniger Schreibarbeit:

```
music = \relative g' {
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand {\repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \removeWithTag #'expand
  \music
}
\score {
  \removeWithTag #'trills
  \music
}
```



Marken können auch auf Artikulationen, Text usw angewendet werden, indem man ihnen

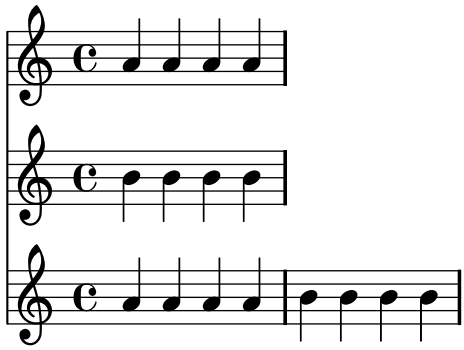
`-\tag #'`*`your-tag`*

voranstellt (jedoch nach der Note, an die sie gebunden sind). Mit diesem Code etwa könnte man entweder Fingersatz oder aber einen Text ausgeben:

```
c1-\tag #'finger ^4
c1-\tag #'warn ^"Achtung!"
```
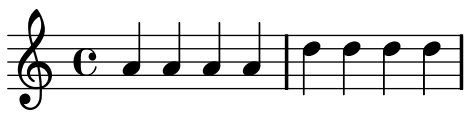
Mehrfache Marken können mithilfe von mehreren `\tag`-Befehlen notiert werden:

```
music = \relative c'' {
  \tag #'a \tag #'both { a4 a a a }
  \tag #'b \tag #'both { b4 b b b }
}
<<
\keepWithTag #'a \music
\keepWithTag #'b \music
\keepWithTag #'both \music
>>
```



Mehrfache `\removeWithTag`-Filter können auf einen musikalischen Ausdruck angewendet werden, um mehrere unterschiedliche markierte Abschnitte aus dem Druckbild zu entfernen.

```
music = \relative c'' {
\tag #'A { a4 a a a }
\tag #'B { b4 b b b }
\tag #'C { c4 c c c }
\tag #'D { d4 d d d }
}
{
\removeWithTag #'B
\removeWithTag #'C
\music
}
```



Zwei oder mehr `\keepWithTag`-Filter in einem musikalischen Ausdruck bewirken, dass *alle* markierten Abschnitte entfernt werden, weil der erste Befehl alle markierten Abschnitt außer dem im Befehl genannten wegfiltert und der zweite Befehl dann auch diesen eben genannten zusätzlich entfernt.

Siehe auch

Handbuch zum Lernen: Abschnitt “Stücke durch Bezeichner organisieren” in *Handbuch zum Lernen*.

Notationsreferenz: [Automatische Kombination von Stimmen], Seite 154, Abschnitt 3.3.1 [LilyPond-Dateien einfügen], Seite 441.

Globale Einstellungen benutzen

Man kann globale Einstellungen aus einer externen Datei einfügen:

```
lilypond -dinclude-settings=MY_SETTINGS.ly MY_SCORE.ly
```

Einstellungsgruppen, wie etwa Seitengröße, Schriftart oder Schriftschnitt, können in eigenen Dateien gespeichert werden. Das ermöglicht es, aus der gleichen Partitur unterschiedliche Editionen zu erstellen bzw. Standardeinstellungen für eine ganze Anzahl von Partituren wiederzuverwenden, indem man einfach die entsprechende Einstellungsdatei angibt.

Diese Technik funktioniert auch gut für globale Formatierungen, wie behandelt in Abschnitt “Globale Formatierung” in *Handbuch zum Lernen*.

Siehe auch

Handbuch zum Lernen: Abschnitt “Stücke durch Bezeichner organisieren” in *Handbuch zum Lernen*, Abschnitt “Globale Formatierung” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 3.3.1 [LilyPond-Dateien einfügen], Seite 441.

3.3.3 Zeichenkodierung

LilyPond benutzt alle Zeichen, die durch das Unicode-Konsortium und ISO/IEC 10646 definiert sind. Hiermit wird den Zeichen fast aller Schriftsysteme der Welt ein eindeutiger Name und ein Code-Punkt zugewiesen, mit dem sie identifizierbar sind. Unicode kann mit mehreren Zeichenkodierungen verwirklicht werden. LilyPond benutzt die UTF-8-Kodierung (UTF = Unicode Transformation Format), in der die normalen Zeichen des lateinischen Alphabets mit einem Byte dargestellt werden, während alle anderen Zeichen zwischen zwei und vier Byte Länge haben.
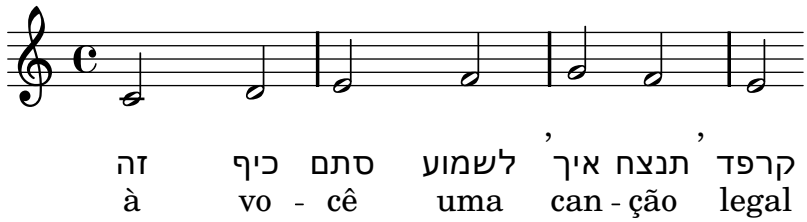
Das Aussehen des Zeichens wird bestimmt durch die gerade benutzte Schriftart (engl. font). In einer Schriftartdatei werden die Nummern der Unicode-Zeichen einem bestimmten Glyphen zugeordnet. LilyPond verwendet die Pango-Bibliothek um mehrsprachige Texte und komplexe Skripte korrekt zu setzen.

LilyPond verändert die Kodierung der Eingabedatei nicht. Das heißt, dass jeder Text – Überschriften, Gesangstext, Spielanweisungen etc. – der nicht nur aus ASCII-Zeichen besteht, in UTF-8 kodiert sein musst. Am einfachsten geht das, indem man einen Texteditor einsetzt, der mit Unicode-Zeichen umgehen kann. Die meisten modernen weit verbreiteten Editoren besitzen heute UTF-8-Unterstützung, wie etwa vim, Emacs, jEdit oder GEdit. Alle MS Windows-Systeme nach NT benutzen Unicode intern, sodass sogar Notepad Dateien in UTF-8 lesen und speichern kann. Ein Editor mit mehr Funktionen unter Windows ist BabelPad oder Notepad++.

Wenn eine LilyPond-Eingabedatei nicht-ASCII-Zeichen enthält und nicht in UTF-8 gespeichert ist, gibt es folgende Fehlermeldung:

```
FT_Get_Glyph_Name () error: invalid argument
```

Heir ein Beispiel mit Kyrilliza, hebräischem und portugiesischem Text:



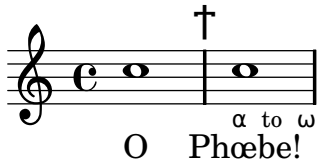
Um einen einzelnen Buchstaben zu notieren, für den die Unicode-Ziffernfolge bekannt ist, der aber nicht auf der Tastatur zu finden ist, kann der Befehl `\char ##xhhhh` oder `\char #dddd` innerhalb einer `\markup`-Umgebung benutzt werden. Hierbei bedeutet `hhhh` die hexadezimale Zahl und `ddd` die entsprechende dezimale Zahl für das erforderliche Zeichen. Nullen zu Beginn können ausgelassen werden, aber normalerweise werden alle vier Zeichen der hexadezimalen Notation notiert. (Achten Sie darauf, dass Sie *nicht* UTF-8-Codepunkte einsetzen, weil UTF-8 zusätzliche Bits enthält, die die Nummer der Oktets bezeichnet.) Unicode-Tabellen und ein Verzeichnis der Zeichenbezeichnungen mit einer hexadezimalen Verweiszahl finden sich auf der Internetseite des Unicode Consortiums: http://www.unicode.org/.

Mit `\char ##x03BE` und `\char #958` wird beispielsweise das Unicode-Zeichen U+03BE notiert, welches die Unicode-Bezeichnung „Greek Small Letter Xi“ hat.

Alle existierenden Unicode-Zeichen können auf diese Weise notiert werden, und wenn für alle Zeichen dieses Format angewandt wird, muss die Datei nicht im utf-8-Format gespeichert werden. Es muss natürlich auch noch eine Schriftart auf dem System installiert sein, die die notierten Zeichen darstellen kann.

Das nächste Beispiel zeigt, wie Unicode-Zeichen an vier Stellen mit dem hexadezimalen Code notiert werden: in einem Übungszeichen, als Artikulationszeichen, im Gesangstext und als normaler Text außerhalb der Partitur.

```
\score {
  \relative c'' {
    c1 \mark \markup { \char ##x03EE }
    c1_\markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } }
  }
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2011" \char ##x00A9 }
```



Copyright 2008--2011 ©

Um das Copyright-Zeichen zu notieren, kann folgender Code eingesetzt werden:

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

3.3.4 LilyPond-Notation anzeigen

Ein musikalischer Ausdruck in LilyPond-Notation kann mit der Funktion `\displayLilyMusic` angezeigt werden. Der Code

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

etwa wird ausgeben:

```
{ a,4 cis e fis g }
```

Normalerweise werden diese Zeilen zusammen mit allen anderen Kompilations-Nachrichten auf der Kommandozeile ausgegeben. Um sie separat zu speichern und das Ergebnis von

`\displayLilyMusic` weiterzubenutzen, kann die Ausgabe mit folgendem Befehl in eine Datei umgeleitet werden:

```
lilypond file.ly >display.txt
```

3.4 Ausgabe kontrollieren

3.4.1 Notationsfragmente extrahieren

Es ist möglich, kleine Abschnitte einer großen Partitur direkt aus der Quelldatei zu erzeugen. Das kann damit verglichen werden, dass man mit der Schere bestimmte Regionen ausschneidet.

Es wird erreicht, indem man die Takte, die ausgeschnitten werden sollen (engl. to clip = ausschneiden), extra definiert. Mit folgender Definition beispielsweise

```
\layout {
  clip-regions
  = #(list
      (cons
       (make-rhythmic-location 5 1 2)
       (make-rhythmic-location 7 3 4)))
}
```

wird ein Fragment ausgeschnitten, dass auf der Mitte des fünften Taktes beginnt und im siebten Takt endet. Die Bedeutung von `5 1 2` ist: nach einer Halben in Takt fünf, `7 3 4` heißt: nach drei Vierteln in Takt 7.

Weitere Bereiche, die ausgeschnitten werden sollen, können definiert werden, indem mehrere derartige Paare definiert werden.

Um diese Funktion auch nutzen zu können, muss LilyPond mit dem Parameter `-dclip-systems` aufgerufen werden. Die Schnipsel werden als EPS ausgegeben und dann zu PDF und PNG konvertiert, wenn diese Formate auch als Parameter angegeben werden.

Zu mehr Information über Ausgabeformate siehe Abschnitt “lilypond aufrufen” in *Anwendungsbenutzung*.

3.4.2 Korrigierte Musik überspringen

Wenn man Noten eingibt oder kopiert, sind meistens nur die Noten nahe dem Ende (wo gerade neue Noten notiert wurden) wichtig für Kontrolle und Korrektur. Um die Korrektur zu beschleunigen, kann eingestellt werden, dass nur die letzten paar Takte angezeigt werden. Das erreicht man mit dem Befehl

```
showLastLength = R1*5
\score { ... }
```

in der Quelldatei. Damit werden nur die letzten fünf Takte (in einem 4/4-Takt) eines jeden `\score`-Abschnitts übersetzt. Besonders bei längeren Stücken ist es meistens sehr viel schneller, nur einen kleinen Teil des Stückes zu setzen als die gesamte Länge. Wenn man am Anfang eines Stückes arbeitet (weil etwa ein neuer Teil hinzugefügt werden soll), kann auch die `showFirstLength`-Eigenschaft nützlich sein.

Nur bestimmte Teile einer Partitur zu überspringen, kann mit der Eigenschaft `Score.skipTypesetting` sehr genau kontrolliert werden. Für den Bereich, für den sie auf „wahr“ gesetzt wird, wird kein Notensatz ausgegeben.

Diese Eigenschaft kann auch benutzt werden, um die MIDI-Ausgabe zu kontrollieren. Hiermit werden alle Ereignisse, auch Tempo- und Instrumentenwechsel ausgelassen. Man muss also sehr genau darauf achten, dass nichts unerwartetes geschieht.

```
c8 d
```

```
\set Score.skipTypesetting = ##t
e8 e e e e e e e
\set Score.skipTypesetting = ##f
c8 d b bes a g c2
```



In polyphoner Notation wirkt sich `Score.skipTypesetting` auf alle Stimmen und Systeme aus, sodass noch mehr Zeit bei der Übersetzung der Datei gespart wird.

3.4.3 Alternative Ausgabeformate
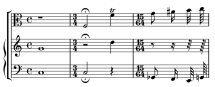
Das Standardausgabeformat für gedruckte Partituren ist PDF (Portable Document Forma) und PS (PostScript). SVG (Scalable Vector Graphics), EPS (Encapsulated PostScript) und PNG (Portable Network Graphics) gibt es auch als Ausgabeformate über die Kommandozeile. Siehe Abschnitt "Optionen von lilypond auf der Kommandozeile" in *Anwendungsbenutzung*.

3.4.4 Die Notationsschriftart verändern
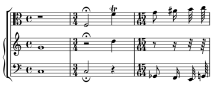
Gonville ist eine Alternative zu der Feta-Schriftart, die in LilyPond eingesetzt wird und kann unter der Adresse

`http://www.chiark.greenend.org.uk/~sgtatham/gonville/`

heruntergeladen werden. Hier einige Takte Noten mit der Gonville-Schriftart:



Und hier einige Beispieltakte in der Feta-Schriftart:



Installationsanweisungen für MacOS

Laden Sie die Datei herunter und entpacken Sie die ZIP-Datei. Kopieren Sie das `lilyfonts`-Verzeichnis nach '*`SHARE_DIR`*`/lilypond/current`'; für mehr Information siehe Abschnitt "Mehr Information" in *Handbuch zum Lernen*. Benennen Sie das existierende `fonts`-Verzeichnis in `fonts_orig` um und benennen Sie das Verzeichnis `lilyfonts` in `fonts`. Das alte Verzeichnis `fonts_orig` können Sie einfach in `fonts` zurückbenennen, um wieder nach Feta zu wechseln.

Siehe auch

Handbuch zum Lernen: Abschnitt "Mehr Information" in *Handbuch zum Lernen*.

Bekannte Probleme und Warnungen

Gonville kann nicht verwendet werden, um Alte Notation zu setzen. Bitte lesen Sie die Webseite des Autoren zu mehr Information hierzu und zu anderen Einzelheiten, wie auch der Lizenz von Gonville.

3.5 MIDI-Ausgabe

MIDI (Musical Instrument Digital Interface) ist ein Standard zur Kontrolle und Interaktion mit digitalen Instrumenten. Eine MIDI-Datei ist eine Anzahl von Noten auf einer Anzahl von Bändern/Stimmen. Es ist keine eigentliche Klangdatei, denn man benötigt spezielle Programme die die Notenereignisse in Klang umwandeln können.

Der Notensatz von LilyPond kann in MIDI umgewandelt werden, so dass man sich anhören kann, was man notiert hat. Das hilft oft sehr gut bei der Überprüfung: falsche Oktaven oder falsche Versetzungszeichen lassen sich meist sehr gut hören.

Die MIDI-Ausgabe benötigt einen Kanal für jedes System und einen für globale Einstellungen. Darum sollte die Quelldatei für eine MIDI-Datei nicht mehr als 15 Systeme (oder 14 wenn kein Schlagzeug benützt wird) besitzen. Jedes weitere System bleibt stumm.

3.5.1 MIDI-Dateien erstellen

Um eine MIDI-Datei aus einer LilyPond-Quelldatei zu erstellen, muss eine `\midi`-Umgebung zu der `\score`-Umgebung hinzugefügt werden, etwa so:

```
\score {
  ...Noten...
  \midi { }
}
```

Wenn in einer `\score`-Umgebung nur eine `\midi`-Umgebung, aber keine `\layout`-Umgebung vorkommt, wird nur MIDI produziert. Wenn auch die Notation gewünscht ist, muss zusätzlich die `\layout`-Umgebung vorhanden sein:

```
\score {
  ...music...
  \midi { }
  \layout { }
}
```

Tonhöhen, Rhythmen, Überbindungen, Dynamik und Tempoänderungen werden korrekt in das MIDI-Format übersetzt. Dynamikzeichen, Crescendo und Decrescendo werden in den MIDI-Lautstärkekanal übertragen. Dynamikzeichen werden in einen bestimmten Lautstärkenwert übersetzt, Crescendo und Decrescendo erreichen einen Übergang zwischen Lautstärkewerten. Die Wirkung von Dynamikzeichen kann auch aus der MIDI-Datei entfernt werden. Siehe hierzu Abschnitt 3.5.2 [Der MIDI-Block], Seite 453.

Das Anfangstempo und spätere Tempoänderungen können mit dem `\tempo`-Befehl innerhalb der Notation notiert werden. Er bewirkt Tempoänderungen auch in der MIDI-Datei. Der Befehl setzt gleichzeitig auch eine Tempobezeichnung in die Noten, welches aber auch unterdrückt werden kann, siehe [Metronomangabe], Seite 63. Eine andere Möglichkeit, ein eigenes MIDI-Tempo anzugeben, wird weiter unten gezeigt, siehe Abschnitt 3.5.2 [Der MIDI-Block], Seite 453.

Aufgrund einiger Einschränkungen auf Windows ist auf Windows-Systemen die Standard-dateierweiterung von MIDI-Dateien `.mid`. Andere Betriebssysteme verwenden weiterhin `.midi`. Wenn eine andere Endung erwünscht ist, kann man die folgende Zeile auf oberster Ebene der Quelldatei, vor Beginn eines `\book`, `\bookpart` oder `\score`-Blocks einfügen:

```
#(ly:set-option 'midi-extension "midi")
```

Diese Codezeile setzt die Dateiendung auf `.midi`.

Als Alternative kann man diese Option auch als Kommandozeilenparameter übergeben:

```
lilypond ... -dmidi-extension=midi lilyDatei.ly
```

Instrumentenbezeichnungen

Das MIDI-Instrument, mit dem ein bestimmtes System wiedergegeben werden soll, wird durch die `Staff.midiInstrument`-Eigenschaft bestimmt, die auf eine Instrumentenbezeichnung gesetzt werden muss. Die Bezeichnungen sind aufgelistet in Abschnitt A.5 [MIDI-Instrumente], Seite 576 und müssen in der dort definierten Schreibweise notiert werden.

```
\new Staff {
  \set Staff.midiInstrument = #"glockenspiel"
  ...Noten...
}
\new Staff \with {midiInstrument = #"cello"} {
  ...Noten...
}
```

Wenn die Schreibweise nicht genau einem definierten Instrument aus der Liste entspricht, wird ein Piano-Klang benutzt (`"acoustic grand"`).

Ausgewählte Schnipsel
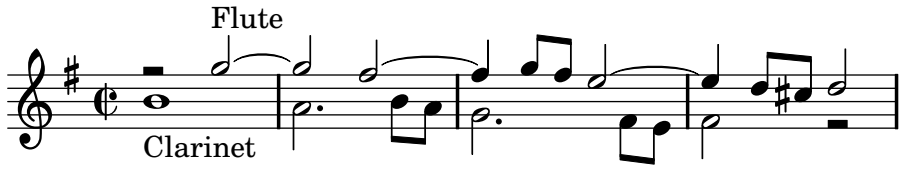
Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per track.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `midiInstrument`.

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = #"flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = #"clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
```

```
    \context {
      \Voice
      \consists "Staff_performer"
    }
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
    }
  }
}
```



Bekannte Probleme und Warnungen

Veränderungen der MIDI-Lautstärke sind nur effektiv, wenn sie zu Beginn einer Note angefordert werden, sodass die Lautstärke während einer Notendauer nicht geändert werden kann.

Nicht alle MIDI-Spieler können Tempoänderungen richtig wiedergeben. Spieler, die hierzu in der Lage sind, sind unter Anderen MS Windows Media Player und timidity.

3.5.2 Der MIDI-Block

Eine `\midi`-Umgebung muss innerhalb von einer `\score`-Umgebung vorkommen, wenn MIDI-Ausgabe gewünscht ist. Sie entspricht der `\layout`-Umgebung, aber ist etwas einfacher aufgebaut. Oft wird die MIDI-Umgebung einfach leer gelassen, aber hier können auch Kontexte umgeändert werden, neue Kontexte definiert werden oder neue Werte definiert werden. Das folgende Beispiel etwa definiert das MIDI-Tempo, ohne dass in der Partitur eine Metronombezeichnung gesetzt wird:

```
\score {
  ...Noten...
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 4)
    }
  }
}
```

Hier wird das Tempo auf 72 Viertelnoten pro Minute definiert. Wenn das Tempo auf diese Weise definiert wird, kann keine punktierte Note als Einheit angegeben werden. Wenn sie benötigt wird, muss man sie in kleinere Einheiten auflösen. Ein Tempo von 90 punktierten Viertelnoten pro Minute kann beispielsweise dargestellt werden als 270 Achtelnoten pro Minute:

```
tempoWholesPerMinute = #(ly:make-moment 270 8)
```

Kontextdefinitionen des `\midi`-Kontextes entsprechen der Syntax, wie sie in der `\layout`-Umgebung benutzt wird. Klangübersetzungsmodule werden `performer` genannt. Die Kontexte für die MIDI-Ausgabe sind in der Datei '`../ly/performer-init.ly`' definiert, siehe Abschnitt "Mehr Information" in *Handbuch zum Lernen*. Um beispielsweise die Auswirkung von Dynamikzeichen aus der MIDI-Ausgabe zu entfernen, müssen folgende Zeilen eingefügt werden:

```
\midi {
  ...
  \context {
    \Voice
    \remove "Dynamic_performer"
  }
}
```

Eine MIDI-Ausgabe wird nur erstellt, wenn die `\midi`-Umgebung in eine Partiturumgebung eingefügt wird, die mit dem Befehl `\score` beginnt.

```
\score {
  { ...Noten... }
  \midi { }
}
```

3.5.3 Was geht in die MIDI-Ausgabe

In MIDI unterstützt

Die folgenden Notationselemente werden in die MIDI-Ausgabe aufgenommen:

- Tonhöhen
- Mikrotöne (siehe [Versetzungszeichen], Seite 5. Für die Ausgabe wird ein Spieler benötigt, der Tonhöhen verändern kann.)
- Akkorde, die als Symbole notiert werden
- Rhythmen, die als Dauern notiert sind, inklusive N-tolen
- Tremolo, das ohne ‚:[*Zahl*]‘ notiert ist
- Überbindungen
- Dynamikzeichen
- Crescendi, decrescendi zu mehreren Noten
- Tempoänderungen, die mit einer Tempo-Bezeichnung eingegeben werden
- Gesangstext

In MIDI nicht unterstützt

Folgende Notationselemente werden nicht in die MIDI-Ausgabe einbezogen:

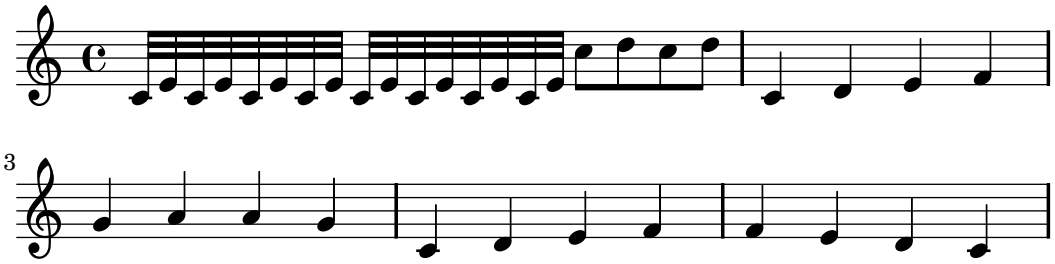
- Rhythmus, der als Anmerkung notiert wird, bspw. Swing
- Tempoveränderungen, die als Anmerkung ohne Tempobezeichnung notiert werden
- Staccato und andere Artikulationen und Ornamente
- Legato- und Phrasierungsbögen
- Crescendi, decrescendi zu einer einzelnen Note
- Tremolo, notiert mit ‚:[*number*]‘
- Bezifferter Bass
- Akkorde mit Mikrotönen

3.5.4 Wiederholungen im MIDI

Mit einigen Veränderungen im Notentext können alle Wiederholungstypen auch in der MIDI-Ausgabe wiedergegeben werden. Das wird erreicht, indem die `\unfoldRepeats`-Funktion eingesetzt wird. Diese Funktion verändert alle Wiederholungen in ausgeschriebene Noten.

```
\unfoldRepeats {
  \repeat tremolo 8 { c'32 e' }
```

```
  \repeat percent 2 { c''8 d'' }
  \repeat volta 2 { c'4 d' e' f' }
  \alternative {
    { g' a' a' g' }
    { f' e' d' c' }
  }
}
\bar "|."
```



Wenn eine Partitur mit diesem `\unfoldRepeats`-Befehl erstellt wird, ist er notwendig, zwei `\score`-Umgebungen einzurichten: in der einen werden die Wiederholungen ausgeschrieben und nur eine MIDI-Ausgabe produziert, in der anderen werden die Wiederholungen notiert und als Partitur gesetzt. Das Beispiel gibt einen Hinweis, wie eine derartige Datei aussehen kann:

```
\score {
  ..music..
  \layout { .. }
}
\score {
  \unfoldRepeats ..music..
  \midi { .. }
}
```

3.5.5 MIDI-Lautstärke kontrollieren

Dynamik in der MIDI-Ausgabe wird durch den Dynamic_performer erstellt, welcher sich in einem `Voice`-Kontext befindet. Es ist möglich, sowohl die generelle Lautstärke einer MIDI-Datei als auch relative Lautstärken von Dynamikanweisungen und auch relative Lautstärke von einzelnen Instrumenten einzustellen.

Dynamik-Zeichen

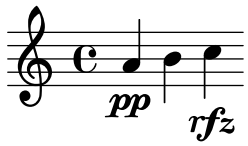
Dynamikanweisungen werden als ein bestimmter Bruch der insgesamt zur Verfügung stehenden MIDI-Lautstärke notiert. Die Standardbrüche reichen von 0,25 für *ppppp* bis hin zu 0,95 für *fffff*. Diese Anweisung befinden sich in der Datei '`../scm/midi.scm`', siehe auch Abschnitt "Mehr Information" in *Handbuch zum Lernen*. Diese Brüche können nach Belieben geändert oder erweitert werden, indem eine Funktion erstellt wird, die ein Dynamikzeichen als Argument nimmt und den erforderlichen Bruch ausgibt; schließlich muss noch `Score.dynamicAbsoluteVolumeFunction` auf diese Funktion gesetzt werden.

Beispielhaft soll gezeigt werden, wie man eine *Rinforzando*-Dynamik, `\rfz`, auch in die MIDI-Ausgabe übernehmen kann. Gleiches gilt für neue, selbstdefinierte Dynamikzeichen, die in den Standarddefinitionen nicht enthalten sind. Die Scheme-Funktion, die hier definiert wird, setzt den Bruch von 0,9 für eine rfz-Anweisung und ruft andernfalls die Standardanweisungen auf:

```
#(define (myDynamics dynamic)
    (if (equal? dynamic "rfz")
      0.9
```

```
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative c'' {
        a4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



Alternativ, insbesondere wenn die gesamte Tabelle der MIDI-Lautstärken umdefiniert werden soll, ist es besser, die *default-dynamic-absolute-volume*-Prozedur in der Datei '`../scm/midi.scm`' und die hiermit verknüpfte Tabelle als Modell zu benutzen. Das letzte Beispiel dieses Abschnittes zeigt, wie das gemacht werden kann.

MIDI-Lautstärke
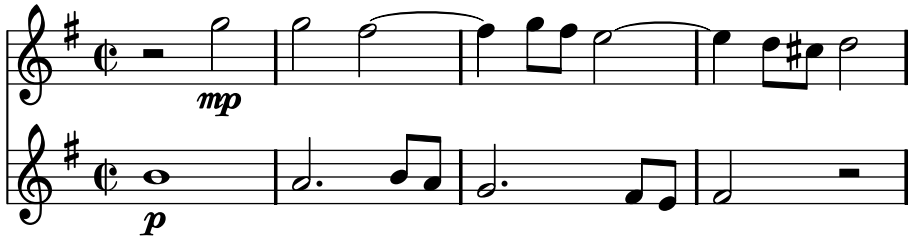
Die generellen Mindest- und Höchstwerte für die Lautstärke der MIDI-Datei wird kontrolliert, indem die Eigenschaften `midiMinimumVolume` und `midiMaximumVolume` auf der `Score`-Ebene gesetzt werden. Diese Eigenschaften haben nur Einfluss auf Dynamikzeichen, sodass ein Dynamikzeichen direkt an den Anfang der Partitur gestellt werden muss, wenn diese Einstellung von Anfang an Wirkung zeigen soll. Der Bruch, der dann den einzelnen Dynamikzeichen entspricht, wird mit der Formel

```
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) * Bruch
```

errechnet. Im folgenden Beispiel wird die generelle MIDI-Lautstärke auf den Bereich zwischen 0.2 und 0.5 eingeschränkt.

```
\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Staff.midiInstrument = #"flute"
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
```

```
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout {}
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}
```
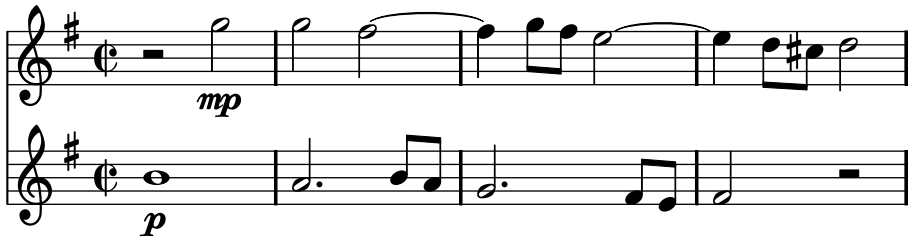


Verschiedene Instrumente angleichen (i)

Wenn die Mindest- und Höchstwerte für die MIDI-Lautstärke innerhalb eines `Staff`-Kontextes gesetzt werden, kann damit die relative Lautstärke einzelner Instrumente kontrolliert werden. Damit kann man die Qualität der MIDI-Datei merklich verbessern.

In diesem Beispiel wird die Lautstärke der Klarinette relativ zur Lautstärke der Flöte verringert. In jeder Stimme muss eine Dynamikanweisung für die erste Note gesetzt werden, damit diese Einstellung korrekt funktioniert.

```
\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Staff.midiInstrument = #"flute"
      \set Staff.midiMinimumVolume = #0.7
      \set Staff.midiMaximumVolume = #0.9
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \set Staff.midiMinimumVolume = #0.3
```

```
      \set Staff.midiMaximumVolume = #0.6
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout {}
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
    }
  }
}
```



Verschiedene Instrumente angleichen (ii)

Wenn Mindest- und Höchstwerte für die Lautstärke der MIDI-Datei nicht vorgegeben werden, nimmt LilyPond standardmäßig einige Anpassungen für die Lautstärken bestimmter Instrumente vor. Diese Instrumente und ihre entsprechende Veränderung lassen sich aus der Tabelle *instrument-equalizer-alist* in der Datei '`../scm/midi.scm`' entnehmen.

Dieser grundlegende Equalizer kann ersetzt werden, indem die Funktion `instrumentEqualizer` im `Score`-Kontext auf eine neue Scheme-Funktion gesetzt wird, die MIDI-Instrumentbezeichnungen als einziges Argument akzeptiert und ein Zahlenpaar ausgibt, das den Höchst- und Mindestwert für die Lautstärke des entsprechenden Instruments darstellt. Die Ersetzung der Standardfunktion wird auf gleiche Weise vorgenommen, wie es schon für die `dynamicAbsoluteVolumeFunction` zu Beginn dieses Abschnittes gezeigt wurde. Der Standard-Equalizer, *default-instrument-equalizer* in der Datei '`../scm/midi.scm`' zeigt, wie solche eine Funktion erstellt werden kann.

Das folgende Beispiel definiert für die Flöte und Klarinette relative Lautstärkewerte, die denen des vorigen Beispiels entsprechen.
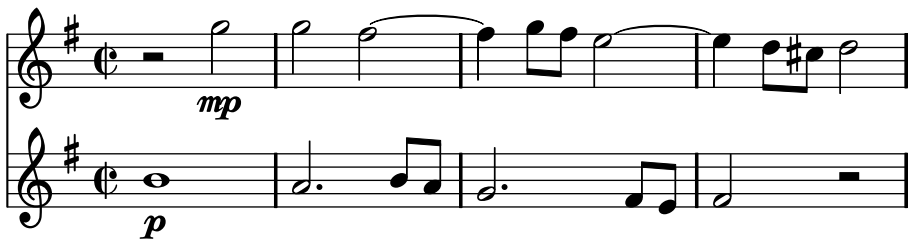
```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(
      ("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
```

```
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = #"flute"
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
    }
  }
}
```



3.5.6 Schlagzeug in MIDI

Schlagzeuginstrumente werden üblicherweise in einem `DrumStaff`-Kontext notiert. Aus diese Weise werden sie korrekt in den MIDI-Kanal 10 ausgegeben. Eine Schlagzeuge mit diskreten Tonhöhen, wie Xylophon, Marima, Vibraphone, Pauken usw. werden wie „normale“ Instrumente in einem `Staff`-Kontext notiert. Nur so lässt sich auch hier eine richtige MIDI-Ausgabe erreichen.

Einige Instrumente, die keine diskreten Tonhöhen haben, können nicht über den MIDI-Kanal 10 erreicht werden und müssen deshalb in einem normalen `Staff`-Kontext mit passenden normalen Tonhöhen notiert werden. Es handelt sich um `melodic tom`, `taiko drum`, `synth drum` u. A.

Viele Schlagzeuginstrumente sind nicht in den MIDI-Standard aufgenommen, z. B. Kastagnetten. Die einfachste Methode, derartige Instrumente zu ersetzen, ist, einen Klang auszuwählen, der ihnen halbwegs ähnlich kommt.

Bekannte Probleme und Warnungen

Weil der MIDI-Standard keine Peitschenschläge kennt, wird ein Schlagstock (sidestick) für diesen Zweck eingesetzt.

4 Abstände

Das finale Layout der Seite wird von drei Faktoren bestimmt: dem Layout der Seite, den Zeilenumbrüchen und der Platzverteilung. Jeder Faktor beeinflusst auch die anderen mit. Die Wahl der Platzverteilung entscheidet, wie eng die Notensysteme gesetzt werden. Das wiederum hat Einfluss auf die gewählten Zeilenumbrüche und letztendlich also auch darauf, wieviele Seiten ein Stück beansprucht.

Die Verteilung der Musik auf der Seite geschieht grob gesagt in vier Schritten. Zuerst werden flexible Entfernungen („springs“) gewählt, die auf den Notendauern basieren. Alle möglichen Zeilenumbrüche werden getestet und ein „Schlechtigkeitsscore“ für die Umbrüche erstellt. Danach wird die mögliche Höhe eines Systems ermittelt und schließlich wird eine bestimmte Kombination aus Seiten- und Zeilenumbruch ausgewählt, sodass weder die horizontale noch die vertikale Platzverteilung zu eng oder zu weit gesetzt wird.

Einstellungen, die das Layout beeinflussen, können in zwei Umgebungen gesetzt werden: in der `\paper {...}`- und der `\layout {...}`-Umgebung. Die `\paper`-Umgebung enthält Einstellungen des Seitenlayouts, die für alle Partituren innerhalb eines `\book` die gleichen sein sollen, wie etwa Papierhöhe oder ob Seitenzahlen ausgegeben werden sollen. Siehe Abschnitt 4.1 [Seitenlayout], Seite 461. Die `\layout`-Umbegung enthält Layouteinstellungen der Partitur selber, wie etwa die Zahl der Systeme oder den Platz zwischen Systemgruppen usw. Siehe Abschnitt 4.2 [Partiturlayout], Seite 471.

4.1 Seitenlayout

Dieser Abschnitt behandelt Seitenlayout-Optionen innerhalb der `\paper`-Umgebung.

4.1.1 Die \paper-Umgebung

Die `\paper`-Umgebung kann innerhalb einer `\book`-, nicht aber innerhalb einer `\score`-Umgebung vorkommen. Einstellungen in `\paper` wirken sich auf das gesamte Buch aus, welches viele einzelne Partituren beinhalten kann. Einstellungen, die in der `\paper`-Umgebung vorkommen können, beinhalten:

- die `set-paper-size`-Scheme-Funktion,
- `\paper`-Variablen, die zum Verändern des Seitenlayouts eingesetzt werden und
- Beschriftungsdefinitionen, mit denen das Layout von Kopf- und Fußleisten sowie Titeln beeinflusst wird.

Die `set-paper-size`-Funktion wird im nächsten Abschnitt behandelt: Abschnitt 4.1.2 [Papierformat und automatische Skalierung], Seite 462. Die `\paper`-Variablen, die das Seitenlayout beeinflussen, werden in späteren Abschnitten behandelt. Die Beschriftungsdefinitionen für Kopf- und Fußzeilen sowie Titeln werden behandelt in Abschnitt 3.2.2 [Eigene Kopf und Fußzeilen sowie Titel], Seite 436.

Die meisten `\paper`-Variablen funktionieren nur innerhalb der `\paper`-Umgebung. Die wenigen, die auch in der `\layout`-Umgebung funktionieren, finden sich in Abschnitt 4.2.1 [Die \layout-Umgebung], Seite 471.

Außer wenn anders angegeben, werden alle `\paper`-Variablen, die Abständen auf der Seite antsprechen, in Millimetern gemessen, es sei denn, eine andere Maßeinheit ist definiert. Beispielsweise wird mit folgender Definition der obere Rand (top-margin) 10 mm breit definiert:

```
\paper {
  top-margin = 10
}
```

Damit etwa `0.5` Zoll benutzt werden, muss `\in` dem Maß nachgestellt werden:

```
\paper {
  top-margin = 0.5\in
}
```

Mögliche Maßeinheiten sind `\mm`, `\cm`, `\in` und `\pt`. Diese Maßeinheiten sind einfach Werte, um von Millimetern zu Konvertieren, sie sind in '`ly/paper-defaults-init.ly`' definiert. Um Missverständnisse zu vermeiden, wird normalerweise `\mm` geschrieben, auch wenn es eigentlich nicht notwendig wäre.

Man kann die `\paper`-Werte auch mit Scheme definieren. Die Scheme-Entsprechung der obigen Definition ist:

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

Siehe auch

Notationsreferenz Abschnitt 4.1.2 [Papierformat und automatische Skalierung], Seite 462, Abschnitt 3.2.2 [Eigene Kopf und Fußzeilen sowie Titel], Seite 436, Abschnitt 4.2.1 [Die \layout-Umgebung], Seite 471.

Installierte Dateien: '`ly/paper-defaults-init.ly`'.

4.1.2 Papierformat und automatische Skalierung

Das Papierformat einstellen

Zwei Funktionen ermöglichen es, die Papiergröße zu ändern: `set-default-paper-size` und `set-paper-size`. `set-default-paper-size` muss auf der obersten Ebene in der Quelldatei gesetzt werden, `set-paper-size` hingegen muss sich in einer `\paper`-Umgebung befinden:

```
#(set-default-paper-size "a4")
\paper {
  #(set-paper-size "a4")
}
```

Auf oberster Ebene kann `set-default-paper-size` überall vor der ersten `\paper`-Umgebung aufgerufen werden. Innerhalb einer `\paper`-Umgebung ist der beste Platz für `set-paper-size` gleich am Anfang, über der Liste der Variablen-Definitionen. Der Grund dafür wird behandelt in [Automatische Skalierung auf ein Papierformat], Seite 463

`set-default-paper-size` bestimmt die Größe aller Seiten, während `set-paper-size` nur die Seitengröße für die Seiten definiert, auf die sich die aktuelle `\paper`-Umgebung bezieht. Wenn die `\paper`-Umgebung gleich am Anfang der Datei steht, bezieht sich die Papiergröße auf alle Seiten, wenn sie aber innerhalb einer `\book`-Umgebung definiert wird, nur auf die Seiten innerhalb dieses Buches.

Die normalen Papierformate sind definiert, u.A. `a4`, `letter`, `legal` und `11x17` (auch als Tabloit bekannt). Sehr viel mehr Formate sind unterstützt. Einzelheiten finden sich in der Datei '`scm/paper.scm`' in der Definition von `paper-alist`.

Achtung: Das Standardformat ist `a4`.

Weitere Papierformate können hinzugefügt werden, indem die Definition von `paper-alist` in der Datei '`scm/paper.scm`' verändert wird. Derartige Änderungen werden jedoch bei einer Aktualisierung des Programmes überschrieben.

Wenn das Symbol `'landscape` als Argument an die Funktion `set-default-paper-size` gehängt wird, werden die Seiten um 90 ° gedreht und die Notensysteme entsprechend breiter gesetzt.

```
#(set-default-paper-size "a6" 'landscape)
```

Siehe auch

Notationsreferenz: [Automatische Skalierung auf ein Papierformat], Seite 463.

Installierte Dateien: ‘`scm/paper.scm`’.

Automatische Skalierung auf ein Papierformat

Wenn das Papierformat mit einer der Scheme-Funktionen (`set-default-paper-size` oder `set-paper-size`) geändert wird, werden die Werte einiger `\paper`-Variablen automatisch an die neue Größe angepasst. Um die automatische Skalierung für eine bestimmte Variable zu umgehen, kann die Variable definiert werden, nachdem man das Papierformat angegeben hat. Es sollte beachtet werden, dass die automatische Anpassung nicht ausgelöst wird, wenn man nur die `paper-height` oder `paper-width`-Variablen verändert, obwohl `paper-width` andere Werte beeinflussen kann (das muss von der automatischen Skalierung unterschieden werden und wird unten behandelt). Die Funktionen `set-default-paper-size` und `set-paper-size` werden behandelt in [Das Papierformat einstellen], Seite 462.

Die vertikalen Dimensionen, die durch die automatische Skalierung verändert werden sind: `top-margin` und `bottom-margin` (siehe [Vertikale \paper-Variablen mit festen Abständen], Seite 463). Die horizontalen Dimensionen, die durch die automatische Skalierung verändert werden, sind `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent` und `short-indent` (siehe [\paper-Variablen für horizontale Abstände], Seite 466).

Die Standardwerte für diese Dimensionen sind in ‘`ly/paper-defaults-init.ly`’ definiert, wobei interne Variablen mit den Bezeichnungen `top-margin-default`, `bottom-margin-default`, usw. benutzt werden. Das sind die Werte für die Standardpapiergröße `a4`. Zum Vergleich: `a4` hat Werte von `297\mm` für `paper-height` und `210\mm` für `paper-width`.

Siehe auch

Notationsreferenz: [Vertikale \paper-Variablen mit festen Abständen], Seite 463, [\paper-Variablen für horizontale Abstände], Seite 466.

Installierte Dateien: ‘`ly/paper-defaults-init.ly`’, ‘`scm/paper.scm`’.

Vertikale \paper-Variablen mit festen Abständen

Achtung: Einige `\paper`-Dimensionen werden automatisch nach Papierformat skaliert, was zu ungewolltem Verhalten führen kann. Siehe [Automatische Skalierung auf ein Papierformat], Seite 463.

Standardwerte (vor der Skalierung) sind definiert in ‘`ly/paper-defaults-init.ly`’.

`paper-height`
: Die Höhe der Seite, standardmäßig nicht definiert. Die automatische Skalierung einiger vertikalen Dimensionen wird hiervon nicht betroffen.

`top-margin`
: Der Rand zwischen dem oberen Ende der Seite und dem oberen Ende des bedruckbaren Bereichs. Wenn das Papierformat verändert wurde, wird der Standardwert dieser Dimension entsprechend skaliert.

`bottom-margin`
: Der Rand zwischen dem unteren Ende der Seite und dem unteren Ende des bedruckbaren Bereichs. Wenn das Papierformat verändert wurde, wird der Standardwert dieser Dimension entsprechend skaliert.

`ragged-bottom`
: Wenn auf wahr gesetzt, werden die Systeme nicht vertikal bis zum unteren Seitenrand verteilt. Sollte auf wahr gesetzt sein für Stücke, die nur ein bis zwei Notensystemgruppen pro Seite haben, etwa Orchesterpartituren.

`ragged-last-bottom`
: Wenn auf falsch gesetzt, werden die Systeme vertikal auf der letzten Seite verteilt. Bei Stücken, die grob zwei oder mehr Seiten füllen, sollten es auf wahr gesetzt werden. Hiermit wird auch die letzte Seite von Teilen eines `\book`, die mit `\bookpart` erstellt sind, beeinflusst.

Siehe auch

Notationsreferenz: [Automatische Skalierung auf ein Papierformat], Seite 463.

Installierte Dateien: '`ly/paper-defaults-init.ly`'.

Schnipsel: Abschnitt "Spacing" in *Schnipsel*.

Bekannte Probleme und Warnungen

Die Titel (aus der `\header`-Umgebung) werden als Systemgruppe behandelt, sodass `ragged-bottom` und `ragged-last-bottom` auch zusätzlichen Abstand zwischen den Titel und dem ersten System einer Paritur einfügt.

Vertikale \paper-Variablen mit flexiblen Abständen

In den meisten Fällen bietet es sich an, dass die vertikalen Abstände zwischen bestimmten Objekten (wie Ränder, Titel, Notensystemgruppen und einzelne Partituren) flexibel gehalten werden, sodass sie je nach Situation gedehnt oder komprimiert werden können. Es gibt eine Anzahl von Variablen für die `\paper`-Umgebung, mit denen man das Dehnungsverhalten dieser Dimensionen beeinflussen kann. Sie finden sich unten aufgelistet.

Dabei sollte beachtet werden, dass die Variablen, die in diesem Abschnitt behandelt werden, nicht die Platzierung und das Dehnungsverhalten von Notensystemen innerhalb der einzelnen Systemgruppen behandelt. Die Dehnung zwischen Notensystemen wird mit Grob-Eigenschaften kontrolliert, deren Einstellungen normalerweise innerhalb der `\score`-Umgebung vorgenommen werden, und nicht innerhalb der `\paper`-Umgebung. Siehe auch Abschnitt 4.4.1 [Flexible vertikale Abstände in Systemgruppen], Seite 482.

Struktur der Alisten für flexible vertikale Abstände

Jede der flexiblen vertikalen Abstandsvariablen ist eine Aliste (eine assoziative Liste), die vier *Schlüssel* (engl. key) enthält:

- `basic-distance` (Grund-Abstand) – der vertikale Abstand, gemessen in Systemzwischenräumen, zwischen den *Referenzpunkten* zweier Elemente, wenn keine Zusammenstöße vorkommen würden und keine Dehnung oder Kompression stattfindet. Der Referenzpunkt einer (Titel-)Beschriftung (auf höchster Ebene) ist sein höchster Punkt und der Referenzpunkt einer Systemgruppe ist der vertikale Mittelpunkt des nächsten `StaffSymbol` – sogar wenn eine Nicht-Notensystemzeile (wie etwa ein `Lyrics`-Kontext) dazwischen kommt. Werte für `basic-distance`, die weniger als entweder `padding` oder `minimum-distance` sind, haben keine Bedeutung, weil der sich daraus ergebende Abstand niemals weniger als entweder `padding` oder `minimum-distance` ergibt.
- `minimum-distance` (minimaler Abstand) – der kleinste erlaubte vertikale Abstand, gemessen in Systemzwischenräumen, zwischen den Referenzpunkten der zwei Elemente, wenn Kompression stattfindet. Werte für `minimum-distance`, die geringer als `padding` sind, haben keine Bedeutung, weil der sich daraus ergebende Abstand niemals weniger als `padding` ergibt.

- `padding` (Verschiebung) – der minimal benötigte vertikale blanke Freiraum zwischen den Bounding-Boxen (oder Skyline) der zwei Objekten, gemessen in Notenlinienabständen.
- `stretchability` (Dehnbarkeit) – ein einheitsloses Maß der Leichtigkeit, mit der sich die Dimension dehnen lässt (ohne dass Zusammenstöße auftreten). Wenn es null ist, wird der Abstand nicht gedehnt (außer ein Zusammenstoß würde auftreten), wenn es positiv ist, hängt die Wichtigkeit der Dehnbarkeit eines bestimmten Objekts nur noch von seiner Beziehung zu dem Wert des anderen Objekts ab. Beispielsweise wennn eine Dimension die doppelte Dehnbarkeit als die andere hat, wird sie auch zweimal so einfach gedehnt. Werte sollten nicht-negativ und reale Zahlen sein. Der Wert `+inf.0` ruft einen `programming_error` hervor und wird ignoriert, aber `1.0e7` kann für einen so gut wie unendlich dehnbaren Abstand eingesetzt werden. Wenn der Wert nicht gesetzt wird, ist der Standardwert der von `basic-distance`. Die Wahrscheinlichkeit einer Dimension, sich zu verkleinern, kann man nicht direkt beeinflussen, sondern sie ergibt sich aus (`space` − `minimum-distance`).

Wenn eine Seite einen nicht ausgeglichenen unteren Rand hat, ist der resultierende Abstand der größte von:

- `basic-distance`,
- `minimum-distance` und
- `padding` plus der kleinste nötige Abstand, um Zusammenstöße zu vermeiden.

Spezifische Methoden, um Alisten zu verändern, werden behandelt in Abschnitt 5.3.6 [Alisten verändern], Seite 534. Das folgende Beispiel demonstriert beide Arten, wie diese Alisten verändert werden können. Der erste Aufruf verändert nur einen Schlüsselwert einzeln, während der zweite die Variable vollständig neu definiert:

```
\paper {
  system-system-spacing #'basic-distance = #8
  score-system-spacing =
    #'((basic-distance . 12)
       (minimum-distance . 6)
       (padding . 1)
       (stretchability . 12))
}
```

Liste der flexiblen vertikalen Abstandsvariblen in \paper

Die Bezeichnungen dieser Variablen entsprechen dem Format *`obere`*`-`*`untere`*`-platzierung`, wobei *`obere`* und *`untere`* die zu platzierenden Elemente darstellen. Jeder Abstand wird zwischen den Referenzpunkten der beiden Elemente gemessen (siehe Beschreibung der Alistenstruktur oben). In diesen Variablenbezeichnungen bedeutet ‚`markup`‘ (Beschriftung) sowohl *Titelbeschriftungen* (`bookTitleMarkup` oder `scoreTitleMarkup`) als auch *Beschriftungen auf höchster Ebene* (siehe Abschnitt 3.1.5 [Die Dateistruktur], Seite 431). Alle Entfernungen werden in Systemzwischenräumen gemessen.

Standardwerte sind in ‘`ly/paper-defaults-init.ly`’ definiert.

`markup-system-spacing`
: der Abstand zwischen einer (Titel-)Beschriftung (auf höchster Ebene) und der darauf folgenden Systemgruppe.

`score-markup-spacing`
: der Abstand zwischen dem letzten System einer Partitur und der darauf folgenden (Titel-)Beschriftung (auf höchster Ebene).

`score-system-spacing`
: der Abstand zwischen dem letzten System einer Partitur und dem ersten System der folgenden Partitur, wenn keine (Titel-)Beschriftung (auf höchster Ebene) dazwischen vorkommt.

`system-system-spacing`
: der Abstand zwischen zwei Systemgruppen der selben Partitur.

`markup-markup-spacing`
: der Abstand zwischen zwei (Titel-)Beschriftungen (auf höchster Ebene).

`last-bottom-spacing`
: der Abstand vom letzten System oder Beschriftung auf höchster Ebene auf einer Seite zum unteren Rand des bedruckbaren Bereichs (also bis zum Anfang des unteren Randes).

`top-system-spacing`
: der Abstand zwischen dem oberen Rand des bedruckbaren Bereichs (also dem Ende des oberen Rands) und dem ersten System auf der Seite, wenn keine (Titel-)Beschriftung (auf höchster Ebene) dazwischen kommt.

`top-markup-spacing`
: der Abstand vom oberen Rand des bedruckbaren Bereichs (also dem Ende des oberen Randes) zur ersten (Titel-)Beschriftung (auf höchster Ebene) auf einer Seite, wenn keine Systemgruppe dazwischen kommt.

Siehe auch

Notationsreferenz: Abschnitt 4.4.1 [Flexible vertikale Abstände in Systemgruppen], Seite 482.

Installierte Dateien: ‘`ly/paper-defaults-init.ly`’.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

\paper-Variablen für horizontale Abstände

Achtung: Einige `\paper`-Dimensionen werden automatische entsprechend dem Papierformat skaliert und können deshalb ungewollte Resultate haben. Siehe [Automatische Skalierung auf ein Papierformat], Seite 463.

\paper-Variablen für Breite und Ränder

Standardwerte (vor der Skalierung), die hier nicht aufgelistet sind, finden sich in ‘`ly/paper-defaults-init.ly`’.

`paper-width`
: Die Breide der Seite, standardmäßig nicht definiert. Während `paper-width` keine Auswirkungen auf die automatische Skalierung einiger horizontaler Dimensionen hat, beeinflusst es dennoch die `line-width`-Variable. Wenn sowohl `paper-width` als auch `line-width` definiert sind, dann werden auch `left-margin` und `right-margin` aktualisiert. Siehe auch `check-consistency`.

`line-width`
: Die horizontale Ausdehnung der Notenlinien in nicht-eingerückten, Systemen mit Ausgleich zum rechten Rand, entspricht `(paper-width − left-margin − right-margin)` wenn nicht definiert. Wenn `line-width` definiert ist und sowohl `left-margin` als auch `right-margin` nicht definiert sind, dann werden die Ränder aktualisiert, sodass die Systeme mittig

auf der Seite zentriert werden. Siehe auch `check-consistency`. Diese Variable kann auch in der `\layout`-Umgebung definiert werden.

`left-margin`
: Der Rand zwischen der linken Papierkante und dem Beginn der Systeme ohne Einrückungen. Wenn das Papierformat verändert wird, wird auch der Standardwert dieser Dimension entsprechend skaliert. Wenn `left-margin` nicht definiert ist und sowohl `line-width` als auch `right-margin` definiert sind, dann wird `left-margin` auf den Wert `(paper-width − line-width − right-margin)` gesetzt. Wenn nur `line-width` definiert ist, dann werden beide Ränder auf den Wert `((paper-width − line-width) / 2)` gesetzt und die Systeme demzufolge auf der Seite zentriert. Siehe auch `check-consistency`.

`right-margin`
: Der Rand zwischen der rechten Papierkante und dem Ende der Systeme mit Randausgleich („Blocksatz“). Wenn das Papierformat geändert wird, wird auch der Standardwert dieser Dimension entsprechend skaliert. Wenn `right-margin` nicht definiert ist und sowohl `line-width` als auch `left-margin` definiert sind, dann wird `right-margin` auf den Wert `(paper-width − line-width − left-margin)` gesetzt. Wenn nur `line-width` definiert ist, dann werden beide Ränder auf den Wert `((paper-width − line-width) / 2)` gesetzt und die Systeme demzufolge auf der Seite zentriert. Siehe auch `check-consistency`.

`check-consistency`
: Wenn wahr, wird eine Warnung ausgegeben, sollten `left-margin`, `line-width` und `right-margin` zusammen nicht exakt den Wert von `paper-width` ergeben, und die Werte (außer `paper-width`) mit ihren Standardwerten belegt (wenn nötig auf das entsprechende Papierformat skaliert). Wenn falsch werden derartige Inkonsistenzen ignoriert und die Systeme dürfen auch über den Seitenrand hinausragen.

`ragged-right`
: Wenn wahr, werden Notensysteme nicht über die gesamte Zeilenbreite gestreckt, sondern sie enden horizontal entsprechend den enthaltenen Noten. Standard: `#t` (wahr) für Partituren mit einem System und `#f` (falsch) für Partituren mit zwei oder mehr Systemen. Diese Variable kann auch in der `\layout`-Umgebung definiert werden.

`ragged-last`
: Wenn wahr, wird das letzte Notensystem einer Partitur nicht über die gesamte Zeilenbreite gestreckt, sondern es endet horizontal entsprechend den enthaltenen Noten. Standard: `#f` (falsch). Diese Variable kann auch in der `\layout`-Umgebung definiert werden.

Siehe auch

Notationsreferenz: [Automatische Skalierung auf ein Papierformat], Seite 463.

Installierte Dateien: ‘`ly/paper-defaults-init.ly`’.

\paper-Variablen für zweiseitigen Satz

Standardwerte (vor der Skalierung) sind definiert in ‘`ly/paper-defaults-init.ly`’.

`two-sided`
: Wenn auf wahr (`##t`) gesetzt, werden `inner-margin`, `outer-margin` und `binding-offset` zusammen benutzt, um die Ränder der Seite in Abhängigkeit von einer geraden oder ungeraden Seitennummer zu errechnen. Damit werden die Werte von `left-margin` und `right-margin` überschrieben. Standard: `##f`.

`inner-margin`
: Der Rand, den alle Seiten auf der Innenseite haben, wenn sie Teil eines Buches (`\book`) sind. Wenn das Papierformat verändert wird, wird der Standardwert dieser Dimension entsprechend skaliert. Funktioniert nur, wenn `two-sided` wahr ist.

`outer-margin`
: Der Rand, den alle Seiten auf der Außenseite haben, wenn sie Teil eines Buches sind. Wenn das Papierformat verändert wird, wird der Standardwert dieser Dimension entsprechend skaliert. Funktioniert nur, wenn `two-sided` wahr ist.

`binding-offset`
: Der Wert, um welchen `inner-margin` erhöht wird, um sicherzugehen, dass nichts in der Bindung verschwindet. Wenn das Papierformat verändert wird, wird der Standardwert dieser Dimension entsprechend skaliert. Funktioniert nur, wenn `two-sided` wahr ist.

Siehe auch

Notationsreferenz: [Automatische Skalierung auf ein Papierformat], Seite 463.

Installierte Dateien: '`ly/paper-defaults-init.ly`'.

\paper-Variablen für Verschiebungen und Einrückungen

Standardwerte (vor der Skalierung), die hier nicht aufgeführt sind, sind definiert in '`ly/paper-defaults-init.ly`'.

`horizontal-shift`
: Der Wert, um den alle Systeme (und auch Überschriften und Systemtrenner) nach rechts verschoben werden. Standard: `0.0\mm`.

`indent`
: Der Einzug für das erste System einer Partitur. Wenn das Papierformat verändert wird, wird der Standardwert dieser Dimension entsprechend skaliert. Diese Variable kann auch in der `\layout`-Umgebung definiert werden.

`short-indent`
: Der Einzug für alle Systeme einer Partitur ausschließlich das erste System. Wenn das Papierformat verändert wird, wird der Standardwert dieser Dimension entsprechend skaliert. Diese Variable kann auch in der `\layout`-Umgebung definiert werden.

Siehe auch

Notationsreferenz: [Automatische Skalierung auf ein Papierformat], Seite 463.

Installierte Dateien: '`ly/paper-defaults-init.ly`'.

Schnipsel: Abschnitt "Spacing" in *Schnipsel*.

4.1.3 Andere \paper-Variablen

\paper-Variablen für den Zeilenumbruch

`max-systems-per-page`
: Die maximale Anzahl an Notensystemgruppen, die auf einer Seite gesetzt werden. Das wird zur Zeit nur von dem `ly:optimal-breaking`-Algorithmus unterstützt. Standard: nicht gesetzt.

`min-systems-per-page`
: Die minimale Anzahl an Notensystemgruppen, die auf einer Seite gesetzt werden. Das kann dazu führen, dass Seiten zu dicht gefüllt werden, wenn der Wert zu

groß gewählt wird. Die Option ist zur Zeit nur von dem `ly:optimal-breaking`-Algorithmus unterstützt. Standard: nicht gesetzt.

`systems-per-page`
: Die Anzahl an Systemen, die auf jede Seite gesetzt werden sollen. Diese Option wird zur Zeit nur von dem `ly:optimal-breaking`-Algorithmus unterstützt. Standard: nicht gesetzt.

`system-count`
: Die Anzahl der Systeme, auf denen eine Partitur gesetzt werden soll. Standard: nicht gesetzt. Diese Variablen kann auch in der `\layout`-Umgebung definiert werden.

Siehe auch

Notationsreferenz: Abschnitt 4.3.1 [Zeilenumbrüche], Seite 473.

\paper-Variablen für den Seitenumbruch

Standardwerte, die hier nicht aufgelistet sind, finden sich in '`ly/paper-defaults-init.ly`'

`blank-after-score-page-force`
: Die Strafpunke, die erteilt werden, wenn eine leere Seite nach einer Partitur und vor der nächsten vorkommt. Der Standardwert hiervon ist kleiner als `blank-page-force`, sodass leere Seiten nach einer Partitur leeren Seiten innerhalb einer Partitur vorgezogen werden.

`blank-last-page-force`
: Die Strafpunkte, wenn eine Partitur auf einer ungeraden Seite beendet wird. Standard: `0`.

`blank-page-force`
: Die Strafpunkte, wenn eine leere Seite mitten in einer Partitur auftritt. Das wird nicht benutzt von `ly:optimal-breaking`, weil hiermit niemals leere Seiten mitten in einer Partitur zugelassen werden.

`page-breaking`
: Der Algorithmus, der für Seitenumbrüche eingesetzt wird. Mögliche Algorithmen sind: `ly:minimal-breaking` (minimale Umbrüche), `ly:page-turn-breaking` (Umbrüche an guten Stellen zum Umblättern) und `ly:optimal-breaking`.

`page-breaking-system-system-spacing`
: Überlistet die Seitenumbruchfunktion, indem ihr ein anderer Wert für `system-system-spacing` mitgeteilt wird, als in Wirklichkeit eingestellt ist. Wenn beispielsweise `page-breaking-system-system-spacing #'padding` auf einen deutlich größeren Wert als `system-system-spacing #'padding` gesetzt wird, setzt die Seitenumbruchsfunktion weniger Systeme auf eine Seite. Standard: nicht gesetzt.

`page-count`
: Die Zahl der Seiten, die für eine Partitur benutzt werden sollen. Standard: nicht gesetzt.

Siehe auch

Notationsreferenz: Abschnitt 4.3.2 [Seitenumbrüche], Seite 475, Abschnitt 4.3.3 [Optimale Seitenumbrüche], Seite 476, Abschnitt 4.3.4 [Optimale Umbrüche zum Blättern], Seite 476, Abschnitt 4.3.5 [Minimale Seitenumbrüche], Seite 477.

Installierte Dateien: '`ly/paper-defaults-init.ly`'.

\paper-Variablen für Seitenzahlen

Standardwerte, die hier nicht aufgelistet sind, finden sich in ‘ly/paper-defaults-init.ly’

auto-first-page-number
: Der Seitenumbruchsalgorithmus wird davon beeinflusst, ob die erste Seitenzahl gerade oder ungerade ist. Wenn die Variable auf wahr gesetzt wird, entscheidet der Seitenumbruchsalgorithmus selber, ob die Noten auf einer geraden oder ungeraden Seite beginnen sollen. Das hat dann zur Folge, dass die erste Seite entweder bleibt wie sie ist oder um eins erhöht wird. Standard: #f.

first-page-number
: Der Wert der Seitenzahl auf der ersten Seite.

print-first-page-number
: Wenn wahr, wird auch auf der ersten Seite die Seitenzahl ausgegeben. Standard: #f.

print-page-number
: Wenn falsch, werden Seitenzahlen nicht ausgegeben.

Siehe auch

Installierte Dateien: ‘ly/paper-defaults-init.ly’.

Bekannte Probleme und Warnungen

Ungrade Seitenzahlen befinden sich immer auf der rechten Seite. Wenn Sie die Noten auf Seite 1 beginnen lassen wollen, müssen Sie eine leere Seite nach dem Deckblatt einfügen, damit die Noten auf der rechten Seite mit Seite 1 beginnen.

Verschiedene \paper-Variablen

page-spacing-weight
: Die relative Gewichtung von (vertikalem) Abstand auf der Seite und (horizontalem) Abstand innerhalb der Zeilen. Hohe Werte gewichten die vertikalen Abstände mehr. Standard: #10.

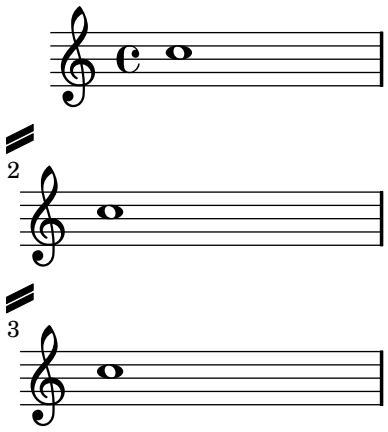
print-all-headers
: Wenn wahr, werden alle Einträge des Titelfeldes (\header-Umgebung) für jede Partitur (\score) ausgegeben. Normalerweise wird nur die Satzbezeichnung und die Opuszahl (piece und opus) ausgegeben. Standard: ##f.

system-separator-markup
: Ein Beschriftungsobjekt, das zwischen zwei Systeme gesetzt wird. Das wird oft in Orchesterpartituren eingesetzt. Standard: nicht gesetzt. Der Beschriftungsbefehl \slashSeparator, definiert in ‘ly/titling-init.ly’, kann für einen Trenner benutzt werden, etwa so:

```
#(set-default-paper-size "a8")

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative c'' { c1 \break c1 \break c1 }
```

```
  }
}
```



Siehe auch

Installierte Dateien: ‘`ly/titling-init.ly`’.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Bekannte Probleme und Warnungen

Die Standard-Kopfzeilendefinition setzt die Seitenzahl und das `instrument`-Feld aus der `\header`-Umgebung in eine Zeile.

4.2 Partiturlayout

4.2.1 Die ayout-Umgebung

Während die `\paper`-Umgebung Einstellungen für die Formatierung der Seiten eines gesamten Dokuments enthalten, enthält die `\layout`-Umgebung Einstellungen für einzelne Partituren. Um Layoutoptionen für Partituren global einzustellen, müssen sie in einer `\layout`-Umgebung gesetzt werden, die sich auf höchster Ebene in der Datei befindet. Um sie für einzelne Partituren festzulegen, muss die `\layout`-Umgebung innerhalb der `\score`-Umgebung nach den Noten eingetraten werden. Einstellungen, die in einer `\layout`-Umgebung vorkommen können, beinhalten:

- die `layout-set-staff-size`-Scheme-Funktion,
- Kontextveränderungen in `\context`-Umgebungen und
- `\paper`-Variablen, die das Aussehen einer Partitur beeinflussen.

Die `layout-set-staff-size`-Funktion wird im nächsten Abschnitt behandelt, Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472. Kontextveränderungen werden in einem eigenen Kapitel behandelt, siehe Abschnitt 5.1.4 [Umgebungs-Plugins verändern], Seite 519 and Abschnitt 5.1.5 [Die Standardeinstellungen von Kontexten ändern], Seite 521. Die `\paper`-Variablen, die innerhalb der `\layout`-Umgebungen erlaubt sind, sind:

- `line-width`, `ragged-right` und `ragged-last` (siehe [\paper-Variablen für Breite und Ränder], Seite 466)
- `indent` und `short-indent` (siehe [\paper-Variablen für Verschiebungen und Einrückungen], Seite 468)
- `system-count` (siehe [\paper-Variablen für den Zeilenumbruch], Seite 468)

Hier ist ein Beispiel für eine `\layout`-Umgebung:

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGrouper #'staff-staff-spacing #'basic-distance = #8
  }
  \context {
    \Voice
    \override TextScript #'padding = #1
    \override Glissando #'thickness = #3
  }
}
```

Siehe auch

Notationsreferenz: Abschnitt 5.1.5 [Die Standardeinstellungen von Kontexten ändern], Seite 521, Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472, Abschnitt 5.1.4 [Umgebungs-Plugins verändern], Seite 519.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.2.2 Die Notensystemgröße einstellen

Die Standardgröße der Notensysteme beträgt 20 Punkte (pt). Das kann auf zwei Arten geändert werden:

Um die Systemgröße global für alle Partituren einer Datei (bzw. einer `\book`-Umgebung) zu verändern, wird `set-global-staff-size` benutzt:

```
#(set-global-staff-size 14)
```

Hiermit wird die Standardhöhe der Notensysteme auf 14 pt gesetzt. Die Schriftarten werden entsprechend verkleinert.

Um die Systemhöhe für jede Partitur einzeln zu verändern, muss

```
\score{
  ...
  \layout {
    #(layout-set-staff-size 15)
  }
}
```

eingesetzt werden.

Die Feta-Schriftart stellt die Noten- und Musiksymbole für acht verschiedene Größen zur Verfügung. Jede Schriftgröße ist einer bestimmten Systemgröße angepasst: für kleinere Schriftgrößen werden die Zeichen etwas schwerer, um mit den ebenfalls dickeren Notenlinien zu harmonieren. Die empfohlenen Notensystemgrößen sind in der Tabelle aufgeführt:

Schriftbezeichnung	Höhe des Systems (pt)	Höhe des Systems (mm)	Benutzung
feta11	11.22	3.9	Taschenparituren
feta13	12.60	4.4	
feta14	14.14	5.0	
feta16	15.87	5.6	
feta18	17.82	6.3	Liederbücher

feta20	20	7.0	Orchesterstimmen
feta23	22.45	7.9	
feta26	25.2	8.9	

Diese Schriftarten sind in allen Größen erhältlich. Die Kontext-Eigenschaft `fontSize` und die Layout-Eigenschaft `staff-space` (in `StaffSymbol`) können benutzt werden, um die Schriftgröße für einzelne Systeme zu verändern. Die Größe von einzelnen Systemen ist relativ zur globalen Systemgröße.

Siehe auch

Notationsreferenz: [Auswahl der Notations-Schriftgröße], Seite 189.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Bekannte Probleme und Warnungen

`layout-set-staff-size` verändert nicht den Abstand zwischen den Notenlinien.
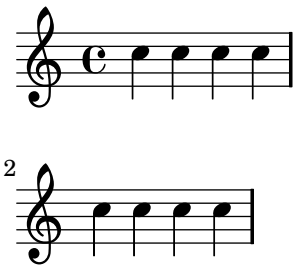
4.3 Umbrüche

4.3.1 Zeilenumbrüche

Zeilenumbrüche werden normalerweise automatisch erstellt. Sie werden so ausgewählt, dass die Zeilen weder gedrängt noch zu weit gespreizt wirken und aufeinander folgende Seiten einen ähnlichen Grauwert haben.
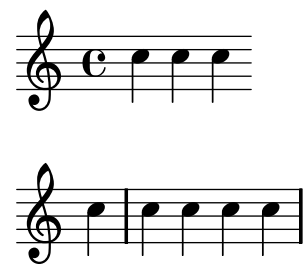
Einen manuellen Zeilenumbruch fügt man mit dem Befehl `\break` ein:

```
c4 c c c | \break
c4 c c c |
```



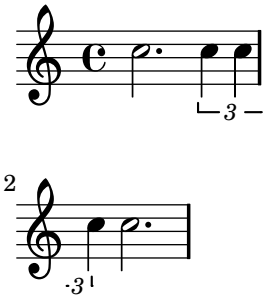
Normalerweise wird ein `\break` in der Mitte eines Takes ignoriert und eine Warnung ausgegeben. Um einen Zeilenumbruch in der Mitte eines Taktes zu erzwingen, können Sie mit `\bar "" ` eine unsichtbare Taktlinie hinzufügen, die dann den Zeilenumbruch erlaubt.

```
c4 c c
\bar "" \break
c |
c4 c c c |
```
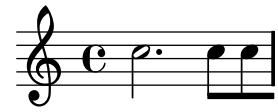


Ein `\break` an einem Taktstrich wird auch ignoriert, wenn der letzte Takt mitten in einer Note endet, wenn etwa eine N-tole in unterschiedlichen Takten beginnt und endet. Damit `\break` auch in derartigen Situationen funktioniert, muss `Forbid_line_break_engraver` aus der `Voice`-Umgebung entfernt werden. Dabei sollte beachtet werden, dass manuell hervorgerufene Umbrüche parallel mit den Noten hinzugefügt werden müssen.
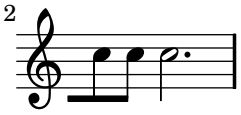
```
\new Voice \with {
  \remove Forbid_line_break_engraver
} \relative c'' {
  <<
    { c2. \times 2/3 { c4 c c } c2. | }
    { s1 | \break s1 | }
  >>
}
```



Genauso werden normalerweise Zeilenumbrüche auch verhindert, wenn Balken über die Taktenden hinausragen. Dieses Verhalten kann verändert werden, indem man `\override Beam #'breakable = ##t` einstellt:

```
\override Beam #'breakable = ##t
c2. c8[ c | \break
c8 c] c2. |
```





Mit dem Befehl `\noBreak` wird ein Zeilenumbruch an dem entsprechenden Taktstrich verboten.

Die grundlegenden Einstellungen, die Einfluss auf die Zeilenlänge haben, sind `indent` (Einzug) und `line-width` (Zeilenbreite). Sie werden in der `\layout`-Umgebung eingestellt. Der erste Befehl bestimmt den Einzug der ersten Zeile, der zweite die Zeilenlänge der weiteren Notenzeilen.

Wenn `ragged-right` eingestellt ist (also in der `\layout`-Umgebung auf den Wert `#t` gesetzt wurde), werden die Systeme linksbündig gesetzt und nicht bis zum rechten Rand hin durchgezogen, sondern den Noten entsprechend gesetzt. Das ist oftmals nützlich für kleine Notenfragmente und um zu überprüfen, wie eng die Noten natürlicherweise gesetzt werden würden.

Die Option `ragged-last` verhält sich ähnlich zu `ragged-right`, aber wirkt sich nur auf die letzte Zeile eines Stückes aus.

```
\layout {
indent = #0
line-width = #150\mm
ragged-last = ##t
}
```

Um Zeilenumbrüche zu erzwingen, die in festgelegten Intervallen stattfinden, kann der Befehl `\break` in Kombination mit unsichtbaren Noten und einer Wiederholung (`\repeat`) eingesetzt werden. Das folgende Beispiel etwa setzt die nächsten 28 Takte (im 4/4-Takt) in Zeilen zu jeweils 4 Takten (die auch nur hier umgebrochen werden):

```
<<
  \repeat unfold 7 {
    s1 \noBreak s1 \noBreak
    s1 \noBreak s1 \break
  }
  { Hier die Noten... }
>>
```

Eine Zeilenumbruchkonfiguration kann auch als eine '`.ly`'-Datei automatisch gespeichert werden. Damit kann die vertikale Ausrichtung während eines zweiten Programmdurchlaufs angepasst werden um die Seiten besser zu füllen. Diese Eigenschaft ist recht neu und kompliziert. Mehr Einzelheiten finden sich in Abschnitt "Spacing" in *Schnipsel*.

Vordefinierte Befehle

`\break`, `\noBreak`.

Siehe auch

Notationsreferenz: [\paper-Variablen für den Zeilenumbruch], Seite 468.

Schnipsel: Abschnitt "Spacing" in *Schnipsel*.

Referenz der Interna: Abschnitt "LineBreakEvent" in *Referenz der Interna*.

4.3.2 Seitenumbrüche

Die Standardseitenumbrüche können verändert werden, indem man die Befehle `\pageBreak` bzw. `\noPageBreak` benutzt. Sie verhalten sich analog zu den Befehlen `\break` und `\noBreak`. Sie sollten an einem Taktstrich notiert werden. Diese Befehle erzwingen bzw. verbieten einen Seitenumbruch. Mit dem `\pageBreak`-Befehl wird natürlich gleichzeitig auch ein Zeilenumbruch erzwungen.

Die `\pageBreak` und `\noPageBreak`-Befehle können auch auf der höchsten Ebene einer Datei benutzt werden, etwa zwischen Partituren und Textbeschriftungen.

Es gibt auch vertikale Gegenstücke zu den Variablen `ragged-right` und `ragged-last`: `ragged-bottom` und `ragged-last-bottom`. Wenn diese Variablen auf `##t` (wahr) gesetzt werden, werden im ersten Fall die Notensysteme auf allen Seiten eng nach oben orientiert gesetzt werden. Im zweiten Fall bezieht sich dies nur auf die letzte Seite. Zu Einzelheiten siehe [Vertikale \paper-Variablen mit festen Abständen], Seite 463.

Seitenumbrüche werden von der `page-breaking`-Funktion errechnet. LilyPond kennt drei Algorithmen um Seitenumbrüche zu errechnen: `ly:optimal-breaking`, `ly:page-turn-breaking` und `ly:minimal-breaking`. Der Standard ist `ly:optimal-breaking`, aber der Wert kann in der `\paper`-Umgebung geändert werden:

```
\paper{
  #(define page-breaking ly:page-turn-breaking)
}
```

Wenn ein Buch (`\book`) viele Parituren und Seiten hat, kann die Seitenaufteilung schwer zu ermitteln sein und viel Zeit und Prozessorlast in Anspruch nehmen. Um den Seitenumbruchsprozess zu vereinfachen, werden `\bookpart`-Umgebungen benutzt, um das Buch in mehrere Teil zu trennen: Die Seitenumbrüche werden seperat für jeden Teil berechnet. Unterschiedliche Seitenumbruchsfunktionen können in unterschiedlichen Buchteilen benutzt werden.

```
\bookpart {
  \header {
    subtitle = "Vorwort"
  }
  \paper {
     %% In einem Abschnitt, der vor allem Text hat,
     %% funktioniert womöglich ly:minimal-breaking besser
     #(define page-breaking ly:minimal-breaking)
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In diesem Abschnitt mit Noten wird
  %% die Standard-Seitenumbruchsfunktion benutzt.
  \header {
    subtitle = "Erster Satz"
  }
  \score { ... }
  ...
}
```

Vordefinierte Befehle

`\pageBreak`, `\noPageBreak`.

Siehe auch

Notationsreferenz: [\paper-Variablen für den Seitenumbruch], Seite 469.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.3.3 Optimale Seitenumbrüche

Die `ly:optimal-breaking`-Funktion ist die Standardmethode für LilyPond, um Seitenumbrüche zu errechnen. Hiermit wird versucht, Seitenumbrüche zu finden, die das Stauchen oder Strecken von Zeilen minimieren, sowohl horizontal als auch vertikal. Anders als die `ly:page-turn-breaking`-Funktion hat diese Methode keine Möglichkeit, Überlegungen zum Umblättern mit einzubeziehen.

Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.3.4 Optimale Umbrüche zum Blättern

Es ist oft nötig, die Seiten so umzubrechen, dass sich eine Pause am Ende jeder zweiten Seite befindet, damit der Musiker es leichter hat, die Seite umzublättern ohne das Spielen zu Unterbrechen. Die `ly:page-turn-breaking`-Funktion versucht, Seitenumbrüche zu finden, die das Stauchen oder Strecken von Zeilen minimieren und gleichzeitig auch noch Seitenumbrüchen an angegebenen Stellen den Vorrang zu geben.

Die Funktion wird in zwei Schritten eingesetzt. Zunächst muss sie in der `\paper`-Umgebung aktiviert werden, wie gezeigt in Abschnitt 4.3.2 [Seitenumbrüche], Seite 475. Dann muss noch angegeben werden, welche Stellen bevorzugt für Seitenumbrüche benutzt werden sollen.

Für diesen zweiten Schritt gibt es zwei Methoden. Am Einfachsten ist es, die möglichen Seitenumbrüche mit dem Befehl `\allowPageTurn` an jeder Stelle manuell anzugeben.

Wenn Ihnen das zu aufwändig ist, können Sie den `Page_turn_engraver` zu einem `Staff`- oder `Voice`-Kontext hinzufügen. Dieser Engraver durchsucht den entsprechenden Kontext nach Stellen ohne Noten. (Es wird also nicht nach Pausen gesucht, sondern nach Stellen ohne Noten. Dieses Verhalten verhindert, dass an polyphonen Stellen umgebrochen wird, wo nur in einer Stimme Pausen vorhanden sind.) Wenn eine derartige Stelle ohne Noten gefunden wird, fügt der Engraver den Befehl `\allowPageTurn` am letzten Taktstrich des Abschnitts ein. Wenn in dem Abschnitt ein besonderer Taktstrich vorkommt (wie etwa ein Doppelstrich), wird der Befehl nach diesem Taktstrich gesetzt.

Der `Page_turn_engraver` liest die Kontexteigenschaft `minimumPageTurnLength` um zu erkennen, wie lang eine Stelle frei von Noten sein muss, damit ein Seitenumbruch in Frage kommt. Der Standardwert hierfür ist `#(ly:make-moment 1 1)`. Wenn Sie Seitenumbrüche zum Umblättern ausschalten wollen, können Sie einen sehr großen Wert angeben.

```
\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % Ein Seitenumbruch zum Umblättern erlaubt
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5 2)
  R1 | % Seitenumbruch nicht erlaubt
  a4 b r2 |
  R1*2 | % Seitenumbruch erlaubt
  a1
}
```

Der `Page_turn_engraver` erkennt Wiederholungen vom Typ `volta`. Ein Seitenumbruch zum Umblättern wird nur zugelassen, wenn vor und nach der Wiederholung genug Zeit ist, um die Seite wieder zurückzublättern. Wenn die Wiederholung sehr kurz ist, kann auch Umblättern verboten werden. Wenn Sie die Kontexteigenschaft `minimumRepeatLengthForPageTurn` definieren, erlaubt der `Page_turn_engraver` nur Umblättern in Wiederholungen, deren Dauer länger als dieser Wert ist.

Die Seitenumblätter-Befehle `\pageTurn`, `\noPageTurn` und `\allowPageTurn` können auch auf oberster Dateiebene benutzt werden, etwa zwischen Partituren und Textabschnitten.

Vordefinierte Befehle

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Bekannte Probleme und Warnungen

In einer Partitur sollte nur ein `Page_turn_engraver` vorkommen. Wenn mehr als einer definiert werden, stören sie sich gegenseitig.

4.3.5 Minimale Seitenumbrüche

Die `ly:minimal-breaking`-Funktion benötigt nur minimale Berechnungen, um die Seitenumbrüche zu bestimmen. Die Seite wird mit möglichst vielen Systemen gefüllt und dann zur

nächsten Seite gewechselt. Die Funktion kann benutzt werden um Partituren mit vielen Seiten zu setzen, wenn die anderen Seitenumbruchsfunktionen zu langsam wären oder zu viel Speicher beanspruchen. Auch für Seiten mit viel Text ist die Funktion geeignet. Sie wird folgendermaßen aktiviert:

```
\paper {
  page-breaking = #ly:minimal-breaking
}
```

Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.3.6 Ausdrückliche Umbrüche

Es kann vorkommen, dass LilyPond direkte `\break` oder `\pageBreak`-Befehl nicht beachtet. Mit folgenden Einstellungen kann dieses Verhalten ausgeschaltet werden:
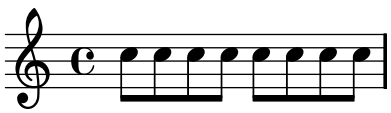
```
\override NonMusicalPaperColumn #'line-break-permission = ##f
\override NonMusicalPaperColumn #'page-break-permission = ##f
```
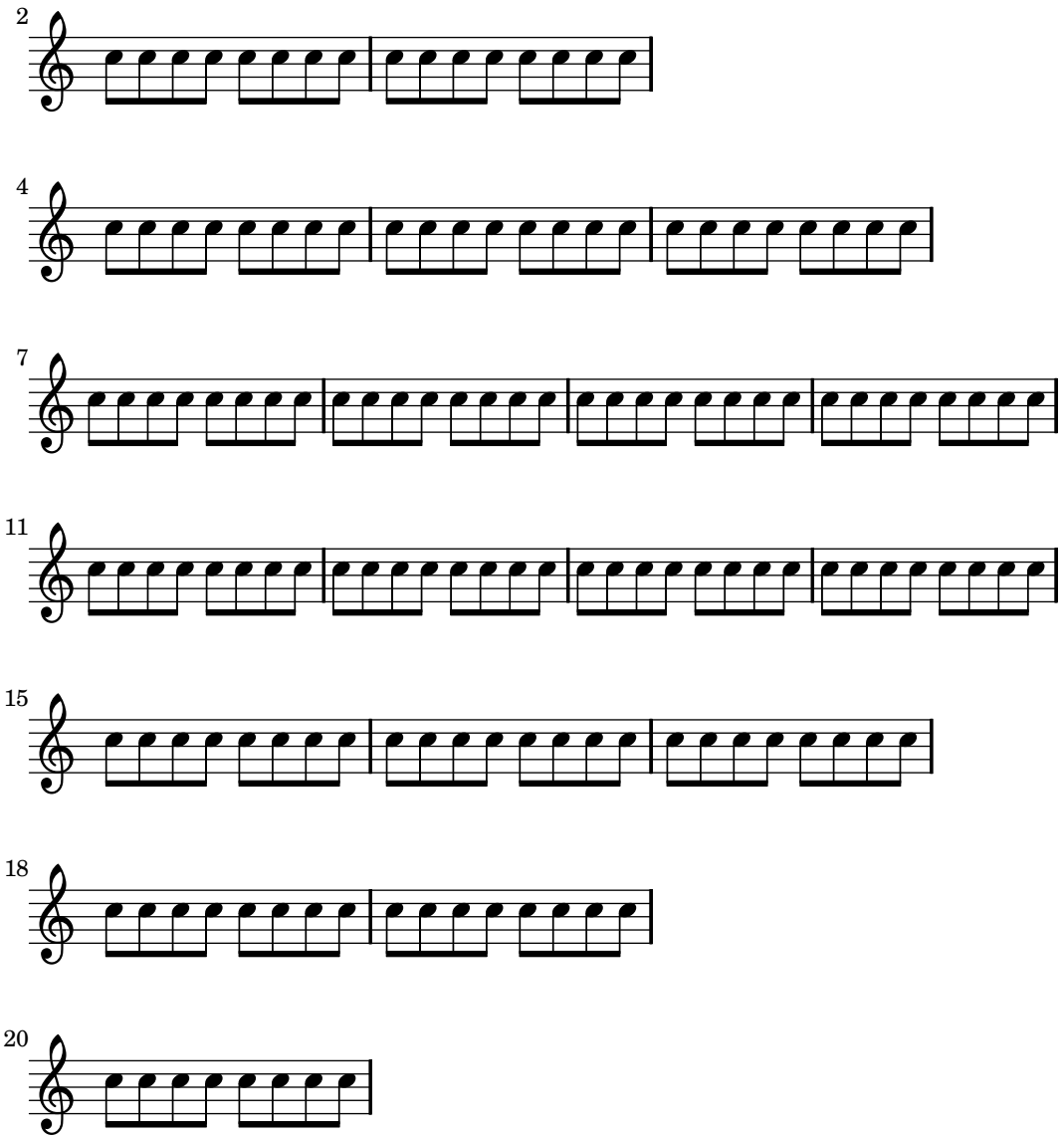
Wenn `line-break-permission` die Einstellung falsch (`##f`) hat, werden Zeilenumbrüche nur an den Befehlen `\break` eingefügt und nirgendwo anders. Wenn `page-break-permission` die Einstellung falsch (`##f`) hat, werden Seitenumbrüche nur an den Befehlen `\pageBreak` eingefügt und nirgendwo anders.

```
\paper {
  indent = #0
  ragged-right = ##t
  ragged-bottom = ##t
}

music = \relative c'' { c8 c c c }

\score {
  \new Staff {
    \repeat unfold 2 { \music } \break
    \repeat unfold 4 { \music } \break
    \repeat unfold 6 { \music } \break
    \repeat unfold 8 { \music } \pageBreak
    \repeat unfold 8 { \music } \break
    \repeat unfold 6 { \music } \break
    \repeat unfold 4 { \music } \break
    \repeat unfold 2 { \music }
  }
  \layout {
    \context {
      \Score
      \override NonMusicalPaperColumn #'line-break-permission = ##f
      \override NonMusicalPaperColumn #'page-break-permission = ##f
    }
  }
}
```





Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.3.7 Eine zusätzliche Stimme für Umbrüche benutzen

Zeilen- und Seitenumbruchbefehle werden normalerweise direkt zusammen mit den Noten eingegeben.
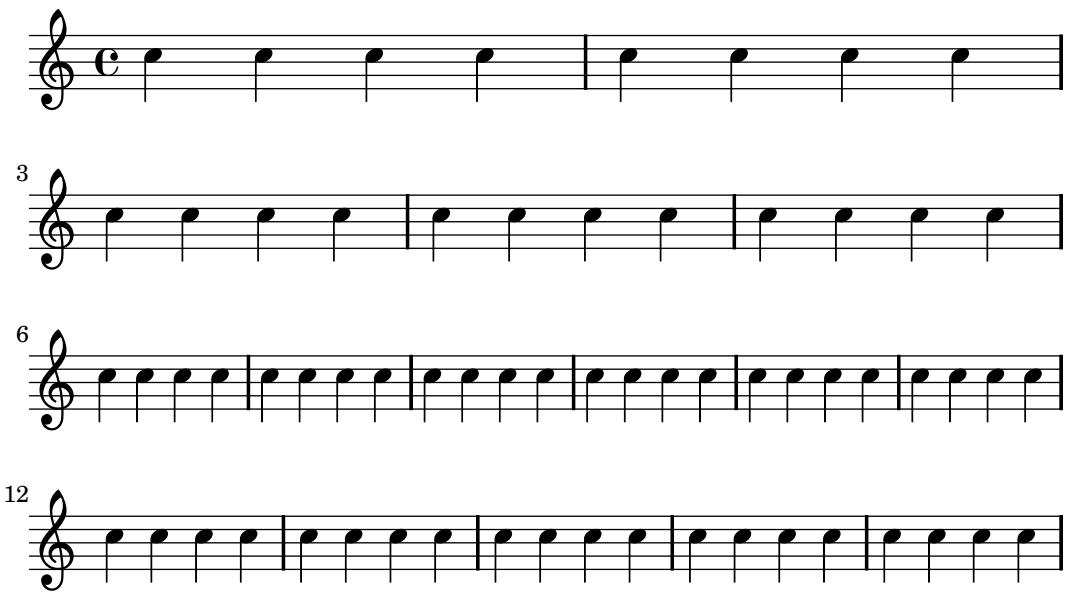
```
music = \relative c'' { c4 c c c }

\score {
  \new Staff {
    \repeat unfold 2 { \music } \break
    \repeat unfold 3 { \music }
  }
}
```

Hierdurch sind zwar die Befehle `\break` und `\pageBreak` einfach zu notieren, es werden aber Informationen zur Notation mit Informationen zur Anordnung auf der Seite vermischt. Man kann diese Informationen auch voneinander trennen, indem man eine zusätzliche Stimme einfügt, in der Zeilen- und Seitenumbrüche vorgenommen werden. Diese zusätzliche Stimme enthält nur unsichtbare Noten und die Umbruchbefehle:

```
music = \relative c'' { c4 c c c }

\score {
  \new Staff <<
    \new Voice {
      s1 * 2 \break
      s1 * 3 \break
      s1 * 6 \break
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 6 { \music }
      \repeat unfold 5 { \music }
    }
  >>
}
```



Mit dieser Herangehensweise kann der Code insbesondere dann klarer notiert werden, wenn man Einstellungen der `line-break-system-details`-Eigenschaft oder anderer Eigenschaften von `NonMusicalPaperColumnGrob` vornimmt (hierzu auch Abschnitt 4.4 [Vertikale Abstände], Seite 481).

```
music = \relative c'' { c4 c c c }
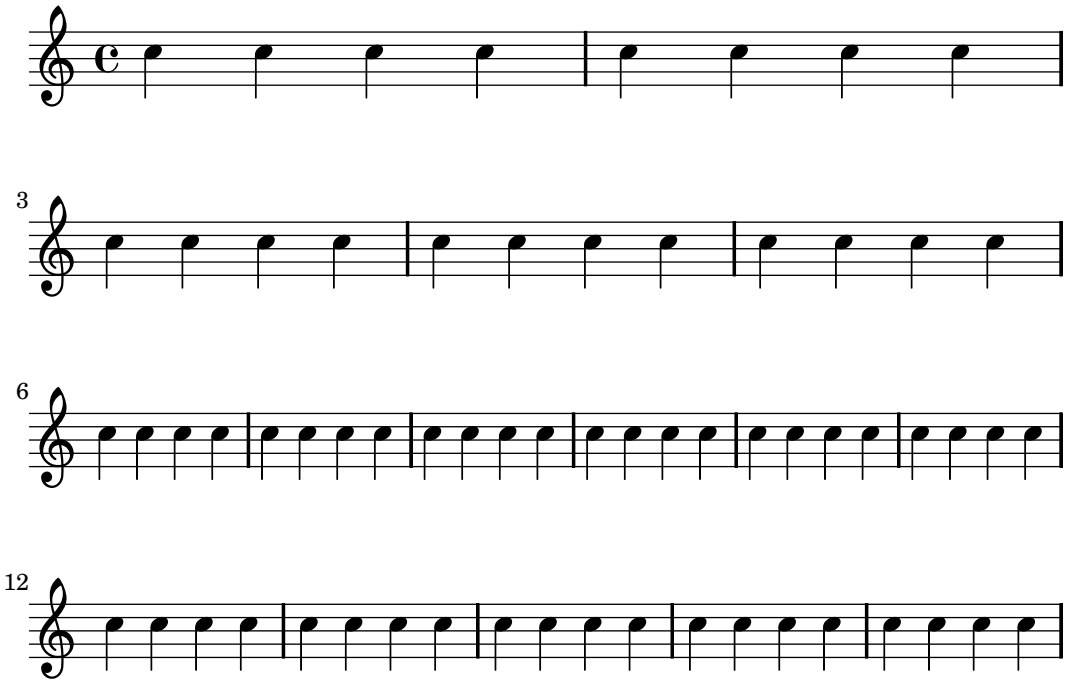
\score {
  \new Staff <<
    \new Voice {
      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 0))
      s1 * 2 \break

      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 35))
```

```
      s1 * 3 \break

      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 70))
      s1 * 6 \break

      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 105))
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 6 { \music }
      \repeat unfold 5 { \music }
    }
  >>
}
```



Siehe auch

Notationsreferenz: Abschnitt 4.4 [Vertikale Abstände], Seite 481.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.4 Vertikale Abstände

Vertikale Abstände werden durch drei Eigenschaften bestimmt: wieviel Platz frei ist (etwa Papiergröße und Ränder), wieviel Platz zwischen Systemgruppen (engl. system) gesetzt werden soll und wieviel Platz zwischen Notensystemen (engl. staff, Pl. staves) innerhalb von Gruppen gesetzt wird.

4.4.1 Flexible vertikale Abstände in Systemgruppen

Drei unterschiedliche Mechanismen kontrollieren das flexible Abstandaufteilung in Systemgruppen, einer für jede der folgenden Kathegorien:

- *ungruppierte Systeme*,
- *Systemgruppen* (Systeme innerhalb einer `staff-group` wie etwa `ChoirStaff` usw.) und
- *Nicht-Notensystemzeilen* (wie etwa `Lyrics` (Gesangstext), `ChordNames` (Akkordbezeichnungen) usw.).

Die Höhe jeder Systemgruppe wird in zwei Schritten bestimmt. Zunächst werden alle Systeme anhand des vorhandenen Platzes aufgeteilt. Dann werden die nicht-Notensysteme (also Akkorde oder Gesangstext) zwischen den Systemen verteilt.

Es ist zu beachten, dass der Platzverteilungsmechanismus, der in diesem Abschnitt behandelt wird, nur die vertikale Platzierung von Systemen und nicht-Systemzeilen in einzelnen Systemgruppen behandelt. Die vertikale Platzierung zwischen einzelnen Systemgruppen, Partituren, Beschriftungen usw. und den Rändern wird durch `\paper`-Variablen kontrolliert, die [Vertikale \paper-Variablen mit flexiblen Abständen], Seite 464.

Eigenschaften für Abstände innerhalb von Systemgruppen

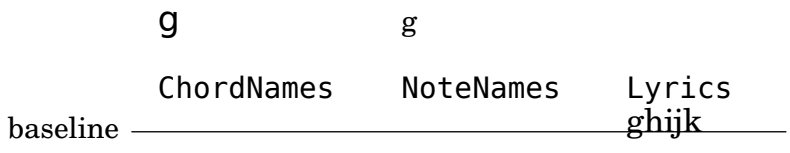
Der vertikalen Platzierungsmechanismen für Abstände innerhalb von Systemgruppen werden durch zwei Gruppen von Grob-Eigenschaften kontrolliert. Die erste Gruppe ist mit dem `VerticalAxisGroup`-Grob verknüpft, der von allen Notensystemen und Nicht-Notensystemzeilen erstellt wird. Die zweite Gruppe ist mit dem `StaffGrouper`-Grob verknüpft, der von Systemgruppen erstellt werden kann, aber nur, wenn das explizit verlangt wird. Die einzelnen Eigenschaften werden am Ende dieses Abschnitts beschrieben.

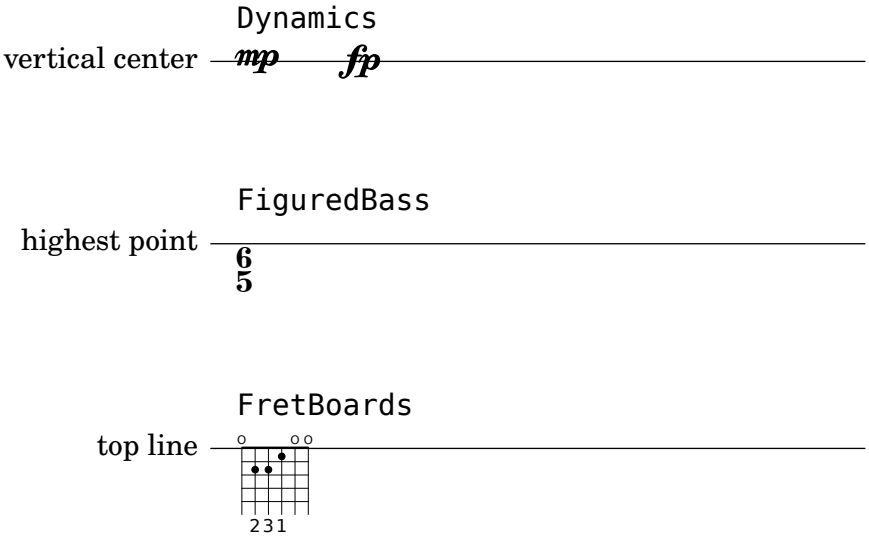
Die Bezeichnungen dieser Eigenschaften (mit Außnahmen von `staff-affinity`) haben das Format *`Element1`*`-`*`Element2`*`-spacing`, wobei *`Element1`* und *`Element2`* die Elemente sind, deren Abstände eingestellt werden sollen. Dabei ist allerdings zu beachten, dass *`Element2`* sich nicht notwendigerweise unterhalb von *`Element1`* befindet; beispielsweise `nonstaff-relatedstaff-spacing` (Nicht-Notensystem-verwandtesNotensystem) misst von dem Nicht-Notensystem nach oben, wenn `staff-affinity` (Richtung, an der sich ein System ausrichtet) auf `#UP` (nach oben) eingestellt ist.

Jeder Abstand wird zwischen den *Referenzpunkten* der zwei Objekten gemessen. Der Referenzpunkt eines Notensystems ist die vertikale Mitte seines `StaffSymbol`-Objekts (also die Mittellinie, wenn `line-count` (Notenlinienzähler) ungrade ist, oder der mittlere Zwischenraum, wenn `line-count` grade ist). Die Referenzpunkte für einzelne Nicht-Notensystemzeilen ergibt sich aus der folgenden Tabelle:

Nicht-Notensystemzeile	**Referenzpunkt**
`ChordNames`	Grundlinie
`NoteNames`	Grundlinie
`Lyrics`	Grundlinie
`Dynamics`	vertikale Mitte
`FiguredBass`	höchster Punkt
`FretBoards`	Oberlinie

Im nächsten Bild zeigen horizontale Striche die Positionen dieser Referenzpunkte an:





Jeder der vertikalen Platzierungs-Grobeigenschaften (außer `staff-affinity`) benutzt die gleiche Alistenstruktur wie die `\paper`-Variablen, behandelt in [Vertikale \paper-Variablen mit flexiblen Abständen], Seite 464. Besondere Methoden um Alisten zu verändern finden sich in Abschnitt 5.3.6 [Alisten verändern], Seite 534. Grob-Eigenschaften sollten mit dem `\override`-Befehle innerhalb einer `\score`- oder `\layout`-Umgebung angepasst werden, nicht innerhalb einer `\paper`-Umgebung.

Das folgende Beispiel zeigt die beiden Arten, Alisten zu modifizieren. Der erste Aufruf verändert nur einen Schlüsselwert einzlen, während der zweite die Eigenschaft komplett neu definiert:

```
\new Staff \with {
  \override VerticalAxisGroup #'default-staff-staff-spacing
       #'basic-distance = #10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup #'default-staff-staff-spacing =
    #'((basic-distance . 10)
       (minimum-distance . 9)
       (padding . 1)
       (stretchability . 10))
} { ... }
```

Um Platzierungseinstellungen global vorzunehmen, müssen sie in der `\layout`-Umgebung vorgenommen werden:

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup #'default-staff-staff-spacing
         #'basic-distance = #10
  }
}
```

Standardeinstellungen für die vertikalen Platzierungs-Grobeigenscahften finden sich in Abschnitt “VerticalAxisGroup” in *Referenz der Interna* und Abschnitt “StaffGrouper” in *Referenz der Interna* aufgelistet. Standardveränderungen für bestimmte Typen von Nicht-Notensystemzeilen finden sich im relevanten Abschnitt in in Abschnitt “Contexts” in *Referenz der Interna* aufgelistet.

Eigenschaften des `VerticalAxisGroup`-Grobs

`VerticalAxisGroup`-Eigenschaften werden normalerweise mit einem `\override`-Befehl auf `Staff`-(Notensystem-)Ebene (oder entsprechend) vorgenommen.

`staff-staff-spacing`
: System-System-Platzierung

  Wird benutzt, um den Abstand zwischen dem aktuellen Notensystem und dem Notensystem direkt darunter in der gleichen Notensystemgruppe zu bestimmen, auch wenn eine oder mehrere Nicht-Notensystemzeilen (wie etwa `Lyrics`) dazwischen stehen. Gilt nicht für das unterste System einer Systemgruppe.

  Die Eigenschaft `staff-staff-spacing` einer `VerticalAxisGroup` ist eine Scheme-Funktion, welche die Eigenschaften vom `StaffGrouper`-Grob den Notensystemen zuweist, wenn sie einer Systemgruppe angehören, bzw. die `default-staff-staff-spacing`-Eigenschaft bei einem einzelnen Notensystem. Dadurch können Systeme unterschiedlich in Abhängigkeit von ihrer Zugehörigkeit zu einer Gruppe (StaffGroup) platziert werden. Wenn gleichmäßige Aufteilung ohne Berücksichtigung von Gruppierungen gewünscht ist, kann diese Funktion durch eine flexible Platzierungs-Aliste ersetzt werden, wobei man die vollständige Redefinition auf die oben gezeigte Weise vornehmen muss.

`default-staff-staff-spacing`
: Normale-System-System-Platzierung

  Eine flexible Platzierungs-Aliste, die den Wert von `staff-staff-spacing` für ungruppierte Notensysteme einstellt, es sei denn, `staff-staff-spacing` wirde explizit mit `\override` eingestellt.

`staff-affinity`
: System-Anziehung

  Die Richtung des Systems, die benutzt wird, um die aktuelle Nicht-Notensystemzeile zu platzieren. Mögliche Werte sind `UP` (nach oben), `DOWN` (nach unten) und `CENTER` (mittig). Wenn `CENTER` wird die Nicht-Notensystemzeile vertikal mittig zwischen den beiden nächsten Systemen oben und unten platziert, außer Zusammenstöße und andere Platzierungsprobleme verhindern das. Aufeinanderfolgende Nicht-Notensystemzeilen sollten nicht-aufsteigende `staff-affinity` von oben nach unten haben; also ein Nicht-Notensystemzeile mit `UP` sollte nicht direkt auf eine mit `DOWN` folgen. Nicht-Notensystemzeilen über einem Notensystem sollten `DOWN` benutzen, unter einem Notensystem dagegen `UP`. Wenn `staff-affinity` für eine Notensystem eingestellt wird, wird es wie eine Nicht-Notensystemzeile behandelt. Wenn `staff-affinity` auf `#f` gesetzt wird, wird eine Nicht-Notensystemzeile wie ein Notensystem behandelt. Wird `staff-affinity` auf `UP`, `CENTER` oder `DOWN` gesetzt, wird das Notensystem als Nicht-Notensystemzeile plaziert.

`nonstaff-relatedstaff-spacing`
: Nicht-Notensystem-verwandtesSystem-Platzierung

  Der Abstand zwischen der aktuellen Nicht-Notensystemzeile und dem nächsten Notensystem in der Richtung von `staff-affinity`, wenn keine Nicht-Notensystemzeilen dazwischen auftreten und `staff-affinity` entweder `UP` oder `DOWN` ist. Wenn `staff-affinity CENTER` ist, dann wird `nonstaff-relatedstaff-spacing` für die nächsten Notensysteme auf *beiden* Seiten benutzt, auch wenn andere Nicht-Notensystemzeilen zwischen der aktuellen und einem der Notensystem auftreten. Das heißt, dass die Platzierung eine Nicht-Notensystemzeile sowohl von den umliegenden Notensystemen als auch den umliegenden Nicht-Notensystemzeilen abhängt. Wenn `stretchability` einer dieser

Platzierungstypen auf einen kleinen Wert gesetzt wird, dominiert diese Platzierung. Wird es dagegen auf einen großen Wert gesetzt, hat die Platzierung dieses Objekts nur einen kleinen Einfluss.

`nonstaff-nonstaff-spacing`
: Nicht-Notensystemzeile-Nicht-Notensystemzeile-Platzierung

  Der Abstand zwischen der aktuellen Nicht-Notensystemzeile und der Nicht-Notensystemzeile in der Richtung von `staff-affinity`, wenn beide sich auf der gleichen Seite des verwandten Notensystems befinden und `staff-affinity` entweder `UP` oder `DOWN` ist.

`nonstaff-unrelatedstaff-spacing`
: Nicht-Notensystemzeile-Nicht-verwandtesSystem-Platzierung

  Der Abstand zwischen der aktuellen Nicht-Notensystemzeile und dem Notensystem in der gegenüberliegenden Richtung von `staff-affinity`, wenn keine anderen Nicht-Notensystemzeilen dazwischen auftreten und `staff-affinity` entweder `UP` oder `DOWN` ist. Das kann benutzt werden, um einen Minimalfüllabstand (padding) zwischen einer `Lyrics`-Gesangstextzeile und dem zugehörigen Notensystem zu verlangen.

Eigenschaften des `StaffGrouper`-Grobs

`StaffGrouper`-Eigenschaften werden normalerweise mit einem `\override`-Befehl auf `StaffGroup`-Ebene (oder entsprechend) eingestellt.

`staff-staff-spacing`
: Notensystem-Notensystem-Abstand

  Der Abstand zwischen zwei aufeinanderfolgenden Notensystemen in der aktuellen `StaffGroup`. Die `staff-staff-spacing`-Eigenschaft des `VerticalAxisGroup`-Grobs eines einzelnen Notensytems kann mit `\override` in andere Platzierungswerte für dieses Notensystem geändert werden.

`staffgroup-staff-spacing`
: Systemgruppe-System-Abstand

  Der Abstand zwischen dem letzten Notensystem der aktuellen `StaffGroup` und dem Notensystem dirket darunter in der selben Notensystemgruppe, auch wenn eine oder mehrere Nicht-Notensystemzeilen (wie etwa Gesangstext) zwischen den zwei Notensystemen vorkommen. Gilt nicht für das letzte Notensystem einer Systemgruppe. Die `staff-staff-spacing`-Eigenschaft des `VerticalAxisGroup`-Grobs individueller Notensysteme kann mit `\override` in andere Platzierungswerte für dieses Notensystem geändert werden.

Siehe auch

Notationsreferenz: [Vertikale \paper-Variablen mit flexiblen Abständen], Seite 464, Abschnitt 5.3.6 [Alisten verändern], Seite 534.

Referenz der Interna: Abschnitt “Contexts” in *Referenz der Interna*, Abschnitt “VerticalAxisGroup” in *Referenz der Interna*, Abschnitt “StaffGrouper” in *Referenz der Interna*.

Installierte Dateien: ‘`ly/engraver-init.ly`’, ‘`scm/define-grobs.scm`’.

Abstände von nicht gruppierten Notensystemen

Notensysteme (wie etwa `Staff`, `DrumStaff`, `TabStaff` usw.) sind Kontexte, die eine oder mehrere Stimmen-Kontexte enthalten, aber keine anderen Notensysteme enthalten können.

Folgende Eigenschaften beeinflussen die Abstände von *nicht gruppierten* Notensystemen:

- `VerticalAxisGroup`-Eigenschaften:
  - `default-staff-staff-spacing`
  - `staff-staff-spacing`

Diese Eigenschaften sind einzlen oben behandelt worden, siehe [Eigenschaften für Abstände innerhalb von Systemgruppen], Seite 482.

Zusätzliche Eigenschaften kommen hinzu für Notensysteme, die Teil einer Gruppieren (`StaffGroup`) werden, siehe [Abstände von gruppierten Notensystemen], Seite 487.

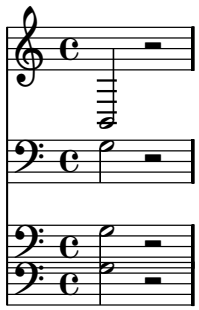
Folgendes Beispiel zeigt, wie die `default-staff-staff-spacing`-Eigenschaft sich auf die Platzierung von nicht-gruppierten Notensystemen auswirken kann. Wenn man die gleichen `\override`-Befehle auf `staff-staff-spacing` anwendet, ergäbe das den selben Effekt, würde sich aber auch in Fällen auswirken, in denen die Systeme gruppiert sind.

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup #'default-staff-staff-spacing =
      #'((basic-distance . 8)
         (minimum-distance . 7)
         (padding . 1))
  }
}

<<
  % The very low note here needs more room than 'basic-distance
  % can provide, so the distance between this staff and the next
  % is determined by 'padding.
  \new Staff { b,2 r | }

  % Here, 'basic-distance provides enough room, and there is no
  % need to compress the space (towards 'minimum-distance) to make
  % room for anything else on the page, so the distance between
  % this staff and the next is determined by 'basic-distance.
  \new Staff { \clef bass g2 r | }

  % By setting 'padding to a negative value, staves can be made to
  % collide.  The lowest acceptable value for 'basic-distance is 0.
  \new Staff \with {
    \override VerticalAxisGroup #'default-staff-staff-spacing =
      #'((basic-distance . 3.5)
         (padding . -10))
  } { \clef bass g2 r | }
  \new Staff { \clef bass g2 r | }
>>
```



Siehe auch

Installierte Dateien: ‘`scm/define-grobs.scm`’.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Referenz der Interna: Abschnitt “VerticalAxisGroup” in *Referenz der Interna*.

Abstände von gruppierten Notensystemen

In Orchesterpartituren und anderen großen Partituren werden Notensysteme normalerweise in Gruppen zusammengefasst. Der Platz zwischen Gruppen ist normalerweise größer als der Zwischenraum zwischen einzelnen Notensystemen der gleichen Gruppe.

Gruppierte Notensysteme (wie `StaffGroup`, `ChoirStaff`, `GrandStaff` usw.) sind Kontexte, die mehr als ein Notensystem gleichzeitig enthalten können.

Folgende Eigenschaften beeinflussen die Platzierung von Notensystemen innerhalb von Gruppen:

- `VerticalAxisGroup`-Eigenschaften:
  - `staff-staff-spacing`
- `StaffGrouper`-Eigenschaften:
  - `staff-staff-spacing`
  - `staffgroup-staff-spacing`

Diese Grob-Eigenschaften sind weiter oben einzeln beschrieben, siehe [Eigenschaften für Abstände innerhalb von Systemgruppen], Seite 482.

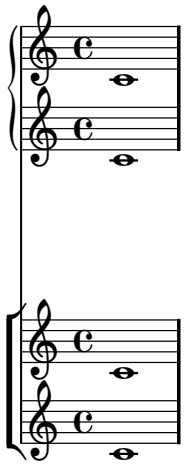
Das folgende Beispiel zeigt, wie Eigenscchaften des `StaffGrouper`-Grobs die Platzierung von gruppierten Notensystemen beeinflussen kann:

```
\layout {
  \context {
    \Score
    \override StaffGrouper #'staff-staff-spacing #'padding = #0
    \override StaffGrouper #'staff-staff-spacing #'basic-distance = #1
  }
}

<<
  \new PianoStaff \with {
    \override StaffGrouper #'staffgroup-staff-spacing #'basic-distance = #20
  } <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>

  \new StaffGroup <<
    \new Staff { c'1 }
    \new Staff { c'1 }
```

```
  >>
>>
```



Siehe auch

Installierte Dateien: ‘`scm/define-grobs.scm`’.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Referenz der Interna: Abschnitt “VerticalAxisGroup” in *Referenz der Interna*, Abschnitt “StaffGrouper” in *Referenz der Interna*.

Abstände von nicht-Notensystemzeilen

Nicht-Notensystemzeilen (wie `Lyrics`, `ChordNames` usw.) sind Kontexte, deren Layoutobjekte wie Notensysteme gesetzt werden (also als horizontale Zeilen zwischen Notensystemen). Genau gesagt sind Nicht-Notensystemzeilen Nicht-Notensystemkontexte, die ein `VerticalAxisGroup`-Layoutobjekt erstellen.

Folgende Eigenschaften beeinflussen die Abstände von Nicht-Notensystemzeilen:

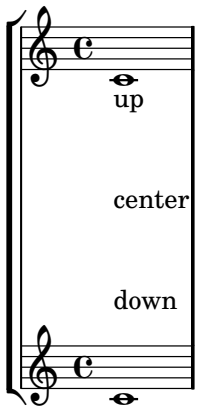
- `VerticalAxisGroup`-Eigenschaften:
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`
  - `nonstaff-unrelatedstaff-spacing`

Diese Grob-Eigenschaften sind weiter oben einzlen beschrieben; siehe [Eigenschaften für Abstände innerhalb von Systemgruppen], Seite 482.

Das folgende Beispiel zeigt, wie die `nonstaff-nonstaff-spacing`-Eigenschaft die Platzierung von aufeinanderfolgenden Nicht-Notensystemzeilen beeinflussen kann. Indem hier der Wert von `stretchability` auf einen sehr hohen Wert gesetzt wird, kann der Gesangstext sehr viel weiter als normal gespreizt werden:

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
      #'nonstaff-nonstaff-spacing #'stretchability = #1000
  }
}
```

```
\new StaffGroup
<<
  \new Staff \with {
    \override VerticalAxisGroup #'staff-staff-spacing = #'((basic-distance . 30))
  } { c'1 }
  \new Lyrics \with {
    \override VerticalAxisGroup #'staff-affinity = #UP
  } \lyricmode { up }
  \new Lyrics \with {
    \override VerticalAxisGroup #'staff-affinity = #CENTER
  } \lyricmode { center }
  \new Lyrics \with {
    \override VerticalAxisGroup #'staff-affinity = #DOWN
  } \lyricmode { down }
  \new Staff { c'1 }
>>
```



Siehe auch

Installierte Dateien: ‘`ly/engraver-init.ly`’, ‘`scm/define-grobs.scm`’.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Referenz der Interna: Abschnitt “Contexts” in *Referenz der Interna*, Abschnitt “VerticalAxisGroup” in *Referenz der Interna*.

4.4.2 Explizite Positionierung von Systemen

Man kann die flexiblen Einstellungen der vertikalen Abstände, wie sie im vorigen Abschnitt erklärt wurden, als eine Sammlung verschiedenerer Einstellmöglichkeiten verstehen, die vor allem die Größe des vertikalen Platzes zwischen Notensystemen und Gruppen auf der Seite kontrollieren.

Die vertikale Platzverteilung kann aber auch auf andere Weise eingestellt werden: mit den Optionen von `NonMusicalPaperColumn #'line-break-system-details`. Während der flexible vertikale Abstandsmechanismus vertikalen Füllplatz definiert, werden mit `NonMusicalPaperColumn #'line-break-system-details` absolute vertikale Positionen auf der Seite festgelegt.

`NonMusicalPaperColumn #'line-break-system-details` akzeptiert eine Liste aus drei unterschiedlichen Einstellungen:

- `X-offset`
- `Y-offset`

- `alignment-distances`

Veränderungen von Grobs (wozu auch `NonMusicalPaperColumn` gehört), können an drei unterschiedlichen Stellen in der Quelldatei vorgenommen werde:

- mitten im Notentext
- in einer `\context`-Umgebung
- in einer `\with`-Umgebung

Wenn der Grob `NonMusicalPaperColumn` verändert werden soll, wird der `\override`-Befehl in der `\context` oder `\with`-Umgebung eingesetzt. Wenn die Veränderungen aber mitten im Notentext stattfinden sollen, müssen Sie den Befehl `\overrideProperty` einsetzen. Einige Beispiele für eine Veränderungen von `NonMusicalPaperColumn` mit dem `\overrideProperty`-Befehl sind hier aufgelistet:

```
\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((X-offset . 20))

\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((Y-offset . 40))
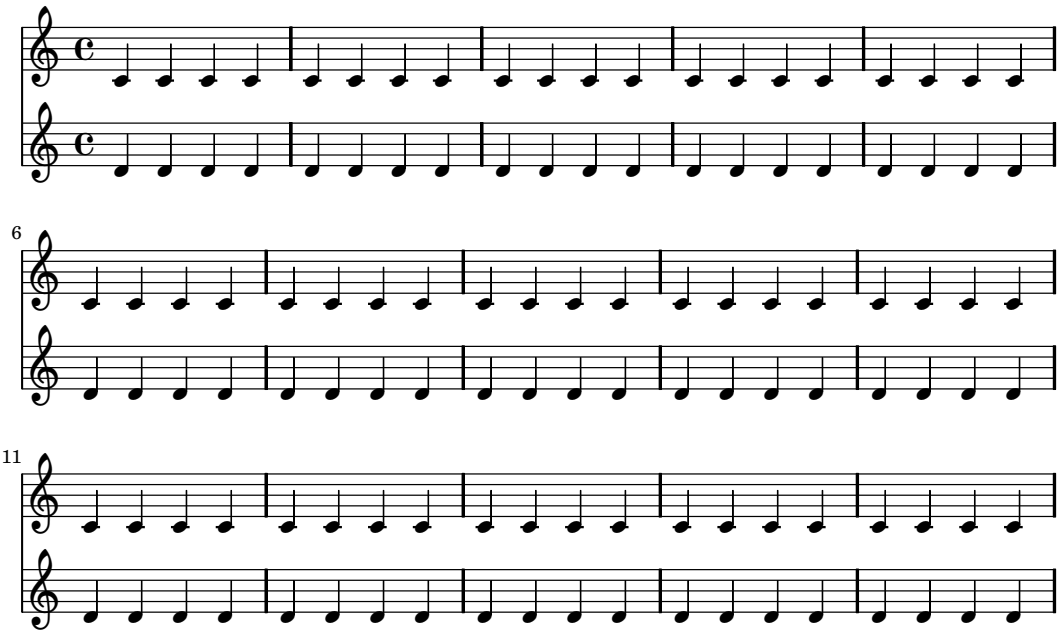
\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((X-offset . 20)
                                 (Y-offset . 40))

\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((alignment-distances . (15)))

\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((X-offset . 20)
                                 (Y-offset . 40)
                                 (alignment-distances . (15)))
```

Um zu verstehen, wie jede dieser unterschiedlichen Einstellungen funktioniert, wollen wir uns ein Beispiel vornehmen, dass überhaupt keine Einstellungen (d.h. `\override`-Befehle) enthält:
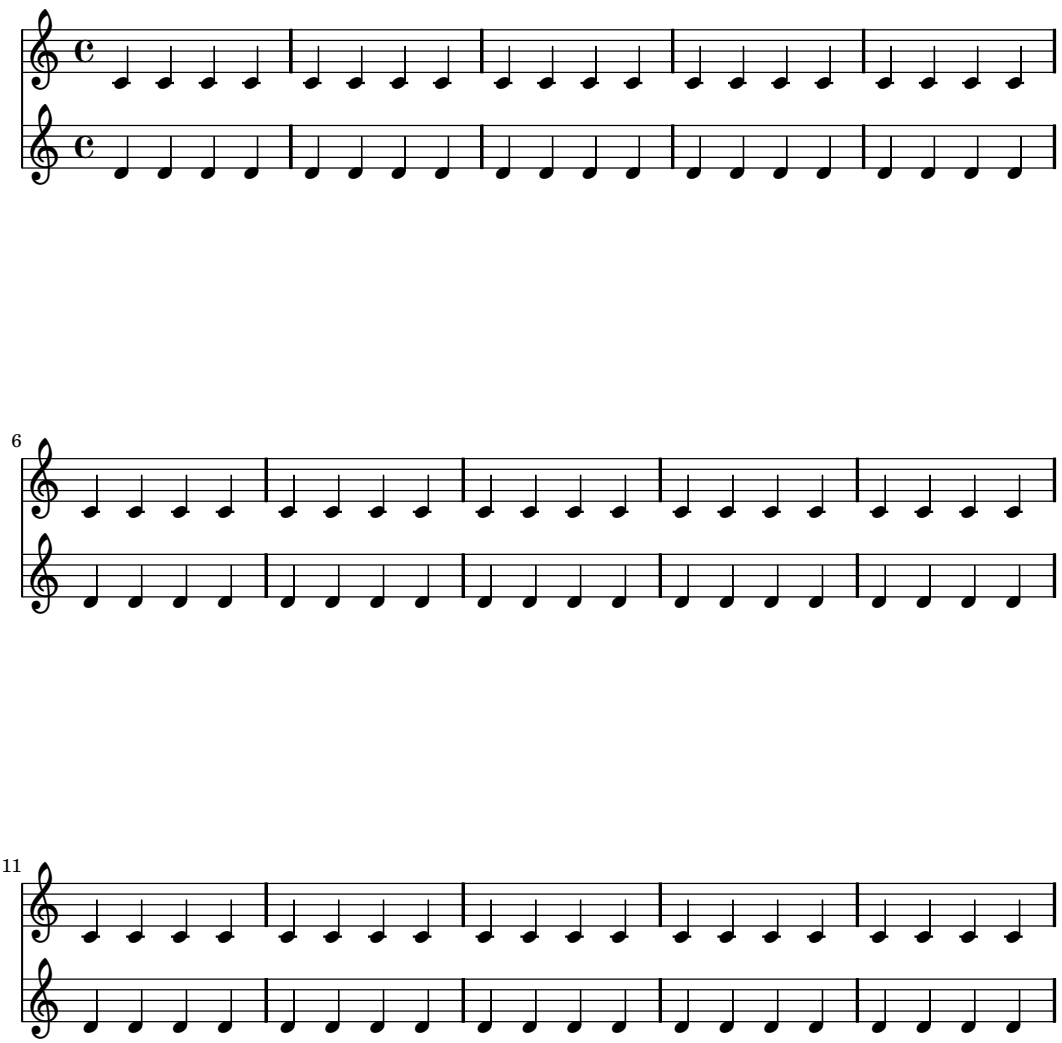
```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```



Diese Partitur nimmt Zeilen- und Seitenumbruchinformationen in einer eigenen Stimme vor. Mit dieser Methode kann die Layout-Information einfach von den Noten getrennt werden, was sehr hilfreich ist, wenn das Beispiel komplizierter wird. Siehe auch Abschnitt 4.3.7 [Eine zusätzliche Stimme für Umbrüche benutzen], Seite 479.

Ausdrückliche `\break`-Befehle teilen die Noten in sechs Takte lange Zeilen. Die vertikale Platzverteilung wird von LilyPond errechnet. Um den vertikalen Beginn einer jeden Systemgruppe genau anzugeben, kann `Y-offset` in der `line-break-system-details`-Eigenschaft des `NonMusicalPaperColumn`-Grobs wie in dem Beispiel ersichtlich benutzt werden:

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 0))
          s1*5 \break
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 40))
          s1*5 \break
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 80))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```
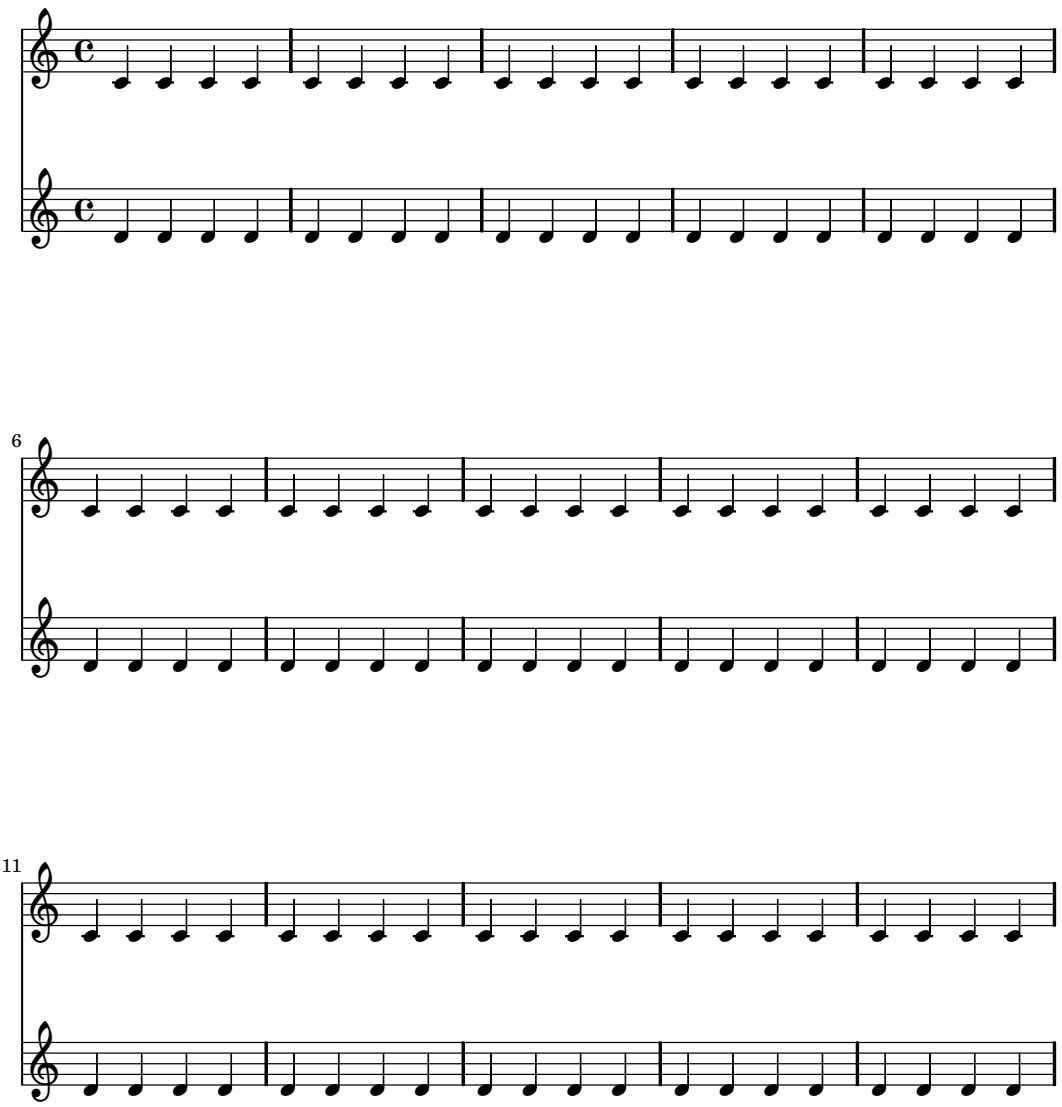


In der `line-break-system-details`-Eigenschaft kann eine Liste mit vielen Einstellungen eingegeben werden, aber hier wird nur eine Einstellung angegeben. Die `Y-offset`-Eigenschaft bestimmt hier die exakte vertikale Position auf der Seite, an welcher jede neue Systemgruppe begonnen wird.

Da jetzt der exakte Beginn einer jeden Systemgruppe explizit festgelegt wurde, können wir auch den exakten Beginn eines jeden Notensystems in der Gruppe festlegen. Dies geschieht mit der `alignment-distances`-Eigenschaft von `line-break-system-details`.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 20)
                                           (alignment-distances . (15)))
          s1*5 \break
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 60)
                                           (alignment-distances . (15)))
          s1*5 \break
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 100)
```
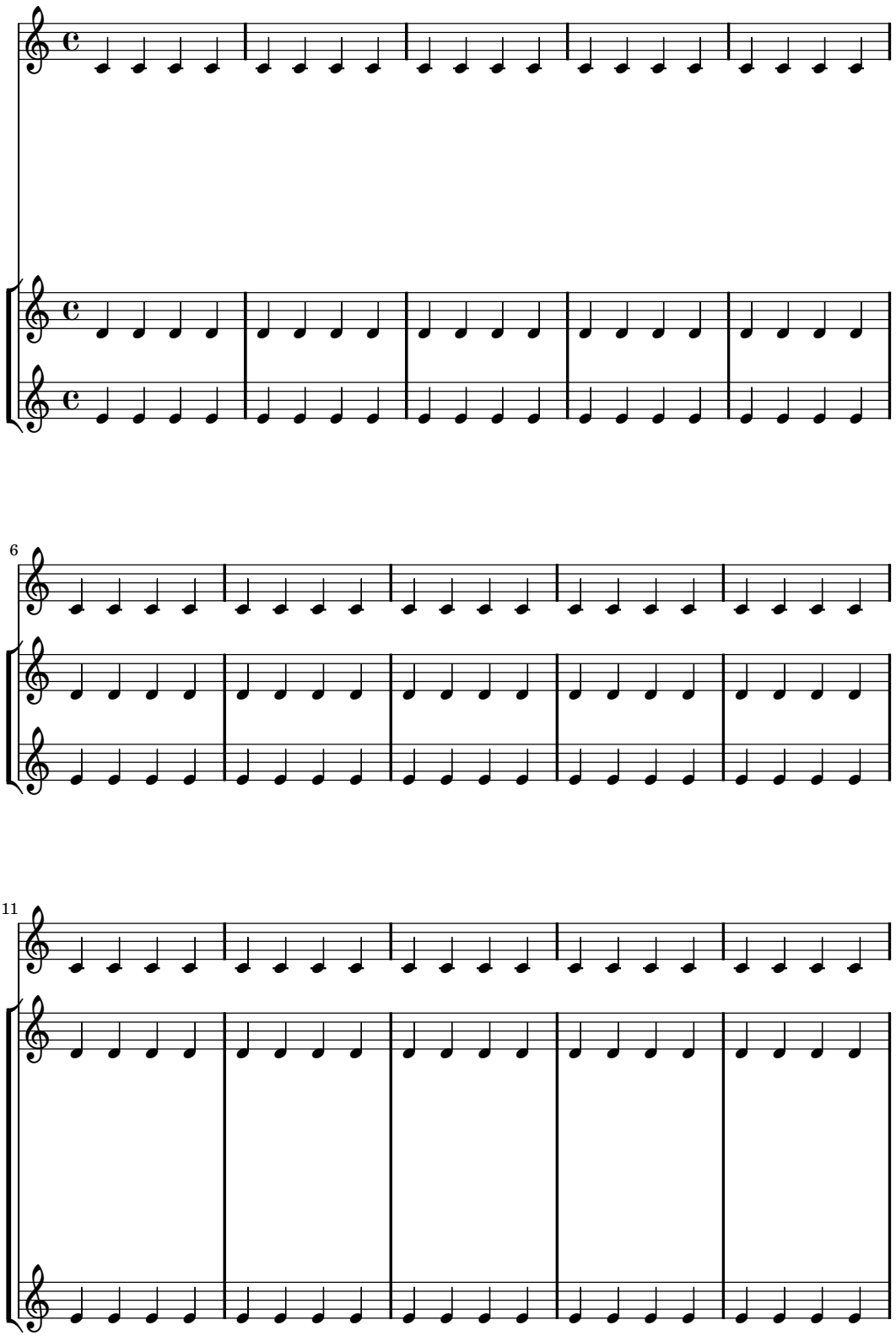
```
                                        (alignment-distances . (15)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```



Dem `line-break-system-details`-Attribut des `NonMusicalPaperColumn`-Grobs werden zwei unterschiedliche Eigenschaften zugewiesen. Auch wenn die Aliste der Attribute von

`line-break-system-details` sehr viel mehr Platzierungsparameter akzeptiert (wie etwa ein korrespondierendes `X-offset`-Paar), müssen hier nur die Parameter `Y-offset` und `alignment-distances` gesetzt werden, um den vertikalen Beginn jedes Systems und jeder Systemgruppe zu kontrollieren. `Y-offset` bestimmt also die vertikale Position von Systemgruppen und `alignment-distances` die vertikale Position von einzelnen Notensystemen.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 0)
                                           (alignment-distances . (30 10)))
          s1*5 \break
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 60)
                                           (alignment-distances . (10 10)))
          s1*5 \break
          \overrideProperty #"Score.NonMusicalPaperColumn"
            #'line-break-system-details #'((Y-offset . 100)
                                           (alignment-distances . (10 30)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new StaffGroup <<
        \new Staff { \repeat unfold 15 { d'4 d' d' d' } }
        \new Staff { \repeat unfold 15 { e'4 e' e' e' } }
      >>
    >>
  }
}
```



Einige Dinge sollten beachtet werden:

- Wenn `alignment-distances` benutzt wird, werden Gesangstextzeilen nicht als ein System gezählt.
- Die Einheiten der Zahlen, die für `X-offset`, `Y-offset` und `alignment-distances` benutzt werden, werden als Vielfaches des Abstandes zwischen zwei Notenlinien gewertet. Positive Werte verschieben Systeme und Gesangstext nach oben, negative Werte nach unten.
- Weil die Einstellungen von `NonMusicalPaperColumn #'line-break-system-details` es möglich machen, Notensysteme und Gruppen an beliebigen Stellen auf der Seite

zu platzieren, kann man damit auch Ränder überschreiben oder sogar Notensysteme übereinander platzieren. Sinnvolle Werte für diese Parameter werden derartiges Verhalten vermeiden.

Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

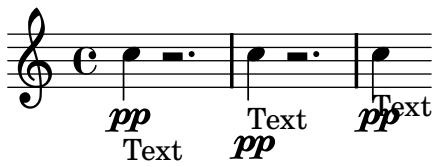
4.4.3 Vermeidung von vertikalen Zusammenstößen

Intuitiv gibt es in der Notation einige Objekte, die zu dem Notensystem gehören, und einige andere, die immer außerhalb des Notensystems positioniert werden sollten. Zu diesen letzteren gehören etwa Übungszeichen, Textbeschriftung und Dynamikbezeichnung (die als Objekte außerhalb des Systems bezeichnet werden können). LilyPonds Regeln um diese Objekte zu positionieren lautet: so nah am Notensystem wie möglich, aber gerade so weit weg, dass sie nicht mit anderen Objekten zusammenstoßen.

Dabei setzt LilyPond die `outside-staff-priority`-Eigenschaft ein um herauszufinden, ob ein Grob ein Objekt außerhalb des Systems ist: wenn `outside-staff-priority` eine Zahl ist, dann handelt es sich um ein Objekt außerhalb des Systems. Zusätzlich teilt `outside-staff-priority` noch mit, in welcher Reihenfolge die Objekte außerhalb des Systems gesetzt werden sollen.

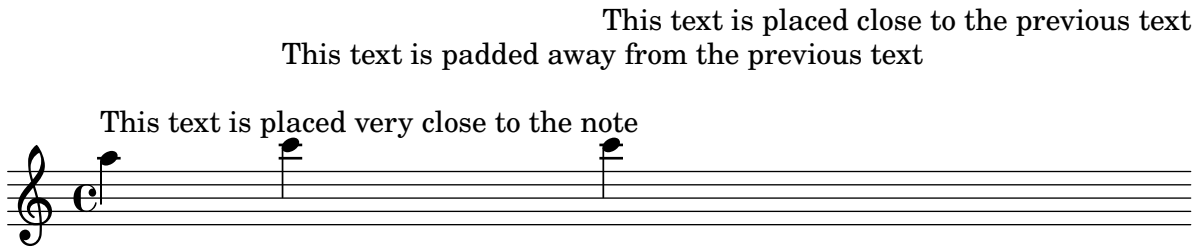
Zuerst werden alle Objekte gesetzt, die nicht außerhalb des Systems gehören. Dann werden die Objekte außerhalb des Systems nach dem Wert ihrer `outside-staff-priority` (in aufsteigender Anordnung) sortiert. Eins nach dem anderen werden diese Objekte schließlich genommen und so platziert, dass sie nicht mit den Objekten zusammenstoßen, die bereits platziert worden sind. Wenn also zwei Objekte außerhalb des Systems um den gleichen Platz streiten, wird das mit dem geringeren Wert von `outside-staff-priority` näher an das entsprechende Notensystem gesetzt.

```
c4_"Text"\pp
r2.
\once \override TextScript #'outside-staff-priority = #1
c4_"Text"\pp % this time the text will be closer to the staff
r2.
% by setting outside-staff-priority to a non-number,
% we disable the automatic collision avoidance
\once \override TextScript #'outside-staff-priority = ##f
\once \override DynamicLineSpanner #'outside-staff-priority = ##f
c4_"Text"\pp % now they will collide
```



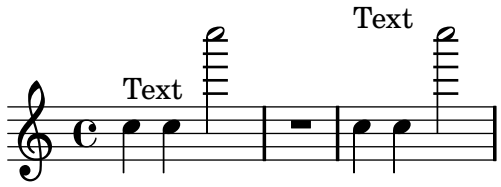
Der Platz, der zwischen einem Objekt außerhalb des Systems und dem vorhergehenden Objekt eingefügt werden kann (auch als padding bezeichnet), kann durch `outside-staff-padding` kontrolliert werden.

```
\once \override TextScript #'outside-staff-padding = #0
a'^"This text is placed very close to the note"
\once \override TextScript #'outside-staff-padding = #3
c^"This text is padded away from the previous text"
c^"This text is placed close to the previous text"
```



Standardmäßig werden Objekte außerhalb des Systems so gesetzt, dass sie eine horizontale Überschneidung mit einem der vorher gesetzten Grobs vermeiden. Das kann zu Situationen führen, in denen Objekte sehr dicht nebeneinander gesetzt werden. Der vertikale Platz zwischen Notensystemen kann auch gesetzt werden, sodass Objekte außerhalb des Systems ineinander greifen. Mit der Eigenschaft `outside-staff-horizontal-padding` können Objekte vertikal verschoben werden und derartige Situationen kommen nicht vor.

```
% the markup is too close to the following note
c4^"Text"
c4
c''2
% setting outside-staff-horizontal-padding fixes this
R1
\once \override TextScript #'outside-staff-horizontal-padding = #1
c,,4^"Text"
c4
c''2
```



Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.
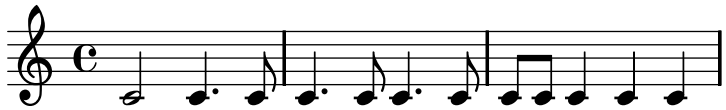
4.5 Horizontale Abstände

4.5.1 Überblick über horizontale Abstände

Die Setzmaschine interpretiert unterschiedliche Notendauern als dehnbare Abstände (engl. spring) unterschiedlicher Länge. Längere Dauern erhalten mehr Platz, kürzere weniger. Die kürzeste Dauer erhält eine feste Breite (die mit `shortest-duration-space` im `SpacingSpanner`-Objekt kontrolliert werden kann). Je länger die Dauer, umso mehr Platz erhält die Note: wenn ihre Dauer verdoppelt wird, wird ein bestimmter Platz hinzugefügt (dessen Breite durch `spacing-increment` bestimmt werden kann).

Das folgende Stück beispielsweise enthält Halbe, Viertel und Achtel. Die Achtelnote wird gefolgt von einem Notenkopfabstand (NKA). Die Viertel wird von 2 NKA gefolgt, die Halbe von 3 NKA usw.

```
c2 c4. c8 c4. c8 c4. c8 c8
c8 c4 c4 c4
```



Normalerweise ist `spacing-increment` definiert als 1.2 mal der Abstand zwischen zwei Notenlinien, was in etwa die Breite eines Notenkopfes ist. `shortest-duration-space` ist definiert als 2.0, was bedeutet, dass die kürzeste Note 2.4 Notenlinienabstände 2.0 mal der Wert von `spacing-increment`) horizontalen Abstand erhält. Der Abstand wird von der linken Kante des Symbols errechnet, so dass die kürzeste Note üblicherweise von 1 NKA Abstand gefolgt wird.
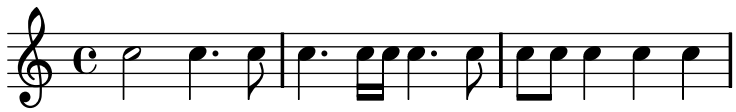
Wenn diese Herangehensweise konsequent angewandt würde, würde eine einzige Zweiunddreißigstel eine Partitur, in der vor allem Achtel und Sechzehntel vorkommen, sehr weit auseinanderdehnen. Die kürzeste Note wäre nun keine Sechzehntel mehr, sondern eine Zweiunddreißigstel, wodurch an jede Note der Wert von 1 NKA hinzugefügt würde. Um das zu vermeiden, ist die kürzeste Dauer für die Platzverteilung nicht die kürzeste Note einer Partitur, sondern die, die am häufigsten vorkommt.

Die Notendauer, die am häufigsten vorkommt, wird auf folgende Weise bestimmt: in jedem Takt wird die kürzeste Note bestimmt. Die häufigste kürzeste Note wird dann als Grundlage für die Platzverteilung der Noten herangezogen, mit der Bedingung, dass diese kürzeste Note immer ein Achtel oder kürzer sein soll. Die kürzeste Dauer wird ausgegeben, wenn `lilypond` mit der Option `--verbose` aufgerufen wird.
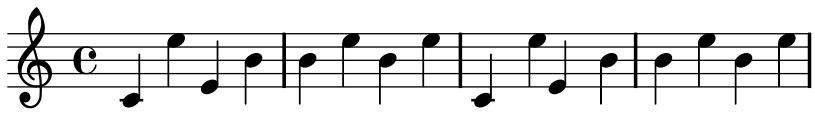
Diese Dauern können aber auch angepasst werden. Wenn Sie die Eigenschaft `common-shortest-duration` in dem `SpacingSpanner` setzen, dann wird hiermit die Grunddauer für die Platzverteilung eingestellt. Die maximale Dauer für diesen Grundwert (normalerweise eine Achtel) wird definiert mit `base-shortest-duration`.

Noten, die noch kürzer sind als die häufigste kürzeste Note, werden durch einen Platz voneinander getrennt, der proportional zu ihrer Dauer in Beziehung zur häufigsten kürzesten Note ist. Wenn also nur ein paar Sechzehntel zu dem obigen Beispiel hinzugefügt werden, würden sie von 1/2 NKA gefolgt werden:

```
c2 c4. c8 c4. c16[ c] c4. c8 c8 c8 c4 c4 c4
```



In dem *Aufsatz zum automatisierten Notensatz* wurde erklärt, dass die Richtung der Notenhälse die Platzverteilung beeinflusst (siehe Abschnitt “Optischer Ausgleich” in *Aufsatz*). Das wird kontrolliert durch die `stem-spacing-correction`-Eigenschaft in dem `NoteSpacing`-Objekt. Dieses Objekt wird für jeden `Voice`-Kontext erstellt. Das `StaffSpacing`-Objekt (in einem `Staff`-Kontext erstellt) enthält die gleiche Eigenschaft, um die Verteilung von Hälsen neben Taktlinien zu kontrollieren. In dem folgenden Beispiel werden diese Einstellungen gezeigt, einmal mit den Standardwerten und dann mit größeren Werten, damit man sie besser sieht:



Proportionale Notation ist unterstützt, siehe Abschnitt 4.5.5 [Proportionale Notation], Seite 502.

Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

Referenz der Interna: Abschnitt “SpacingSpanner” in *Referenz der Interna*, Abschnitt “NoteSpacing” in *Referenz der Interna*, Abschnitt “StaffSpacing” in *Referenz der Interna*, Abschnitt “NonMusicalPaperColumn” in *Referenz der Interna*.

Aufsatz über den automatischen Notensatz: Abschnitt “Optischer Ausgleich” in *Aufsatz*.

Bekannte Probleme und Warnungen

Es gibt keine sinnvolle Möglichkeit, die horizontale Verteilung der Noten zu unterdrücken. Die folgende Problemumgehung, mit der dehnbare Abstände (padding) eingesetzt werden, kann benutzt werden, um zusätzlichen Platz in eine Partitur einzufügen.

```
\once \override Score.SeparationItem #'padding = #10
```
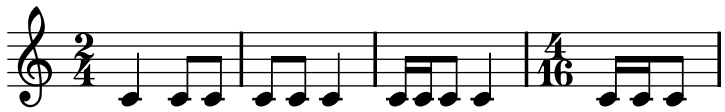
Es gibt derzeit keine Möglichkeit, den Platz zu verringern.

4.5.2 Eine neuer Bereich mit anderen Abständen

Neue Abschnitte mit unterschiedlichen Notenabstandsparametern können mit dem Befehl `newSpacingSection` begonnen werden. Das ist hilfreich, wenn in verschiedenen Abschnitten die Verhältnisse von kurzen und langen Noten sehr unterschiedlich ausfallen.

Im folgenden Beispiel wird durch die neue Taktart ein neuer Abschnitt begonnen, in dem die Sechzehntel weiter außeinander gesetzt werden sollen.

```
\time 2/4
c4 c8 c
c8 c c4 c16[ c c8] c4
\newSpacingSection
\time 4/16
c16[ c c8]
```



Der `\newSpacingSection`-Befehl erstellt ein neues `SpacingSpanner`-Objekt, weshalb auch neue Anpassungen mit dem `\override`-Befehl an dieser Stelle eingesetzt werden können.

Siehe auch
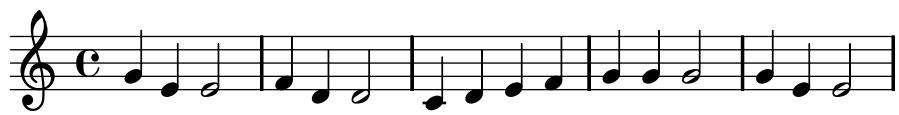
Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

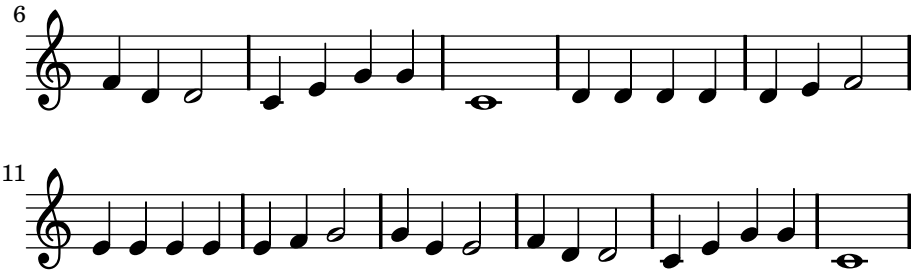
Referenz der Interna: Abschnitt “SpacingSpanner” in *Referenz der Interna*.

4.5.3 Horizontale Abstände verändern
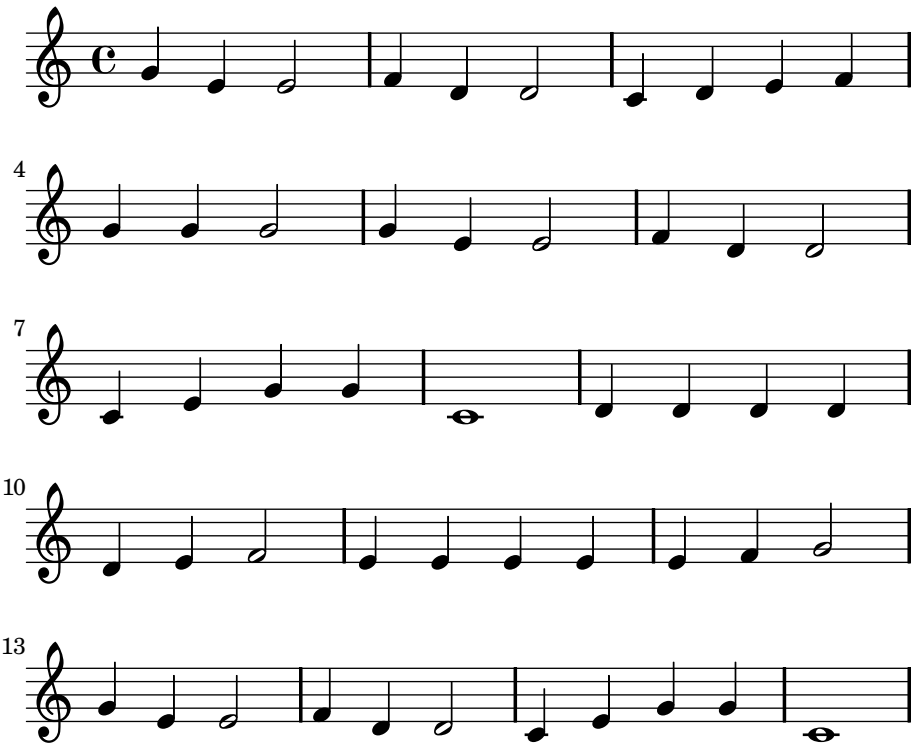
Die horizontalen Abstände können mit der `base-shortest-duration`-Eigenschaft verändert werden. In den folgenden Beispielen werden die gleichen Noten eingesetzt, zuerst ohne die Eigenschaft zu verändern, im zweiten Beispiel dann mit einem anderen Wert. Größere Werte für `ly:make-moment` ergeben dichtere Noten. `ly:make-moment` erstellt eine Dauer, die als Bruch notiert wird, sodass `1 4` eine größere Dauer ist als `1 16`.

```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
}
```





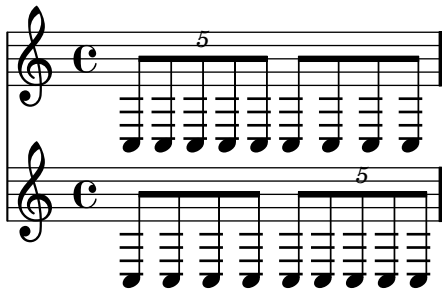

```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner
        #'base-shortest-duration = #(ly:make-moment 1 16)
    }
  }
}
```



Ausgewählte Schnipsel

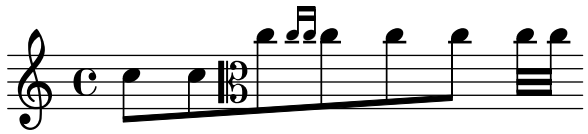
Standardmäßig wird die Platzverteilung in Triolen und andern rhythmischen Aufteilungen nach verschiedenen nicht von der Dauer abgeleiteten Faktoren (wie Versetzungszeichen, Schlüsselwechseln usw.) berechnet. Um diese Symbole zu ignorieren und eine gleichmäßige Verteilung der Noten zu erzwingen, kann die gleichmäßige Dehnung (engl. uniform stretching) zu Beginn einer Partitur mit `Score.SpacingSpanner #'uniform-stretching` eingeschaltet werden:

```
\score {
  <<
    \new Staff {
      \times 4/5 {
        c8 c8 c8 c8 c8
      }
      c8 c8 c8 c8
    }
    \new Staff {
      c8 c8 c8 c8
      \times 4/5 {
        c8 c8 c8 c8 c8
      }
    }
  >>
  \layout {
    \context {
      \Score
      \override SpacingSpanner #'uniform-stretching = ##t
    }
  }
}
```



Wenn `strict-note-spacing` eingestellt ist, werden Noten gesetzt, ohne dass Schlüssel, Taktlinie oder Verzierungsnoten zusätzlichen Platz erhalten.

```
\override Score.SpacingSpanner #'strict-note-spacing = ##t
\new Staff { c8[ c \clef alto c \grace { c16[ c] } c8 c c]  c32[ c32] }
```



Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.5.4 Zeilenlänge

Die grundlegenden Einstellungen, die Einfluss auf die Zeilenlänge haben, sind `indent` (Einzug) und `line-width` (Zeilenbreite). Sie werden in der `\layout`-Umgebung eingestellt. Der erste Befehl bestimmt den Einzug der ersten Zeile, der zweite die Zeilenlänge der weiteren Notenzeilen.

Wenn `ragged-right` eingestellt ist (als in der `\layout`-Umgebung auf den Wert `#t` gesetzt wurde), werden die Systeme linksbündig gesetzt und nicht bis zum rechten Rand hin durchgezogen, sondern den Noten entsprechend gesetzt. Das ist oftmals nützlich für kleine Notenfragmente

und um zu überprüfen, wie eng die Noten natürlicherweise gesetzt werden würden. Die normale Einstellung ist unwahr (`#f`), aber wenn eine Partitur nur aus einer Zeile besteht, ist der Standardwert wahr.

Die Option `ragged-last` verhält sich ähnlich zu `ragged-right`, aber wirkt sich nur auf die letzte Zeile eines Stückes aus. Für diese letzte Zeile gibt es keine Einschränkungen. Das Resultat erinnert an Textabsätze im Blocksatz, wo die letzte Zeile des Absatzes mit ihrer natürlichen Länge gesetzt wird.

```
\layout {
indent = #0
line-width = #150
ragged-last = ##t
}
```

Siehe auch

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.5.5 Proportionale Notation

LilyPond hat Unterstützung für proportionale Notation. Dabei handelt es sich um eine horizontale Platzverteilung, die jeder Note einen exakt ihrer Dauer entsprechenden Platz zuordnet. Man kann es vergleichen mit der Notenplatzierung auf einem Raster. In einigen Partituren des späten 20. und frühen 21. Jahrhunderts wird dies proportionale Notation benutzt, um sehr komplizierte rhythmische Verhältnisse klarer darzustellen, oder um einen Zeitstrahl oder ähnliche Graphiken direkt in die Partitur zu integrieren.
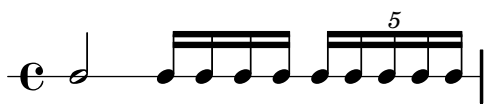
LilyPond hat Unterstützung für fünf verschiedene Einstellungen der proportionalen Notation, die alle zusammen oder jede für sich benutzt werden können:

- `proportionalNotationDuration` (proportionale Notendauer)
- `uniform-stretching` (gleichmäßige Dehnung)
- `strict-note-spacing` (strenge Notenverteilung)
- `\remove Separating_line_group_engraver` (entferne Liniengruppentrennungsengraver)
- `\override PaperColumn #'used = ##t` (PapierSpalte benutzt = wahr)

In den Beispielen unten werden diese fünf unterschiedlichen Einstellungen für die proportionale Notation vorgestellt und ihre Wirkungen untereinander illustriert.

Es soll mit diesem 1 Takt langen Beispiel begonnen werden, in welchem die klassischen Abstände und Flattersatz (ragged-right) eingesetzt werden:

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
}
```



Die Halbe, mit der der Takt beginnt, braucht weitaus weniger Platz als die Hälfte des Taktes. Gleichermaßen haben die Sechzehntel und die Sechzehntel-Quintolen (oder Zwanzigstel), mit denen der Takt endet, insgesamt weitaus mehr als die Hälfte der Taktbreite.

Im klassischen Notensatz kann dieses Verhalten genau das gewünschte Ergebnis bringen, weil dadurch horizontaler Platz von der Halben weggenommen werden kann und so insgesamt Platz in dem Takt eingespart wird.

Wenn allerdings ein Zeitstrahl oder andere zeitliche ablaufende Graphiken über oder unter dem Takt eingefügt werden soll, braucht man eine Notenplatzierung, die exakt der von ihnen eingenommenen Dauer entspricht. Auf folgende Art wird die proportionale Notation eingeschaltet:

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
 \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 20)
    }
  }
}
```



Die Halbe zu Beginn des Taktes und die schnelleren Noten in der zweiten Takthälfte nehmen jetzt genau den gleichen horizontalen Platz ein. Jetzt könnte man einen Zeitstrahl mit dem Takt synchronisieren.

Die Einstellung von `proportionalNotationDuration` gehört zum `Score`-Kontext. Kontexteinstellungen können an drei verschiedenen Stellen in der Quelldatei geschrieben werden: in einer `\with`-Umgebung, in einer `\context`-Umgebung oder direkt in den Noten mit dem `\set`-Befehl. Alle drei Positionen sind gleichwertig und es hängt vom Benutzer ab, welche bevorzugt wird.

Die Eigenschaft `proportionalNotationDuration` braucht ein Argument, welches die Referenzdauer ist, anhand welcher alle Noten platziert werden. Hier wird die LilyPond Scheme-Funktion `make-moment` eingesetzt. Sie braucht zwei Argumente: einen Zähler und einen Nenner, die einen Bruch einer Ganzen darstellen. Die Funktion `#(ly:make-moment 1 20)` ergibt also eine Referenzdauer von einer Zwanzigstel. Genauso gut können etwa die Dauern `#(ly:make-moment 1 16)`, `#(ly:make-moment 1 8)` oder `#(ly:make-moment 3 97)` eingesetzt werden.

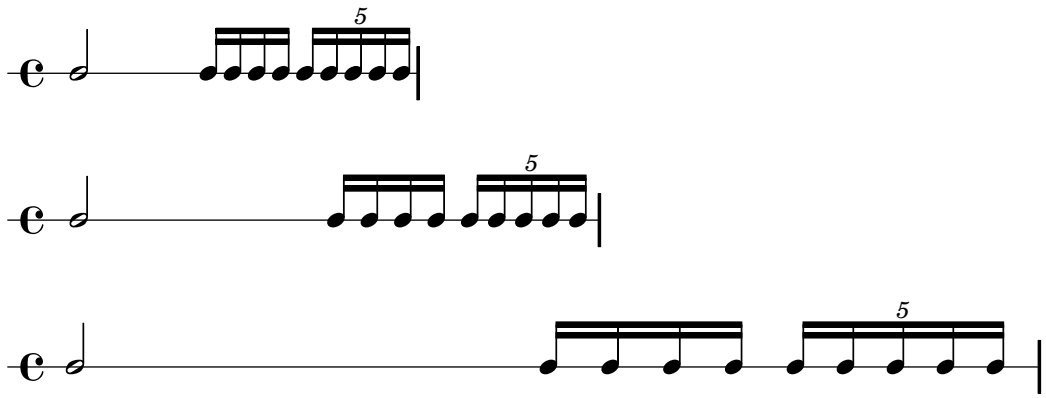
Die richtige Referenzdauer, mit der eine vernünftige Verteilung der Noten proportional möglich ist, muss durch Ausprobieren herausgefunden werden. Dabei sollte man mit einer Dauer beginnen, die der kleinsten Note des Stückes nahekommt. Kleine Referenzdauern lassen die Noten sehr gedehnt erscheinen, größere Referenzdauern zwingen sie dichter zusammen.

```
\score {
  <<
```

```
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 8)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 16)
    }
  }
}

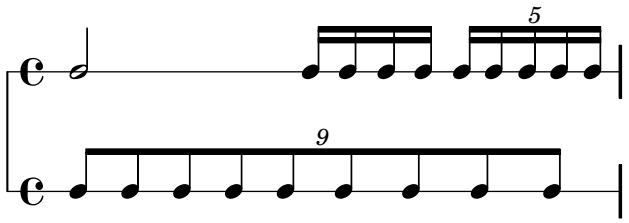
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 32)
    }
  }
}
```



Man muss beachten, dass die Referenzdauer nicht zu groß ist (wie die Achtel in dem Beispiel oben), denn dadurch werden die Noten so dicht gesetzt, dass sich eventuell sogar Notenköpfe von sehr kleinen Notenwerten überschneiden können. Die proportionale Notation nimmt üblicherweise mehr Platz ein als die klassische Platzverteilung. Der rhythmischen Klarheit muss ein eng gesetztes Notenbild geopfert werden.

In Folgenden soll betrachtet werden, wie sich überlappende rhythmische Aufteilungen am besten positioniert werden. Als Referenz wird das erste Beispiel herangezogen, zu welchem ein zweites System mit anderen rhythmischen Werten hinzugefügt wird:

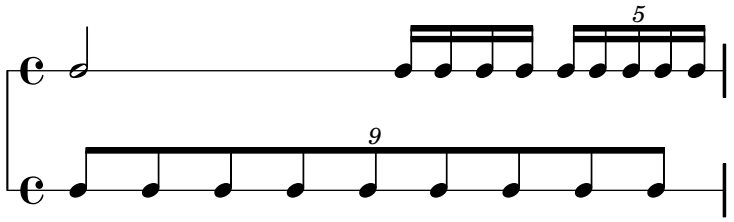
```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
    \new RhythmicStaff {
      \times 8/9 {
        c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
      }
    }
  >>
}
```



Die Platzaufteilung ist schlecht, weil die gleichlangen Noten des untersten Systems nicht gleichmäßig verteilt sind. Im klassischen Notensatz kommen komplexe rhythmische Verhältnisse wie dieses sehr selten vor, sodass der Notensatz nicht in Hinsicht auf sie optimiert ist. `proportionalNotationDuration` hilft in dieser Situation deutlich:

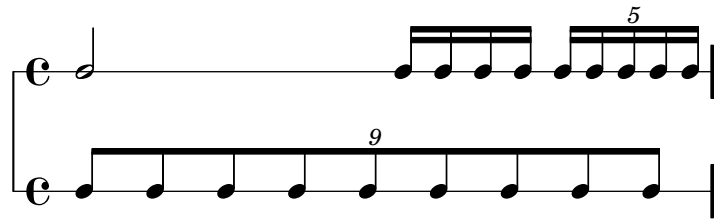
```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
```

```
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
    \new RhythmicStaff {
      \times 8/9 {
        c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 20)
    }
  }
}
```



Aber bei sehr genauer Betrachtung sind die Noten der zweiten Hälfte der Nonole doch immer noch eine Spur weiter gesetzt als die Noten der ersten Hälfte. Um wirklich gleichmäßige Abstände zu erzwingen, sollte auch noch die gleichmäßige Dehnung (`uniform-stretching`) angeschaltet werden, die eine Eigenschaft von `SpacingSpanner` ist:

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
    \new RhythmicStaff {
      \times 8/9 {
        c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 20)
      \override SpacingSpanner #'uniform-stretching = ##t
    }
  }
}
```



Das Beispiel mit den zwei Systemen ist nun exakt nach den rhythmischen Werten der Noten gesetzt, sodass ein Zeitstrahl oder ähnliches eingefügt werden könnte.

Alle Einstellungen zur proportionalen Notation erwarten, dass die `uniform-stretching`-Eigenschaft des `SpacingSpanner`-Objekts auf wahr (`#t`) gesetzt wird. Andernfalls kann es vorkommen, dass bestimmte Abstände (etwa von unsichtbaren Noten) nicht richtig gesetzt werden.

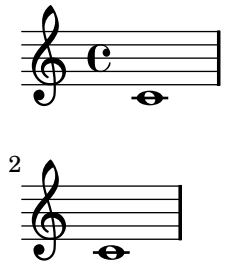
Das `SpacingSpanner`-Objekt ist ein abstraktes Grob, dass sich im `Score`-Kontext befindet. Genauso wie die Einstellungen von `proportionalNotationDuration` können auch diese Veränderungen an den drei Stellen in der Quelldatei vorkommen: in der `\with`-Umgebung innerhalb von `Score`, in einer `\context`-Umgebung oder direkt im Notentext.

Standardmäßig gibt es nur ein `SpacingSpanner` pro `Score`. Das heißt, dass `uniform-stretching` für die gesamte Partitur (d.h. für die Reichweite von `Score`) entweder an- oder ausgeschaltet ist. Man kann allerdings in einer Partitur unterschiedliche Abschnitte mit verschiedenem Platzierungsverhalten definieren. Hierzu ist der Befehl `\newSpacingSection` da. Siehe auch Abschnitt 4.5.2 [Eine neuer Bereich mit anderen Abständen], Seite 499.

Im Folgenden soll gezeigt werden, wie sich der `Separating_line_group_engraver` auswirkt und warum er normalerweise für proportionale Notation ausgeschaltet wird. In diesem Beispiel wird verdeutlicht, dass vor jeder ersten Note eines Notensystems immer etwas zusätzlicher Platz gesetzt wird:

```
\paper {
  indent = #0
}

\new Staff {
  c'1
  \break
  c'1
}
```

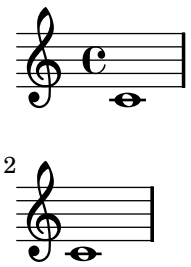


Der gleiche horizontale zusätzliche Platz wird vor eine Noten gesetzt, wenn sie einer Taktart, einem Schlüssel oder einer Tonartbezeichnung folgt. Dieser Platz wird durch `Separating_line_group_engraver` eingefügt; wenn wir ihn aus der Partitur entfernen, entfällt auch dieser zusätzliche Platz:

```
\paper {
  indent = #0
}

\new Staff \with {
```

```
  \remove Separating_line_group_engraver
} {
  c'1
  \break
  c'1
}
```
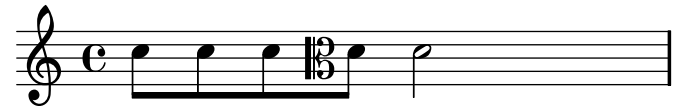


Nichtmusikalische Elemente wie Takt- und Tonartangaben, Schlüssel und Versetzungszeichen sind problematisch in proportionaler Notation. Keine dieser Elemente hat eine rhythmische Dauer, aber alle brauchen horizontalen Platz. Das Problem wird auf unterschiedliche Weise gelöst.
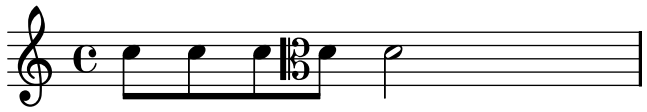
Es ist manchmal möglich, Probleme mit Tonarten zu lösen, indem keine benutzt werden. Das ist durchaus eine ernstzunehmende Option, weil die meisten Partituren mit proportionaler Notation für heutige Musik geschrieben werden. Ähnliches gilt für Taktarten, insbesondere, wenn ein Zeitstrahl in die Partitur eingearbeitet werden soll. In den meisten Partituren kommt jedoch irgendeine Taktart vor. Schlüssel und Versetzungszeichen sind noch wichtiger; auf sie kann selten verzichtet werden.

Eine Lösungsmöglichkeit ist es, die `strict-note-spacing`-Eigenschaft des `SpacingSpanner`-Objekts zu benutzen. Zum Vergleich die beiden Partituren unten:

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1 16)
  c''8
  c''8
  c''8
  \clef alto
  d'8
  d'2
}

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1 16)
  \override Score.SpacingSpanner #'strict-note-spacing = ##t
  c''8
  c''8
  c''8
  \clef alto
  d'8
  d'2
}
```





Bei beiden handelt es sich um proportionale Notation, aber die Platzverteilung im oberen Beispiel ist zu weit wegen des Schlüsselwechsels. Die Platzverteilung des zweiten Beispiels dagegen bleibt rhythmisch korrekt. `strict-note-spacing` bewirkt, dass Takt- und Tonartbezeichnungen, Schlüssel und Versetzungszeichen keine Rolle bei der Berechnung der Abstände spielen.

Zusätzlich zu den hier vorgestellten Einstellungen gibt es noch eine Reihe von Möglichkeiten, die oft in proportionaler Notation benutzt werden. Dazu gehören:

- `\override SpacingSpanner #'strict-grace-spacing = ##t`
- `tupletFullLength = ##t`
- `\override Beam #'breakable = ##t`
- `\override Glissando #'breakable = ##t`
- `\override TextSpanner #'breakable = ##t`
- `\remove Forbid_line_break_engraver in the Voice context`

Diese Einstellungen bewirken, dass auch Verzierungsnoten proportional gesetzt werden, dass Klammern von rhythmischen Gruppen bis zu den Anfangs- und Endpunkten ausgedehnt werden und lassen dehnbare Objekte wie Balken und Glissandi auch über Taktstriche hinweg zu.

Siehe auch

Notationsreferenz: Abschnitt 4.5.2 [Eine neuer Bereich mit anderen Abständen], Seite 499.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.6 Die Musik auf weniger Seiten zwingen

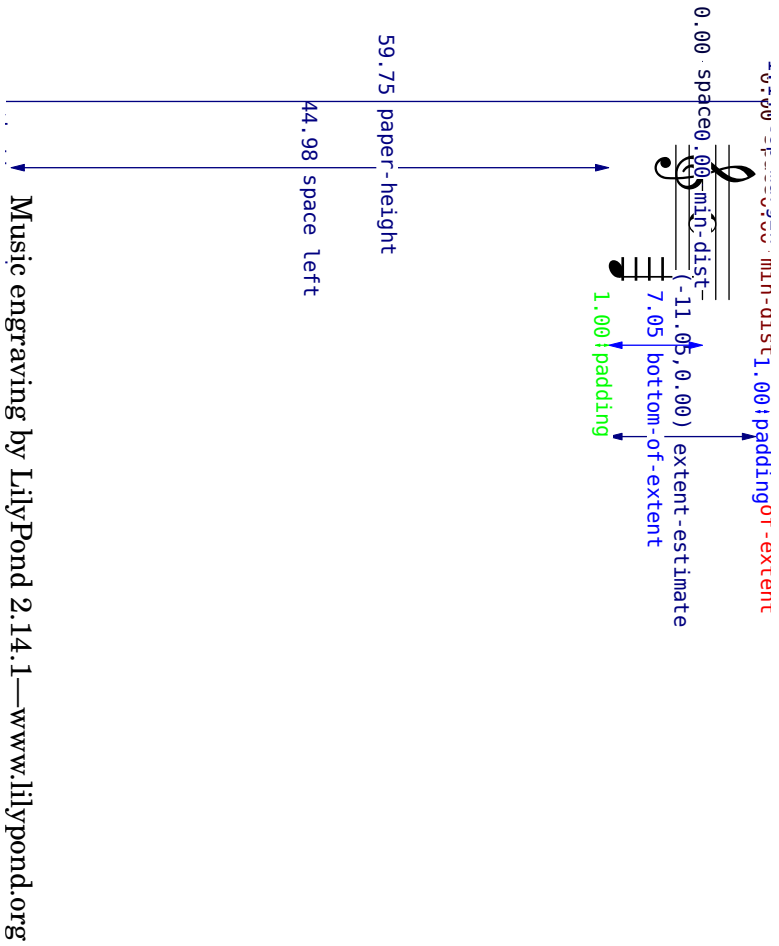
Manchmal kommt es vor, dass nur ein oder zwei Systeme auf die nächste Seite geraten, obwohl es so aussieht, als ob auf der vorigen Seite genügend Platz ist, um diese Systeme auch noch unterzubringen.

Wenn man derartige Platzierungsprobleme untersucht, ist die Funktion `annotate-spacing` von sehr großer Hilfe. Hiermit wird in den Musiksatz zusätzlich Information darüber ausgegeben, wieviel Platz bestimmten Parametern zugewiesen wird. Genaueres hierzu in Abschnitt 4.6.1 [Abstände anzeigen lassen], Seite 509.

4.6.1 Abstände anzeigen lassen

Die Dimensionen von vertikalen und horizontalen Platzierungsvariablen, die veränderbar sind, lassen sich mit ihren aktuellen Werten im Notentext anzeigen, wenn man die Funktion `annotate-spacing` in der `\paper`-Umgebung einschaltet:

```
#(set-default-paper-size "a6" 'landscape)
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



Alle Layoutdimensionen werden in Notenlinienzwischenräumen aufgelistet, unabhängig von den Einheiten, mit denen sie in der `\paper`- oder `\layout`-Umgebung definiert worden sind. In dem letzten Beispiel hat `paper-height` einen Wert von 59.75 Notenlinienzwischenräumen und `staff-size` Systemhöhe) ist 20 Punkte. Dabei gilt:

1 Punkt	= (25.4/72.27) mm
1 Notenlinienzwischenraum	= (`staff-size`)/4 pts = (`staff-size`)/4 * (25.4/72.27) mm

In diesem Fall ist ein `staff-space` (Notenlinienzwischenraum) etwa gleich 1.757 mm. Deshalb entspricht der Wert von 95.75 `staff-space` für `paper-height` (Papierhöhe) 105 mm, die Höhe eines quer gelegten A6-Papiers. Die Paare (*a*,*b*) sind Intervalle, wobei *a* der untere Rand und *b* der obere Rand des Intervalls.

Siehe auch

Notationsreferenz: Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

4.6.2 Abstände verändern

Die Ausgabe von `annotate-spacing` bietet sehr viele Details zu den vertikalen Dimensionen einer Partitur. Zu Information, wie Seitenränder und andere Layout-Variablen geändert werden können, siehe ⟨undefined⟩ [Seitenformatierung], Seite ⟨undefined⟩.

Neben Rändern gibt es einige weitere Optionen, Platz zu sparen:

- LilyPond kann die Systeme so dicht wie möglich platzieren (damit so viele Systeme wie möglich auf eine Seite passen), aber sie dann so anordnen, dass kein weißer Rand unten auf der Seite entsteht.

```
\paper {
  system-system-spacing = #'((padding . 0) (basic-distance . 0.1))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

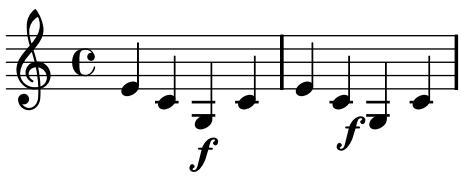
- Die Anzahl der Systeme kann erzwungen werden. Das kann auf zwei Arten helfen: wenn einfach nur ein Wert gesetzt wird, auch wenn es die gleiche Anzahl ist, die auch schon vorher von LilyPond erstellt wurde, kann manchmal dazu führen, dass mehr Systeme auf eine Seite gesetzt werden. Das liegt daran, dass ein Schritt im Notensatz ausgelassen wird, der die Seitenverteilung nur grob einschätzt, sodass eine bessere Seitenverteilung entsteht. Auch wenn man eine Verringerung der Anzahl an Systemen erzwingt, kann oft eine Seite eingespart werden. Wenn LilyPond die Musik etwa auf 11 Systeme verteilt, kann man die Benutzung von nur 10 Systemen erzwingen.

```
\paper {
  system-count = #10
}
```

- Vermeidung (oder Verminderung) von Objekten, die den vertikalen Abstand von Systemen vergrößern, hilft oft. Die Verwendung von Klammern bei Wiederholungen (oder alternativen Wiederholungen) etwa braucht mehr Platz. Wenn die Noten innerhalb der Klammern auf zwei Systeme verteilt sind, brauchen sie mehr Platz, als wenn sie nur auf einer Zeile gedruckt werden.

  Ein anderes Beispiel ist es, Dynamik-Zeichen, die besonders weit „hervorstehen“, zu verschieben.
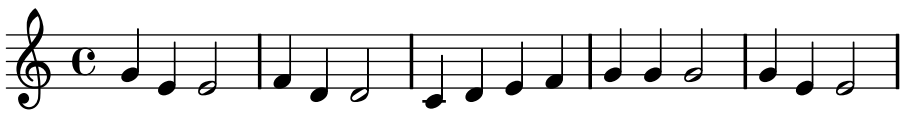
```
e4 c g\f c
e4 c g-\tweak #'X-offset #-2.7 -\tweak #'Y-offset #2.5 \f c
```



- Die horizontalen Abstände können mit der `SpacingSpanner`-Eigenschaft verändert werden. Siehe Abschnitt 4.5.3 [Horizontale Abstände verändern], Seite 499 für Einzelheiten. Dieses Beispiel zeigt die normalen Abstände:
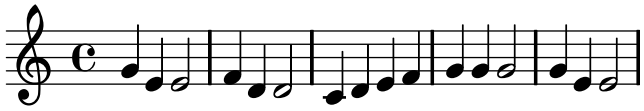
```
\score {
  \relative c'' {
    g4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
```

```
  }
}
```



Das nächste Beispiel verändert `common-shortest-duration` (die häufigste kürzeste Note) von `1/4` zu `1/2`. Die Viertelnote ist dennoch die häufigste Note in diesem Abschnitt, sodass der Notentext zusammengedrängt, wird, wenn eine Halbe als Standard angegeben wird:

```
\score {
  \relative c'' {
    g4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner
        #'common-shortest-duration = #(ly:make-moment 1 2)
    }
  }
}
```



Die `common-shortest-duration`-Eigenschaft kann nicht dynamisch verändert werden, darum muss sie immer in der `\context`-Umgebung definiert werden und wirkt sich somit auf eine ganze `\score`-Umgebung aus.

Siehe auch

Notationsreferenz: ⟨undefined⟩ [Seitenformatierung], Seite ⟨undefined⟩, Abschnitt 4.5.3 [Horizontale Abstände verändern], Seite 499.

Schnipsel: Abschnitt “Spacing” in *Schnipsel*.

5 Standardeinstellungen verändern

Das Ziel von LilyPonds Design ist es, von sich aus gut gesetzte Noten zu produzieren. Es kann aber trotzdem vorkommen, dass Sie diesen Standardsatz ändern wollen. Das Layout kann mithilfe einer recht großen Anzahl von „Schaltern und Knöpfen“ kontrolliert werden. Sie werden als „Eigenschaften“ (engl. properties) bezeichnet. Eine kurze Einführung und Übung, wie man auf diese Eigenschaften zugreifen kann und sie verändern kann, findet sich im Handbuch zum Lernen, siehe Abschnitt “Die Ausgabe verändern” in *Handbuch zum Lernen*. Das Kapitel sollte zuerst gelesen werden. In diesem Kapitel werden die gleichen Themen behandelt, aber der Schwerpunkt liegt eher auf einer technischen Darstellung.

Die definitive Beschreibung der unterschiedlichen Einstellmöglichenkeiten findet sich in einem eigenen Dokument: Abschnitt “der Referenz der Interna” in *Referenz der Interna*. Diese Referenz zeigt alle Variablen, Funktionen und Optionen, die in LilyPond möglich sind. Es existiert als ein HTML-Dokumente, das sich on-line, aber auch lokal in das LilyPond-Dokumentationspaket integriert lesen lässt.

Intern benutzt LilyPond Scheme (ein LISP-Dialekt), um eine Infrastruktur zur Verfügung zu stellen. Wenn Layoutentscheidungen verändert werden sollen, müssen auf die programminternen Prozesse zugegriffen werden, wozu Scheme-Code benötigt wird. Scheme-Abschnitte werden in einer LilyPond-Quelldatei mit einer Raute `#` begonnen (siehe auch Abschnitt “Scheme-Übung” in *Handbuch zum Lernen*).

5.1 Interpretationskontexte

Dieser Abschnitt erklärt, was Kontexte sind und wie man sie verändern kann.

Siehe auch

Handbuch zum Lernen: Abschnitt “Kontexte und Engraver” in *Handbuch zum Lernen*.

Installierte Dateien: ‘`ly/engraver-init.ly`’, ‘`ly/performer-init.ly`’.

Schnipsel: Abschnitt “Contexts and engravers” in *Schnipsel*.

Referenz der Interna: Abschnitt “Contexts” in *Referenz der Interna*, Abschnitt “Engravers and Performers” in *Referenz der Interna*.

5.1.1 Was sind Kontexte?

Kontexte sind hierarchisch geordnet:

Score – der Vater aller Kontexte

`Score` (Partitur) ist der höchste Notationskontext. Kein anderer Kontext kann einen `Score`-Kontext enthalten. Im Normalfall kümmert sich der `Score`-Kontext um die Verwaltung der Taktarten und sorgt dafür, dass Elemente wie Schlüssel und Taktart- oder Tonartbezeichnungen über die Systeme hinweg aneinander ausgerichtet sind.

Ein `Score`-Kontext wird eingerichtet, wenn eine `\score {...}` oder `\layout {...}`-Umgebung interpretiert wird.

Oberste Kontexte – Container für Systeme

Diese Kontexte fassen Systeme zu Gruppen zusammen und werden darum hier als Systemgruppen bezeichnet (engl. staffgroup).

StaffGroup

Gruppiert Systeme und fügt eine eckige Klammer auf der linken Seite hinzu. Die Taktstriche der enthaltenen Systeme werden vertikal miteinander verbunden. `StaffGroup` besteht nur aus

einer Ansammlung von Systemen mit einer eckigen Klammer zu Beginn der Zeile und durchgezogenen Taktstriche.

ChoirStaff

Entspricht `StaffGroup`, außer dass die Taktstriche der enthaltenen Systeme nicht vertikal miteinander verbunden sind.

GrandStaff

Gruppiert Systeme mit einer geschweiften Klammer zur Linken. Die Taktlinien der enthaltenen Systeme werden vertikal verbunden.

PianoStaff

Entspricht `GrandStaff`, hat aber zusätzlich Unterstützung für Instrumentenbezeichnungen zu Beginn jeder Systemgruppe.

Mittlere Kontexte – Systeme

Diese Kontexte stellen verschiedene Arten einzelner Notationssysteme (engl. staff) dar.

Staff

Kümmert sich um Schlüssel, Taktstriche, Tonarten und Versetzungszeichen. Er kann `Voice`-Kontexte enthalten.

RhythmicStaff

Entspricht `Staff`, aber dient zur Notation von Rhythmen: Tonhöhen werden ignoriert und die Noten auf einer einzigen Linie ausgegeben.

TabStaff

Ein Kontext um Tabulaturen zu erstellen. Die Standardeinstellung ist eine Gitarrentabulatur mit sechs Notenlinien.

DrumStaff

Ein Kontext zur Notation von Perkussion. Er kann `DrumVoice`-Kontexte enthalten.

VaticanaStaff

Entspricht `Staff`, aber eignet sich besonders zum Notensatz des Gregorianischen Chorals.

MensuralStaff

Entspricht `Staff`, aber eignet sich zum Notensatz von Noten in der Mensuralnotation.

Unterste Kontexte – Stimmen

Stimmen-(`Voice`-Kontexte initialisieren bestimmte Eigenschaften und laden bestimmte Engraver. Weil es sich bei Stimmen um die untersten Kontexte handelt, können sie keine weiteren Kontexte enthalten.

Voice

Entspricht einer Stimme auf einem Notensystem. Der Kontext kümmert sich um die Umsetzung von Noten, Dynamikzeichen, Hälsen, Balken, diversen Texten, Bögen und Pausen. Wenn mehr als eine Stimme pro System benötigt wird, muss dieser Kontext explizit initialisiert werden.

VaticanaVoice

Entspricht `Voice`, aber eignet sich besonders zum Notensatz des Gregorianischen Chorals.

MensuralVoice

Entspricht `Voice`, aber mit Änderungen, um Mensuralnotation setzen zu können.

Lyrics

Entspricht einer Stimme mit Gesangstext. Kümmert sich um den Satz des Gesangstextes auf einer Zeile.

DrumVoice

Der Stimmenkontext in einem Perkussionssystem.

FiguredBass

Der Kontext, in dem Generalbassziffern (`BassFigure`-Objekte) gesetzt werden, die in der `\figuremode`-Umgebung notiert werden.

TabVoice

Dieser Stimmenkontext wird in einer Tabulatur (`TabStaff`-Kontext) benutzt. Er wird normalerweise implizit erstellt.

CueVoice

Ein Stimmenkontext, der Noten in reduzierter Größe ausgibt und vor allem dazu da ist, Stichnoten zu setzen. Siehe auch [Stichnoten formatieren], Seite 185. Wird normalerweise implizit erstellt, wenn Stichnoten gesetzt werden.

ChordNames

Ausgabe von Akkordsymbolen.

5.1.2 Kontexte erstellen

In Partituren mit einer Stimme und einem System werden die Kontexte normalerweise automatisch erstellt. In komplizierteren Partituren muss man sie aber direkt erstellen. Es gibt drei Möglichkeiten, Kontexte zu erstellen:
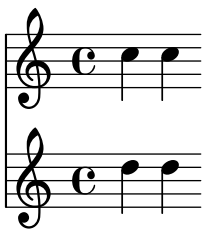
- Der einfachste Befehl ist `\new`. Er wird zusammen mit dem Kontextnamen vor einem musikalischen Ausdruck eingesetzt, etwa

  `\new` *`Kontext musik. Ausdruck`*

  wobei *Kontext* eine Kontextbezeichnung (wie `Staff` oder `Voice`) ist. Dieser Befehl erstellt einen neuen Kontext und beginnt mit der Auswertung von *musik. Ausdruck* innerhalb dieses Kontextes.

  Eine praktische Anwendung von `\new` ist eine Partitur mit vielen Systemen. Jede Stimme wird auf einem eigenen System notiert, das mit `\new Staff` begonnen wird.

  ```
  <<
    \new Staff { c4 c }
    \new Staff { d4 d }
  >>
  ```

  Der `\new`-Befehl kann den Kontext auch benennen:

  `\new` *`Kontext = ID musik. Ausdruck`*

  Dieser vom Benutzer definierte Name wird aber auch nur wirklich benutzt, wenn nicht vorher schon der gleiche Name definiert worden ist.

- Ähnlich dem `\new`-Befehl wird auch mit dem `\context`-Befehl ein musikalischer Ausdruck in einen Kontext umgeleitet. Diesem Kontext wird ein expliziter Name zugewiesen. Die Syntax lautet:

  `\context` *`Kontext = ID musik. Ausdruck`*

  Diese Art von Befehl sucht nach einem existierenden Kontext vom Typus *Kontext* mit der Bezeichnung *ID*. Wenn ein derartiger Kontext nicht existiert, wird ein neuer Kontext mit

der entsprechenden Bezeichnung erstellt. Das ist nützlich, wenn auf den Kontext später zurückverwiesen werden soll. Um etwa Gesangstext zu einer Melodie hinzuzufügen, wird die Melodie in einem bezeichneten Kontext notiert:

```
\context Voice = "Tenor" musik. Ausdruck
```

sodass der Text an den Noten ausgerichtet werden kann:
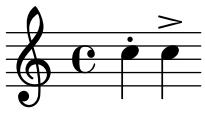
```
\new Lyrics \lyricsto "Tenor" Gesangstext
```

Eine andere Möglichkeit für bezeichnete Kontexte ist es, zwei unterschiedliche musikalische Ausdrücke in einen Kontext zu verschmelzen. Im nächsten Beispiel werden Artikulationszeichen und Noten getrennt notiert:

```
Noten = { c4 c4 }
Artik = { s4-. s4-> }
```

Dann werden sie kombiniert, indem sie dem selben `Voice`-Kontext zugewiesen werden:

```
<<
  \new Staff \context Voice = "A" \Noten
  \context Voice = "A" \Artik
>>
```



Durch diesen Mechanismus ist es möglich eine Urtextausgabe zu erstellen, mit der optionalen Möglichkeit, bestimmte zusätzliche Artikulationszeichen zu den gleichen Noten hinzuzufügen und so eine editierte Ausgabe zu erhalten.

- Der dritte Befehl, um Kontexte zu erstellen, ist:

  ```
  \context Kontext musik. Ausdruck
  ```

  Dies entspricht dem `\context` mit `= ID`, aber hier wird ein beliebiger Kontext des Typs *Kontext* gesucht und der musikalische Ausdruck darin ausgewertet, unabhängig von der Bezeichnung, die dem Kontext gegeben wurde.

  Diese Variante wird bei musikalischen Ausdrücken benutzt, die auf verschiedenen Ebenen interpretiert werden können. Beispielsweise der `\applyOutput`-Befehl (siehe ⟨undefined⟩ [Eine Funktion auf alle Layout-Objekte anwenden], Seite ⟨undefined⟩). Ohne einen expliziten `\context` wird die Ausgabe normalerweise einem `Voice`-Kontext zugewiesen:

  ```
  \applyOutput #'Kontext #Funktion   % auf Voice anwenden
  ```

  Damit aber die Funktion auf `Score`- oder `Staff`-Ebene interpretiert wird, muss folgende Form benutzt werden:

  ```
  \applyOutput #'Score #Funktion
  \applyOutput #'Staff #Funktion
  ```

5.1.3 Kontexte am Leben halten

Kontexte werden normalerweise am ersten musikalischen Moment beendet, an dem sie nichts mehr zu tun haben. Ein `Voice`-Kontext stirbt also sofort, wenn keine Ereignisse mehr auftreten, `Staff`-Kontexte sobald alle in ihnen enthaltenen `Voice`-Kontexte keine Ereignisse mehr aufweisen usw. Das kann Schwierigkeiten ergeben, wenn auf frühere Kontexte verwiesen werden soll, die in der Zwischenzeit schon gestorben sind, beispielsweise wenn man Systemwechsel mit `\change`-Befehlen vornimmt, wenn Gesangstext einer Stimme mit dem `\lyricsto`-Befehl zugewiesen wird oder wenn weitere musikalische Ereignisse zu einem früheren Kontext hinzugefügt werden sollen.

Es gibt eine Ausnahme dieser Regel: genau ein `Voice`-Kontext innerhalb eines `Staff`-Kontextes oder in einer `<<...>>`-Konstruktion bleibt immer erhalten bis zum Ende des `Staff`-Kontextes oder der `<<...>>`-Konstruktion, der ihn einschließt, auch wenn es Abschnitte gibt, in der er nichts zu tun hat. Der Kontext, der erhalten bleibt ist immer der erste, der in der ersten enthaltenden `{...}`-Konstruktion angetroffen wird, wobei `<<...>>`-Konstruktionen ignoriert werden.

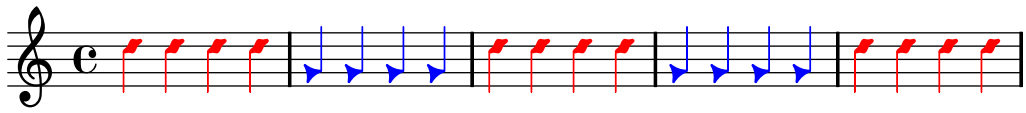
Jeder Kontext kann am Leben gehalten werden, indem man sicherstellt dass er zu jedem musikalischen Moment etwas zu tun hat. `Staff`-Kontexte werden am Leben gehalten, indem man sicherstellt, dass eine der enthaltenen Stimmen am Leben bleibt. Eine Möglichkeit, das zu erreichen, ist es, unsichtbare Pause zu jeder Stimme hinzuzufügen, die am Leben gehalten werden soll. Wenn mehrere Stimmen sporadisch benutzt werden sollen, ist es am sichersten, sie alle am Leben zu halten und sich nicht auf die Ausnahmeregel zu verlassen, die im vorigen Abschnitt dargestellt wurde.

Im folgenden Beispiel werden sowohl Stimme A als auch B auf diese Weise für die gesamte Dauer des Stückes am Leben gehalten.

```
musicA = \relative c'' { d4 d d d }
musicB = \relative c'' { g4 g g g }
keepVoicesAlive = {
  <<
    \new Voice = "A" { s1*5 }  % Keep Voice "A" alive for 5 bars
    \new Voice = "B" { s1*5 }  % Keep Voice "B" alive for 5 bars
  >>
}

music = {
  \context Voice = "A" {
    \voiceOneStyle
    \musicA
  }
  \context Voice = "B" {
    \voiceTwoStyle
    \musicB
  }
  \context Voice = "A" { \musicA }
  \context Voice = "B" { \musicB }
  \context Voice = "A" { \musicA }
}

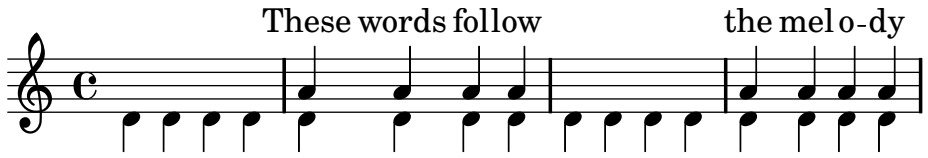
\score {
  \new Staff <<
    \keepVoicesAlive
    \music
  >>
}
```



Das nächste Beispiel zeigt eine Melodie, die zeitweise unterbrochen wird und wie man den entsprechenden Gesangstext mit ihr verknüpfen kann, indem man die Stimme am Leben hält.

In wirklichen Situationen würden Begleitung und Melodie natürlich aus mehreren Abschnitten bestehen.

```
melody = \relative c'' { a4 a a a }
accompaniment = \relative c' { d4 d d d }
words = \lyricmode { These words fol -- low the mel -- o -- dy }
\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          s1*4  % Keep Voice "melody" alive for 4 bars
        }
        {
          \new Voice = "accompaniment" {
            \voiceTwo
            \accompaniment
          }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
          \context Voice = "accompaniment" { \accompaniment }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = #"music" }
    \lyricsto "melody" { \words }
  >>
}
```

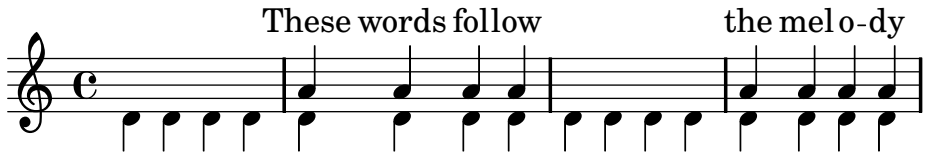


Eine Alternative, die in manchen Umständen besser geeignet sein kann, ist es, einfach unsichtbare Pausen einzufügen, um die Melodie mit der Begleitung passend auszurichten:

```
melody = \relative c'' {
  s1  % skip a bar
  a4 a a a
  s1  % skip a bar
  a4 a a a
}
accompaniment = \relative c' {
  d4 d d d
  d4 d d d
```

```
  d4 d d d
  d4 d d d
}
words = \lyricmode { These words fol -- low the mel -- o -- dy }

\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          \melody
        }
        \new Voice = "accompaniment" {
          \voiceTwo
          \accompaniment
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = #"music" }
    \lyricsto "melody" { \words }
  >>
}
```



5.1.4 Umgebungs-Plugins verändern

Notationskontexte (wie `Score` oder `Staff`) speichern nicht nur Eigenschaften, sie enthalten auch Plugins („engraver“ genannt), die die einzelnen Notationselemente erstellen. Ein `Voice`-Kontext enthält beispielsweise einen `Note_heads_engraver`, der die Notenköpfe erstellt, und ein `Staff`-Kontext einen `Key_signature_engraver`, der die Vorzeichen erstellt.

Eine vollständige Erklärung jedes Plugins findet sich in Referenz der Interna: ↦ Translation ↦ Engravers. Alle Kontexte sind erklärt in Referenz der Interna: ↦ Translation ↦ Context, wobei die in diesem Kontext vorkommenden Engraver aufgelistet sind.
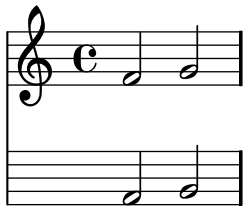
Es kann teilweise nötig sein, diese Engraver umzupositionieren. Das geschieht, indem man einen neuen Kontext mit `\new` oder `\context` beginnt und ihn dann verändert:

```
\new context \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ..Noten..
}
```

... steht hier für die Bezeichnung des Engravers. `\consists` fügt einen Engraver hinzu und `\remove` entfernt ihn. Es folgt ein einfaches Beispiel, in dem der `Time_signature_engraver`

(Engraver für den Takt) und der `Clef_engraver` (Engraver für den Schlüssel) aus dem `Staff`-Kontext entfernt werden:

```
<<
  \new Staff {
    f2 g
  }
  \new Staff \with {
     \remove "Time_signature_engraver"
     \remove "Clef_engraver"
  } {
    f2 g2
  }
>>
```

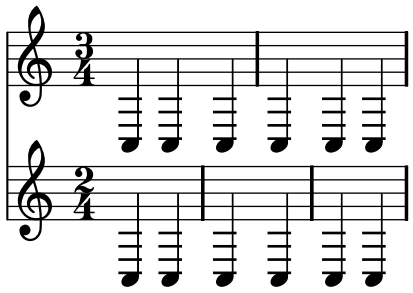


Das zweite Notensystem enthält keine Taktangabe und keinen Notenschlüssel. Das ist eine recht brutale Methode, Objekte zu verstecken, weil es sich auf das gesamte System auswirkt. Diese Methode beeinflusst auch die Platzaufteilung, was erwünscht sein kann. Vielfältigere Methoden, mit denen Objekte unsichtbar gemacht werden können, finden sich in Abschnitt "Sichtbarkeit und Farbe von Objekten" in *Handbuch zum Lernen*.

Das nächste Beispiel zeigt eine Anwendung in der Praxis. Taktstriche und Taktart werden normalerweise in einer Partitur synchronisiert. Das geschieht durch `Timing_translator` und `Default_bar_line_engraver`. Diese Plugins sorgen sich um die Verwaltung der Taktzeiten und die Stelle innerhalb des Taktes, zu dem eine Note erscheint usw. Indem man diese Engraver aus dem `Score`-Kontext in den `Staff`-Kontext verschiebt, kann eine Partitur erstellt werden, in welcher jedes System eine unterschiedliche Taktart hat:

```
\score {
  <<
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    } {
        \time 3/4
        c4 c c c c c
    }
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  } {
      \time 2/4
      c4 c c c c c
  }
>>
\layout {
  \context {
    \Score
```

```
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    }
  }
}
```



Bekannte Probleme und Warnungen

Normalerweise spielt es keine Rolle, in welcher Reihenfolge Engraver angegeben werden, aber in einigen Spezialfällen ist die Reihenfolge sehr wichtig. Das kann beispielsweise vorkommen, wenn ein Engraver eine Eigenschaft erstellt und ein anderer von ihr liest, oder ein Engraver erstellt ein Grob und ein anderer wertet es aus. Die Reihenfolge, in der Engraver angegeben werden, ist die Reihenfolge, in der sie aufgerufen werden, um ihre Tätigkeiten auszuführen.

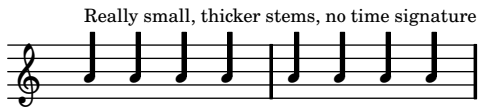
Folgende Reihenfolgen müssen beachtet werden: der `Bar_engraver` muss normalerweise zuerst kommen, und der `New_fingering_engraver` muss vor dem `Script_column_engraver` kommen. Es gibt möglicherweise weitere Abhängigkeiten von der Reihenfolge geben.

5.1.5 Die Standardeinstellungen von Kontexten ändern

Die Kontexteinstellungen, die standardmäßig in `Score`, `Staff` und `Voice`-Kontexten benutzt werden, können in einer `\layout`-Umgebung eingestellt werden, wie das folgende Beispiel zeigt. Die `\layout`-Umgebung sollte innerhalb der `\score`-Umgebung gesetzt werden, auf die sie sich auswirken soll, aber außerhalb von Notation.

Auch muss der `\set`-Befehl und der Kontext weggelassen werden, wenn die Einstellungen für den Kontext auf diese Weise vorgenommen werden:

```
\score {
  \relative c'' {
    a4^"Really small, thicker stems, no time signature" a a a
    a a a a
  }
  \layout {
    \context {
      \Staff
      fontSize = #-4
      \override Stem #'thickness = #4.0
      \remove "Time_signature_engraver"
    }
  }
}
```



Hier zeigt der `\Staff`-Befehl an, dass die folgenden Einstellungen sich auf alle Systeme in dieser Partitur erstrecken sollen.

Veränderungen können auch für den `Score`- oder alle `Voice`-Kontexte auf gleiche Weise vorgenommen werden.

Bekannte Probleme und Warnungen

Es ist nicht möglich, Kontextänderungen als Variable zu definieren und sie dann in der `\context`-Definition anzuwenden, indem man die Variable aufruft.

Der Befehl `\Staff \RemoveEmptyStaves` überschreibt die aktuellen Einstellungen für `Staff`. Wenn die Einstellungen für Systeme verändert werden sollen, die `\Staff \RemoveEmptyStaves` benutzen, müssen die Veränderungen gemacht werden, nachdem `\Staff \RemoveEmptyStaves` aufgerufen wurde, etwa:

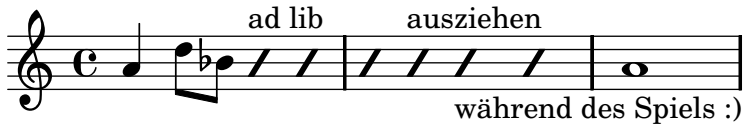
```
\layout {
  \context {
    \Staff \RemoveEmptyStaves

    \override Stem #'thickness = #4.0
  }
}
```

5.1.6 Neue Kontexte definieren

Bestimme Kontexte, wie `Staff` oder `Voice`, werden erstellt, indem man sie mit einer Musikumgebung aufruft. Es ist aber auch möglich, eigene neue Kontexte zu definieren, in denen dann unterschiedliche Engraver benutzt werden.

Das folgende Beispiel zeigt, wie man etwa `Voice`-Kontexte von Grund auf neu bauen kann. Ein derartiger Kontext ähnelt `Voice`, es werden aber nur zentrierte Schrägstriche als Notenköpfe ausgegeben. Das kann benutzt werden, um Improvisation in Jazzmusik anzuzeigen.



Diese Einstellungen werden innerhalb der `\context`-Umgebung innerhalb der `\layout`-Umgebung definiert:

```
\layout {
  \context {
    ...
  }
}
```

Der Beispielcode des folgenden Abschnittest muss anstelle der Punkte im vorigen Beispiel eingesetzt werden.

Zuerst ist es nötig eine Bezeichnung für den neuen Kontext zu definieren:

```
\name ImproVoice
```

Weil dieser neue Kontext ähnlich wie `Voice` ist, sollen die Befehle, die in `Voice`-Kontexten funktionieren, auch in dem neuen Kontext funktionieren. Das wird erreicht, indem der Kontext als Alias `Voice` erhält:

```
\alias Voice
```

Der Kontext gibt Noten und Text aus, darum müssen wir die Engraver hinzufügen, die für diese Aktionen zuständig sind:

```
\consists Note_heads_engraver
\consists Text_engraver
```

aber die Noten sollen nur auf der mittleren Linie ausgegeben werden:

```
\consists Pitch_squash_engraver
squashedPosition = #0
```

Der `Pitch_squash_engraver` verändert Notenköpfe (die vom `Note_heads_engraver` erstellt werden) und setzt ihre vertikale Position auf den Wert von `squashedPosition`, in diesem Fall ist das die Mittellinie.

Die Noten sehen wie ein Querstrich aus und haben keine Hälse:

```
\override NoteHead #'style = #'slash
\override Stem #'transparent = ##t
```

Alle diese Engraver müssen zusammenarbeiten, und das wird erreicht mit einem zusätzlichen Plugin, das mit dem Befehl `\type` gekennzeichnet werden muss. Dieser Typ solle immer `Engraver_group` lauten:

```
\type "Engraver_group"
```

Alles zusammen haben wir folgende Einstellungen:

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists Pitch_squash_engraver
  squashedPosition = #0
  \override NoteHead #'style = #'slash
  \override Stem #'transparent = ##t
  \alias Voice
}
```

Kontexte sind hierarchisch. Wie wollen, dass `ImproVoice` sich als Unterkontext von `Staff` erkennt, wie eine normale Stimme. Darum wird die Definition von `Staff` mit dem `\accepts`-Befehl verändert:

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Das Gegenteil von `\accepts` ist `\denies` (verbietet), was manchmal gebraucht werden kann, wenn schon existierende Kontext-Definitionen wieder benutzt werden sollen.

Beide Definitionen müssen in die `\layout`-Umgebung geschrieben werden:
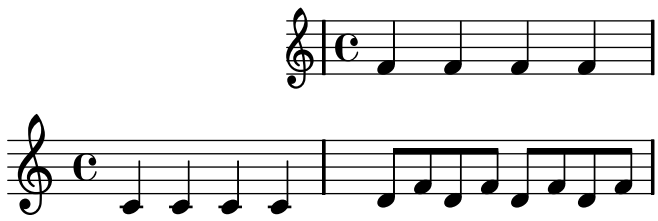
```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \context {
    \Staff
    \accepts "ImproVoice"
  }
}
```

Jetzt kann die Notation zu Beginn des Abschnitts folgendermaßen notiert werden:

```
\relative c'' {
  a4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"ausziehen"
    c c_"während des Spielens :)"
  }
  a1
}
```
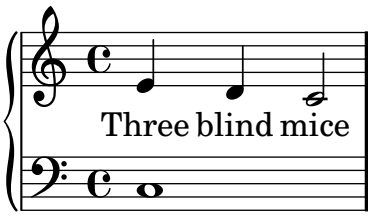
5.1.7 Kontexte aneinander ausrichten

Neue Kontexte können über oder unter existierenden ausgerichtet werden. Das kann nützlich sein, wenn man eine Chorpartitur oder Ossia schreiben will:



Kontexte wie `PianoStaff` können andere Kontexte innerhalb enthalten. Kontexte, die als innere Kontexte akzeptiert werden, werden in einer „accepts“-Liste für den bestimmten Kontext definiert. Kontexte, die sich nicht in dieser Liste finden, werden unter den äußeren Kontext gesetzt. Der `PianoStaff`-Kontext etwa akzeptiert die Kontexte `Staff` und `FiguredBass` innerhalb, aber beispielsweise keinen `Lyrics`-(Gesangstext)-Kontext. In dem folgenden Beispiel wird deshalb der Gesangstext unter das gesamte Klaviersystem gesetzt, anstatt zwischen die beiden Notensysteme zu kommen:
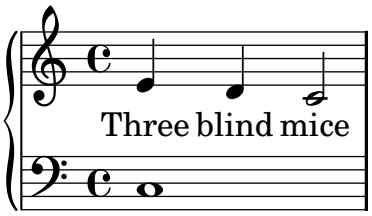
```
\new PianoStaff
<<
  \new Staff { e4 d c2 }
  \addlyrics { Three blind mice }
  \new Staff {
    \clef "bass"
    { c,1 }
  }
>>
```



Die „accepts“-Liste eines Kontextes kann verändert werden, so dass sie weitere innere Kontexte akzeptiert. Wenn also der Gesangstext als Teil eines Klaviersystems gesetzt werden soll, müsste man schreiben:

```
\new PianoStaff \with { \accepts Lyrics }
<<
  \new Staff { e4 d c2 }
  \addlyrics { Three blind mice }
```

```
  \new Staff {
    \clef "bass"
    { c,1 }
  }
>>
```



Das Gegenteil von `\accepts` ist `\denies`; es bedeutet, dass ein Kontext aus der `\accepts`-Liste gestrichen wird.
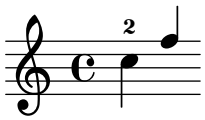
5.2 Die Referenz der Programminterna erklärt

5.2.1 Zurechtfinden in der Programmreferenz

Arbeit mit der Referenz der Interna soll hier an einigen Beispiel illustriert werden. Die Referenz der Interna existiert nur auf Englisch, darum sind auch die Beispiele dieses Abschnittes nicht übersetzt.

Folgende Aufgabe wird bearbeitet: Der Fingersatz aus dem Beispiel unten soll verändert werden:

```
c-2
\stemUp
f
```



In der Dokumentation über Fingersatz (in [Fingersatzanweisungen], Seite 191) gibt es folgenden Abschnitt:

Siehe auch:

Referenz der Interna: Abschnitt “Fingering” in *Referenz der Interna*.

Die Referenz der Interna gibt es als HTML-Dokument. Sie sollten sie als HTML-Dokument lesen, entweder online oder indem Sie die HTML-Dokumentation herunterladen. Dieser Abschnitt ist sehr viel schwieriger zu verstehen, wenn Sie die PDF-Version verwenden.

Gehen Sie über diesen Link zum Abschnitt Abschnitt “Fingering” in *Referenz der Interna*. Oben auf der Seite findet sich:

Fingering objects are created by: Abschnitt “Fingering_engraver” in *Referenz der Interna* and Abschnitt “New_fingering_engraver” in *Referenz der Interna*.

Indem Sie die Links in der Referenz der Interna folgen, können Sie verfolgen, wie LilyPond intern arbeitet:

- Abschnitt “Fingering” in *Referenz der Interna*: Abschnitt “Fingering” in *Referenz der Interna* objects are created by: Abschnitt “Fingering_engraver” in *Referenz der Interna*
- Abschnitt “Fingering_engraver” in *Referenz der Interna*: Music types accepted: Abschnitt “fingering-event” in *Referenz der Interna*

- Abschnitt “fingering-event” in *Referenz der Interna*: Music event type `fingering-event` is in Music expressions named Abschnitt “FingeringEvent” in *Referenz der Interna*

Fingersatz-Objekte werden also durch den `Fingering_engraver` erstellt, welcher folgende Musikereignistypen akzeptiert: `fingering-event`. Ein Musikereignis vom Typ `fingering-event` ist ein musikalischer Ausdruck mit der Bezeichnung Abschnitt “FingeringEvent” in *Referenz der Interna*.

Dieser Pfad geht genau die entgegengesetzte Richtung von LilyPonds Wirkungsweise: er beginnt bei der graphischen Ausgabe und arbeitet sich voran zur Eingabe. Man könnte auch mit einem Eingabe-Ereignis starten und dann die Links zurückverfolgen, bis man zum Ausgabe-Objekt gelangt.

Die Referenz der Interna kann auch wie ein normales Dokument durchsucht werden. Sie enthält Kapitel über `Music definitions`, über Abschnitt “Translation” in *Referenz der Interna* und Abschnitt “Backend” in *Referenz der Interna*. Jedes Kapitel listet alle die Definitionen und Eigenschaften auf, die benutzt und verändert werden können.

5.2.2 Layout-Schnittstellen

Die HTML-Seite, die im vorigen Abschnitt betrachtet wurde, beschreibt ein Layoutobjekt mit der Bezeichnung `Fingering`. Ein derartiges Objekt ist ein Symbol in der Partitur. Es hat Eigenschaften, die bestimmte Zahlen speichern (wie etwa Dicke und Richtung), aber auch Weiser auf verwandte Objekte. Ein Layoutobjekt wird auch als „Grob“ bezeichnet, die Abkürzung für *Gr*aphisches *Ob*jekt. Mehr Information zu Grobs findet sich in Abschnitt “grob-interface” in *Referenz der Interna*.

Die Seite zu `Fingering` enthält Definitionen für das `Fingering`-Objekt. Auf der Seite steht etwa:

`padding` (dimension, in staff space):

`0.5`

was bedeutet, dass der Abstand zu anderen Objekten mindestens 0.5 Notenlinienabstände beträgt.

Jedes Layoutobjekt kann mehrere Funktionen sowohl als typographisches als auch als Notationselement einnehmen. Das Fingersatzobjekt beispielsweise hat folgende Aspekte:

- Seine Größe ist unabhängig von der horizontalen Platzaufteilung, anders als etwa bei Legatobögen.
- Es handelt sich um Text, normalerweise sehr kurz.
- Dieser Text wird durch ein Glyph einer Schriftart gesetzt, anders als bei Legatobögen.
- Der Mittelpunkt des Symbols sollte horizontal mit dem Mittelpunkt des Notenkopfes ausgerichtet werden.
- Vertikal wird das Objekt neben die Note und das Notensystem gesetzt.
- Die vertikale Position wird auch mit anderen Textelementen abgeglichen.

Jeder dieser Aspekte findet sich in sogenannten Schnittstellen (engl. interface), die auf der Abschnitt “Fingering” in *Referenz der Interna*-Seite unten aufgelistet sind:

This object supports the following interfaces: Abschnitt “item-interface” in *Referenz der Interna*, Abschnitt “self-alignment-interface” in *Referenz der Interna*, Abschnitt “side-position-interface” in *Referenz der Interna*, Abschnitt “text-interface” in *Referenz der Interna*, Abschnitt “text-script-interface” in *Referenz der Interna*, Abschnitt “font-interface” in *Referenz der Interna*, Abschnitt “finger-interface” in *Referenz der Interna*, and Abschnitt “grob-interface” in *Referenz der Interna*.

Ein Klick auf einen der Links öffnet die Seite der entsprechenden Schnittstelle. Jede Schnittstelle hat eine Anzahl von Eigenschaften. Einige sind nicht vom Benutzer zu beeinflussen („interne Eigenschaften“), andere aber können verändert werden.

Es wurde immer von einem `Fingering`-Objekt gesprochen, aber eigentlich handelt es sich nicht um sehr viel. Die Initialisierungsdatei '`scm/define-grobs.scm`' zeigt den Inhalt dieses „Objekts“ (zu Information, wo diese Dateien sich finden siehe Abschnitt “Mehr Information” in *Handbuch zum Lernen*):
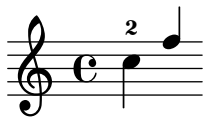
```
(Fingering
  . ((padding . 0.5)
     (avoid-slur . around)
     (slur-padding . 0.2)
     (staff-padding . 0.5)
     (self-alignment-X . 0)
     (self-alignment-Y . 0)
     (script-priority . 100)
     (stencil . ,ly:text-interface::print)
     (direction . ,ly:script-interface::calc-direction)
     (font-encoding . fetaText)
     (font-size . -5)  ; don't overlap when next to heads.
     (meta . ((class . Item)
     (interfaces . (finger-interface
                    font-interface
                    text-script-interface
                    text-interface
                    side-position-interface
                    self-alignment-interface
                    item-interface))))))
```

Wie man sehen kann, ist das Fingersatzobjekt nichts anderes als eine Ansammlung von Variablen, und die Internetseite der Referenz der Interna ist direkt aus diesen Anweisungen generiert.

5.2.3 Die Grob-Eigenschaften

Die Position der **2** aus dem Beispiel unten soll also geändert werden:

```
c-2
\stemUp
f
```



Weil die **2** vertikal an der zugehörigen Note ausgerichtet ist, müssen wir uns mit der Schnittstelle auseinander setzen, die diese Positionierung veranlasst. Das ist hier `side-position-interface`. Auf der Seite für diese Schnittstelle heißt es:

`side-position-interface`

Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

Darunter wird die Variable `padding` (Verschiebung) beschrieben:
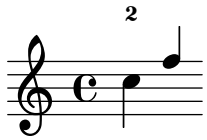
`padding` (dimension, in staff space)
Add this much extra space between objects that are next to each other.

Indem man den Wert von `padding` erhöht, kann die Fingersatzanweisung weiter weg von der Note gesetzt werden. Dieser Befehl beispielsweise fügt drei Notenlinienzwischenräume zwischen die Zahl und den Notenkopf:

```
\once \override Voice.Fingering #'padding = #3
```

Wenn dieser Befehl in den Quelltext eingefügt wird, bevor der Fingersatz notiert ist, erhält man folgendes:

```
\once \override Voice.Fingering #'padding = #3
c-2
\stemUp
f
```



In diesem Fall muss die Veränderung speziell für den `Voice`-Kontext definiert werden. Das kann auch aus der Referenz der Interna entnommen werden, da die Seite des Abschnitt "Fingering_engraver" in *Referenz der Interna* schreibt:

Fingering_engraver is part of contexts: ... Abschnitt "Voice" in *Referenz der Interna*

5.2.4 Benennungskonventionen

Die Bezeichnungen für Funktionen, Variablen, Engraver und Objekte folgen bestimmten Regeln:

- Scheme-Funktionen: kleinbuchstaben-mit-bindestrichen
- Scheme-Funktionen: ly:plus-scheme-stil
- Musikalische Ereignisse, Musikklassen und Musikeigenschaften: wie-scheme-funktionen
- Grob-Schnittstellen: scheme-stil
- backend-Eigenschaften: scheme-stil (aber X und Y)
- Kontexte: Großbuchstabe, oder GroßbuchstabeZwischenWörtern (CamelCase)
- Kontext-Eigenschaften: kleinbuchstabeMitFolgendenGroßbuchstaben
- Engraver: Großbuchstabe_gefolgt_von_kleinbuchstaben_mit_unterstrichen

5.3 Eigenschaften verändern

5.3.1 Grundlagen zum Verändern von Eigenschaften

Jeder Kontext ist verantwortlich für die Erstellung bestimmter graphischer Objekte. Die Einstellungen für diese Objekte werden auch in dem Kontext gespeichert. Wenn man diese Einstellungen verändert, kann die Erscheinung der Objekte geändert werden.

Es gibt zwei unterschiedliche Eigenschaftenarten, die in Kontexten gespeichert werden: Kontexteigenschaften und Grob-Eigenschaften. Kontexteigenschaften sind Eigenschaften, die sich auf den gesamten Kontext beziehen und seine Darstellung beinflussen. Grob-Eigenschaften dagegen wirken sich nur auf bestimmte graphische Objekte aus, die in einem Kontext dargestellt werden.

Die `\set`- und `\unset`-Befehle werden benutzt, um die Werte von Kontexteigenschaften zu ändern. Die Befehle `\override` und `\revert` hingegen verändern die Werte von Grob-Eigenschaften.

Siehe auch

Referenz der Interna: Abschnitt “OverrideProperty” in *Referenz der Interna*, Abschnitt “RevertProperty” in *Referenz der Interna*, Abschnitt “PropertySet” in *Referenz der Interna*, Abschnitt “Backend” in *Referenz der Interna*, Abschnitt “All layout objects” in *Referenz der Interna*.

Bekannte Probleme und Warnungen

Das Back-end ist nicht sehr streng bei der Überprüfung der Typen von Objekteigenschaften. Auf sich selbst verweisende Bezüge in Scheme-Werten der Eigenschaften können Verzögerung oder einen Absturz des Programms hervorrufen.

5.3.2 Der `\set`-Befehl

Jeder Kontext kann unterschiedliche *Eigenschaften* besitzen, Variablen, die in diesem Kontext definiert sind. Sie können während der Interpretation des Kontextes verändert werden. Hierzu wird der `\set`-Befehl eingesetzt:

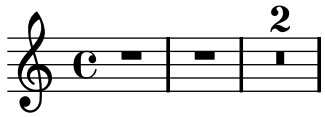
```
\set Kontext.Eigenschaft = #Wert
```

Wert ist ein Scheme-Objekt, weshalb ihm `#` vorangestellt werden muss.

Kontexteigenschaften werden üblicherweise mit `kleinGroßbuchstabe` benannt. Sie kontrollieren vor allem die Übersetzung von Musik in Notation, wie etwa `localKeySignature`, welche bestimmt, wann ein Taktstrich gesetzt werden muss. Kontexteigenschaften können ihren Wert mit der Zeit ändern, während eine Notationsdatei interpretiert wird. Ein gutes Beispiel dafür ist `measurePosition`, was die Position der Noten im Takt angibt. Kontexteigenschaften werden mit dem `\set`-Befehl verändert.
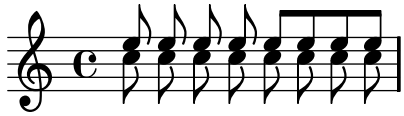
Mehrtaktpausen etwa können in einen Takt zusammengefasst werden, wenn die Kontexteigenschaft `skipBars` (Takte überspringen) auf `#t` (wahr) gesetzt wird:

```
R1*2
\set Score.skipBars = ##t
R1*2
```



Wenn das *Kontext*-Argument ausgelassen wird, bezieht sich der Befehl auf den gerade aktiven unterstmöglichen Kontext, üblicherweise `ChordNames`, `Voice` oder `Lyrics`.
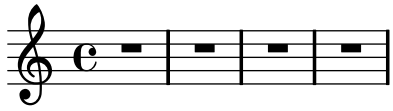
```
\set Score.autoBeaming = ##f
<<
  {
    e8 e e e
    \set autoBeaming = ##t
    e8 e e e
  } \\ {
    c8 c c c c8 c c c
  }
>>
```



Die Änderung wird zur Laufzeit während der Musik interpretiert, sodass diese Einstellung sich nur auf die zweite Gruppe von Achteln auswirkt.

Dabei gilt zu beachten, dass der unterste Kontext nicht immer die Eigenschaft enthält, die verändert werden soll. Wenn man beispielsweise `skipBars` aus dem oberen Beispiel ohne Angabe des Kontextes zu verändern sucht, hat der Befehl keine Auswirkung, weil er sich auf den `Voice`-Kontext bezieht, die Eigenschaft sich aber im `Score`-Kontext befindet:

```
R1*2
\set skipBars = ##t
R1*2
```



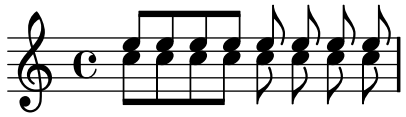
Kontexte sind hierarchisch angeordnet. Wenn ein übergeordneter Kontext angegeben wird, etwa `Staff`, dann beziehen sich die Änderungen auf alle Stimmen (`Voice`), die in diesem Kontext enthalten sind.

Es gibt auch einen `\unset`-Befehl:

`\unset` *`Kontext.Eigenschaft`*

der bewirkt, dass die vorgenommenen Definitionen für *Eigenschaft* entfernt werden. Dieser Befehl macht nur Einstellungen im richtigen Kontext rückgängig. Wenn also im `Staff`-Kontext die Bebalkung ausgeschaltet wird:

```
\set Score.autoBeaming = ##t
<<
  {
    \unset autoBeaming
    e8 e e e
    \unset Score.autoBeaming
    e8 e e e
  } \\ {
    c8 c c c c8 c c c
  }
>>
```
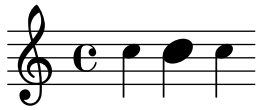


Wie für `\set` muss das *Kontext*-Argument für den untersten Kontext nicht mitangegeben werden. Die zwei Versionen

```
\set Voice.autoBeaming = ##t
\set autoBeaming = ##t
```

verhalten sich gleich, wenn die gegenwärtige Basis der `Voice`-Kontext ist.

Einstellungen, die nur einmal vorgenommen werden sollen, können mit `\once` notiert werden, etwa:

```
c4
\once \set fontSize = #4.7
c4
c4
```



Eine vollständige Beschreibung aller vorhandenen Kontexteigenschaften findet sich in der Referenz der Interna, siehe „Translation ↦ Tunable context properties“.

Siehe auch

Internals Reference:

Abschnitt “Tunable context properties” in *Referenz der Interna*.

5.3.3 Der \\override-Befehl

Es gibt eine besondere Art von Kontexteigenschaft: die Grob-Beschreibung. Grob-Beschreibungen werden mit `GroßGroßbuchstabe` benannt. Sie enthalten „Standardeinstellungen“ für ein bestimmtes Grob als eine assoziative Liste. Siehe ‘`scm/define-grobs.scm`’ für die Einstellungen aller Grob-Beschreibungen. Grob-Beschreibungen werden mit `\override` verändert.

`\override` ist eigentlich eine Kurzform, der Befehl

```
\override Kontext.GrobBezeichnung #'Eigenschaft = #Wert
```
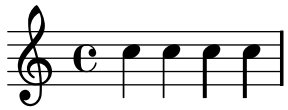
ist äquivalent zu

```
\set Kontext.GrobBezeichnung  =
  #(cons (cons 'Eigenschaft Wert)
         <vorheriger Wert von Kontext.GrobBezeichnung>)
```

Der Wert von `Kontext.GrobBezeichnung` (die assoz. Liste „alist“) wird benutzt um die Eigenschaften von individuellen Grobs zu initialisieren. Grobs haben Eigenschaften, die im Scheme-Stil mit `bindestrich-wörtern` benannt sind. Diese Werte der Grob-Eigenschaften verändern sich während des Notensetzens: LilyPonds Notensatz heißt im Grunde, die Eigenschaften mit Callback-Fuktionen auszurechnen.
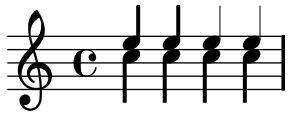
Beispielsweise kann die Dicke eines Notenhalses verändert werden, indem man die `thickness`-Eigenschaft des `Stem`-Objekts verändert:

```
c4 c
\override Voice.Stem #'thickness = #3.0
c4 c
```
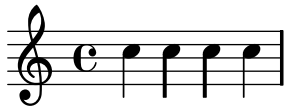


Wenn kein Kontext angegeben wird, wird der tiefste aktuelle Kontext benutzt:

```
{ \override Staff.Stem #'thickness = #3.0
  <<
    {
      e4 e
      \override Stem #'thickness = #0.5
      e4 e
    } \\ {
      c4 c c c
    }
  >>
}
```
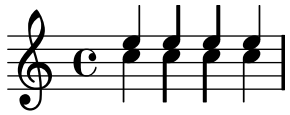


Die Auswirkungen von `\override` können mit `\revert` wieder rückgängig gemacht werden:

```
c4
\override Voice.Stem #'thickness = #3.0
c4 c
\revert Voice.Stem #'thickness
c4
```
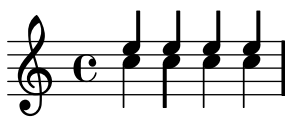


Die Auswirkungen von `\override` und `\revert` wirken sich auf alle Grobs im entsprechenden Kontext aber der Stelle aus, an der sie gesetzt werden:

```
{
  <<
    {
      e4
      \override Staff.Stem #'thickness = #3.0
      e4 e e
    } \\ {
      c4 c c
      \revert Staff.Stem #'thickness
      c4
    }
  >>
}
```



`\once` kann zusammen mit `\override` benutzt werden, um nur den aktuellen Zeitwert zu verändern:

```
{
  <<
    {
      \override Stem #'thickness = #3.0
      e4 e e e
    } \\ {
      c4
      \once \override Stem #'thickness = #3.0
      c4 c c
    }
  >>
}
```



Siehe auch

Referenz der Interna: Abschnitt “Backend” in *Referenz der Interna*.

5.3.4 Der \\tweak-Befehl

Wenn man Grob-Eigenschaften mit `\override` verändert, verändern sich alle fraglichen Objekte zu dem gegebenen musikalischen Moment. Manchmal will man allerdings nur ein Grob verändern, anstatt allen Grobs des aktuellen Kontextes. Das kann mit dem `\tweak`-Befehl erreicht werden, mit dem man Optimierungen vornehmen kann:

```
\tweak #'grob-eigenschaft #Wert
```

Der `\tweak`-Befehl wirkt sich auf das Objekt aus, dass direkt auf `Wert` folgt.

Eine Einleitung der Syntax und Benutzungen des `\tweak`-(Optimierungs)-Befehls findet sich in Abschnitt “Optimierungsmethoden” in *Handbuch zum Lernen*.

Wenn mehrere gleichartige Elemente zum gleichen musikalischen Moment auftreten, kann der `\override`-Befehl nicht benutzt werden, um nur einen von ihnen zu verändern: hier braucht man den `\tweak`-Befehl. Elemente, die mehrfach zum gleichen musikalischen Moment auftreten können sind unter Anderem:

- Notenköpfe von Noten innerhalb eines Akkordes
- Artikulationszeichen an einer einzelnen Note
- Bindebögen zwischen Noten eines Akkordes
- Llammern für rhythmische Verhältnisse (wie Triolen), die zur gleichen Zeit beginnen

In diesem Beispiel wird die Farbe eines Notenkopfes und die Art eines anderen Notenkopfes innerhalb eines Akkordes verändert:

```
< c
  \tweak #'color #red
  d
  g
  \tweak #'duration-log #1
  a
> 4
```
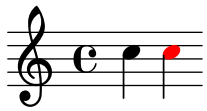


`\tweak` kann auch benutzt werden, um Bögen zu verändern:

```
c-\tweak #'thickness #5 ( d e f)
```



Damit der `\tweak`-Befehl funktioniert, muss er direkt vor dem Objekt stehen, auf das er sich bezieht. Manchmal kommt es vor, dass LilyPond während der Kompilierung der Datei zusätzliche Elemente einfügt, die dann zwischen der Optimierung und dem Objekt stehen. Noten, auch einzlene Noten, werden beispielsweise intern von LilyPond immer wie Akkorde behandelt, sodass auch ein `\tweak`-Befehl für eine einzelne Note innerhalb von Akkordzeichen notiert werden muss:
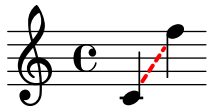
```
\tweak #'color #red c4
<\tweak #'color #red c>4
```



Der `\tweak`-Befehl kann *nicht* eingesetzt werden, um Elemente zu verändern, die sich nicht direkt im Notentext befinden. Insbesondere Hälse, Balken oder Versetzungszeichen lassen sich nicht beeinflussen, weil diese später durch den Notenkopf erstellt werden und nicht direkt durch den Quelltext. `\tweak` kann auch nicht verwendet werden, um Schlüssel oder Taktarten zu verändern, denn sie werden von dem `\tweak`-Befehl während der Interpretation durch automatisches Einfügen von zusätzlichen Kontextelementen getrennt.

Mehrere `\tweak`-Befehle können vor ein Notationselement gesetzt werden und alle werden interpretiert:

```
c
-\tweak #'style #'dashed-line
-\tweak #'dash-fraction #0.2
-\tweak #'thickness #3
-\tweak #'color #red
 \glissando
f'
```



Der Strom der musikalischen Ereignisse (engl. music stream), der aus dem Quelltext erstellt wird, und zu dem auch die automatisch eingefügten Elemente gehören, kann betrachtet werden, siehe ⟨undefined⟩ [Musikalische Funktionen darstellen], Seite ⟨undefined⟩. Das kann nützlich sein, wenn man herausfinden will, was mit dem `\tweak`-Befehl verändert werden kann.

Siehe auch

Handbuch zum Lernen: Abschnitt “Optimierungsmethoden” in *Handbuch zum Lernen*.

Erweitern: Abschnitt “Musikalische Funktionen darstellen” in *Extending*.

Bekannte Probleme und Warnungen

Der `\tweak`-Befehl kann nicht innerhalb von einer Variable eingesetzt werden.

Der `\tweak`-Befehl kann nicht innerhalb von `\lyricmode` eingesetzt werden.

Der `\tweak`-Befehl kann nicht benutzt werden, um die Kontrollpunkte eines von mehreren Bindebögen eines Akkorden zu verändern. Anstelle dessen wird der erste Bogen verändert, der in der Eingabedatei auftritt.

5.3.5 `\\set` versus `\\override`

5.3.6 Alisten verändern

Einige vom Benutzer einstellbare Eigenschaften sind intern als *alists* (Assoziative Listen) dargestellt, die Paare von Schlüsseln und Werten speichern. Die Struktur einer Aliste ist:

```
'((Schlüssel1 . Wert1)
  (Schlüssel2 . Wert2)
  (Schlüssel3 . Wert3)
```

```
...)
```

Wenn eine Aliste eine Grob-Eigenschaft oder eine Variable der `\paper`-Umgebung ist, können ihre Schlüssel einzeln verändert werden, ohne andere Schlüssel zu beeinflussen.

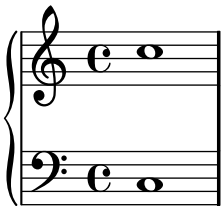
Um beispielsweise den Freiraum zwischen benachbarten Systemen in einer Systemgruppe zu verkleinern, kann man die `staff-staff-spacing`-Eigenschaft des `+StaffGrouper`-Grobs benutzen. Die Eigenschaft ist eine Aliste mit vier Schlüsseln: : `basic-distance` (Grund-Abstand), `minimum-distance` (minimaler Abstand), `padding` (Verschiebung) und `stretchability` (Dehnbarkeit). Die Standardwerte dieser Eigenschaft finden sich im Abschnitt „Backend“ der Referenz der Interna (siehe Abschnitt “StaffGrouper” in *Referenz der Interna*):

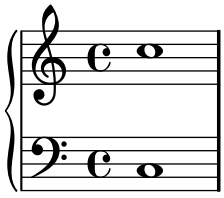
```
'((basic-distance . 9)
  (minimum-distance . 7)
  (padding . 1)
  (stretchability . 5))
```

Eine Möglichkeit, die Systemen dichter zueinander zu zwingen, ist es, der Wert des `basic-distance`-Schlüssels (`9`) zu verändern, sodass der den gleichen Wert wie `minimum-distance` (`7`) hat. Um einen einzelnen Schlüssel zu verändern, wird ein geschachtelter Aufruf benutzt:

```
% default space between staves
\new PianoStaff <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>

% reduced space between staves
\new PianoStaff \with {
  % this is the nested declaration
  \override StaffGrouper #'staff-staff-spacing #'basic-distance = #7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



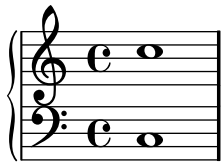


Wenn man diese Art des geschachtelten Aufrufs einsetzt, wird der spezifische Schlüssel (`basic-distance` im obigen Beispiel) verändert, ohne dass sich andere Wert für die gleiche Eigenschaft ändern würden.

Nun sollen die Systeme so dicht wie möglich gesetzt werden, ohne das Überlappungen vorkommen. Die einfachste Möglichkeit, das zu tun, wäre es, alle vier Wert auf 0 zu setzen. Man muss

jedoch nicht vier Werte definieren, sondern die Eigenschaft kann mit einem Aufruf als Aliste vollständig verändert werden:

```
\new PianoStaff \with {
  \override StaffGrouper #'staff-staff-spacing =
    #'((basic-distance . 0)
       (minimum-distance . 0)
       (padding . 0)
       (stretchability . 0))
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



Dabei sollte beachtet werden, dass alle Schlüssel, die bei dieser Weise des Aufrufs nicht explizit aufgelistet sind, auf den Standardwert gesetzt werden, den sie hätten, wenn sie nicht definiert werden. Im Falle von `staff-staff-spacing` würden alle nicht genannten Schlüsselwerte auf 0 gesetzt (außer `stretchability`, welche immer den Wert von `space` hat, wenn sie nicht definiert ist). Somit sind folgende Aufrufe äquivalent:

```
\override StaffGrouper #'staff-staff-spacing =
  #'((basic-distance . 7))

\override StaffGrouper #'staff-staff-spacing =
  #'((basic-distance . 7)
     (minimum-distance . 0)
     (padding . 0)
     (stretchability . 7))
```

Eine möglicherweise ungewollte Konsequenz hiervon ist, dass alle Standardwerte, die etwa in einer Initialisierungsdatei zu Beginn einer LilyPond-Partitur geladen werden, nach dem Aufruf rückgängig gemacht werden. Im obigen Beispiel werden die initialisierten Standardwerte für `padding` und `minimum-distance` (definiert in '`scm/define-grobs.scm`') auf den Standard zurückgesetzt, den sie uninitialisiert hätten (0 in beiden Fällen). Wenn eine Eigenschaft oder Variable in Form einer Aliste (jeder Größe) definiert wird, werden immer alle Schlüsselwerte auf den uninitialisierten Zustand zurückgesetzt. Es ist also sicherer, geschachtelte Aufrufe zu benutzen, wenn man nicht bewusst alle Werte zurücksetzen will.

Achtung: Geschachtelte Aufrufe funktionieren nicht mit Kontexteigenschaften (wie etwa `beamExceptions`, `keySignature`, `timeSignatureSettings`, usw.) Diese Eigenschaften können nur verändert werden, indem man sie vollständig als Alisten umdefiniert.

5.4 Nützliche Konzepte und Eigenschaften

5.4.1 Eingabe-Modi

Die Art, wie die Notation einer Eingabedatei interpretiert wird, hängt vom aktuellen Eingabemodus ab.

Chord (Akkordmodus)

Man erreicht ihn durch den Befehl `\chordmode`. Hierdurch wird die Eingabe entsprechend der Syntax der Akkordnotation interpretiert, siehe Abschnitt 2.7 [Notation von Akkorden], Seite 371. Akkorde werden als Noten auf einem System dargestellt.

Der Akkordmodus wird auch mit dem Befehl `\chords` initiiert. Dadurch wird gleichzeitig ein neuer `ChordNames`-Kontext erstellt, die Eingabe entsprechend der Syntax der Akkordnotation interpretiert und als Akkordbezeichnungen in einem `ChordNames`-Kontext dargestellt. Siehe [Akkordbezeichnungen drucken], Seite 376.

Drum (Schlagzeugmodus)

Man erreicht ihn mit dem Befehl `\drummode`. Die Eingabe wird entsprechend der Syntax der Schlagzeugnotation interpretiert, siehe [Grundlagen der Schlagzeugnotation], Seite 349.

Der Schlagzeugmodus wird auch mit dem Befehl `\drums` aktiviert. Dadurch wird gleichzeitig ein neuer `DrumStaff`-Kontext erstellt, die Eingabe entsprechend der Syntax der Schlagzeugnotation interpretiert und als Schlagzeugsymbole auf einem Schlagzeugsystem dargestellt. Siehe [Grundlagen der Schlagzeugnotation], Seite 349.

Figure (Ziffernmodus)

Man erreicht ihn mit dem Befehl `\figuremode`. Die Eingabe wird entsprechend der Syntax für Generalbass interpretiert, siehe [Eingabe des Generalbass'], Seite 384.

Der Ziffernmodus wird auch mit dem Befehl `\figures` aktiviert. Dadurch wird gleichzeitig ein neuer `FiguredBass`-Kontext erstellt, die Eingabe entsprechend der Syntax für Generalbass interpretiert und als Generalbassziffern im `FiguredBass`-Kontext dargestellt. Siehe [Grundlagen des Bezifferten Basses], Seite 384.

Fret/tab (Griffsymbol-/Tabulaturmodus)

Es gibt keinen besonderen Eingabemodus für Griffsymbole und Tabulaturen.

Um Tabulaturen zu erstellen, werden Noten oder Akkorde im Notenmodus notiert und dann in einem `TabStaff`-Kontext interpretiert, siehe [Standardtabulaturen], Seite 305.

Um Griffsymbole oberhalb eines Notensystems zu erstellen, gibt es zwei Möglichkeiten. Man kann den `FretBoards`-Kontext einsetzen (siehe [Automatische Bund-Diagramme], Seite 340) oder sie können als Beschriftung über den Noten eingefügt werden, indem man den `\fret-diagram`-Befehl einsetzt (siehe [Bund-Diagramm-Beschriftung], Seite 321).

Lyrics (Gesangstextmodus)

Man erreicht ihn mit dem Befehl `\lyricmode`. Die Eingabe wird entsprechend der Syntax für Silben eines Gesangstextes interpretiert, wobei optional Dauern und verknüpfte Gesangstextveränderer möglich sind, siehe Abschnitt 2.1 [Notation von Gesang], Seite 226.

Der Gesangstextmodus wird auch durch den Befehl `\addlyrics` aktiviert. Dadurch wird auch ein neuer `Lyrics`-Kontext erstellt und ein impliziter `\lyricsto`-Befehl, der den nachfolgenden Gesangstext mit der vorhergehenden Musik verknüpft.

Markup (Textbeschriftungsmodus)

Man erreicht ihn mit dem Befehl `\markup`. Die Eingabe wird entsprechend der Syntax für Textbeschriftung interpretiert, siehe Abschnitt A.9 [Textbeschriftungsbefehle], Seite 597.

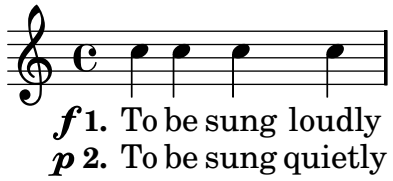
Note (Notenmodus)

Das ist der Standardmodus. Er kann auch mit dem Befehl `\notemode` gefordert werden. Die Eingabe wird als Tonhöhen, Dauern, Beschriftung usw. interpretiert und als musikalische Notation auf einem Notensystem gesetzt.

Es ist normalerweise nicht nötig, den Notenmodus extra anzugeben, aber es kann in bestimmten Situationen durchaus nützlich sein, etwa wenn man in einem Gesangstext-, Akkord- oder einem anderen Modus arbeitet aber ein Zeichen braucht, das nur im Notenmodus benutzt werden kann.

Um etwa Dynamikzeichen vor die Nummern von unterschiedlichen Strophen zu setzen, muss man den Notenmodus betreten:

```
{ c4 c4 c4 c4 }
\addlyrics {
  \notemode{ \set stanza = \markup{ \dynamic f 1. } }
  To be sung loudly
}
\addlyrics {
  \notemode{ \set stanza = \markup{ \dynamic p 2. } }
  To be sung quietly
}
```



5.4.2 Richtung und Platzierung

Die Platzierung und Richtung von Objekten ist im Notensatz oft durch eine enge Auswahl begrenzt: Notenhälse beispielsweise können entweder nach oben oder nach unten zeigen, Gesangstext, Dynamikzeichen und andere Ausdrucksbezeichnungen können über oder unter dem System gesetzt werden, Text kann rechts, links oder mittig ausgerichtet werden usw. Die meisten dieser Entscheidungen können LilyPond direkt überlassen werden; in einigen Fällen kann es allerdings nötig sein, eine bestimmte Richtung oder eine Position zu erzwingen.

Richtungseinstellung von Artikulationszeichen
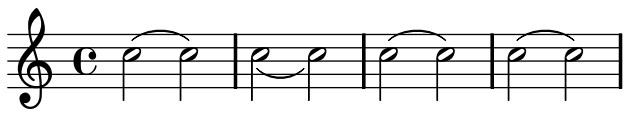
Standardmäßig sind bestimmte Objekte immer nach oben oder unten ausgerichtet, wie Dynamikzeichen oder Fermaten, während andere Objekte zwischen oben und unten wechseln, was vor allem von der Richtung der Notenhälse abhängt und etwa Bögen und Akzente betrifft.

Die Standardeinstellungen können verändert werden, indem dem Artikulationszeichen ein Ausrichtungsmarkierer vorangeht. Drei derartige Ausrichtungsmarkierer sind vorhanden: `^` (bedeutet „nach oben“), `_` (bedeutet „nach unten“) bzw. `-` (bedeutet „Standardrichtung“ benutzen) normalerweise weggelassen werden. In diesem Fall wird `-` angenommen. Eine Richtungsanweisung ist jedoch **immer** erforderlich vor

- `\tweak`-Befehlen
- `\markup`-(Textbeschriftungs-)Befehlen
- `\tag`-Befehlen
- Textbeschriftungen in reiner Textform, wie etwa `-"string"`
- Fingersatzanweisungen: `-1`
- Abkürzungen von Artikulationen, wie `-.`, `->`, `--`

Ausrichtungsmarkierer haben nur eine Auswirkung auf die nächste Note:

```
c2( c)
c2_( c)
c2( c)
c2^( c)
```



Die `direction`-(Richtungs-)Eigenschaft

Die Position oder Richtung vieler Layoutobjekte wird von der `direction`-Eigenschaft kontrolliert.

Der Wert der `direction`-Eigenschaft kann auf den Wert `1` gesetzt werden, was gleichbedeutend mit „nach oben“ bzw. „oberhalb“ ist, oder auf den Wert `-1`, was „nach unten“ bzw. „unterhalb“ bedeutet. Die Symbole `UP` und `DOWN` können anstelle von `1` und `-1` benutzt werden. Die Standardausrichtung kann angegeben werden, indem `direction` auf den Wert `0` oder `CENTER` gesetzt wird. In vielen Fällen bestehen auch vordefinierte Befehle, mit denen die Ausrichtung bestimmt werden kann. Sie haben die Form

`\xxxUp`, `xxxDown`, `xxxNeutral`

wobei `xxxNeutral` bedeutet: „Benutze die Standardausrichtung“. Siehe auch Abschnitt “within-staff (Objekte innerhalb des Notensystems)” in *Handbuch zum Lernen*.

In wenigen Fällen, von denen Arpeggio das einzige häufiger vorkommende Beispiel darstellt, entscheidet der Wert von `direction`, ob das Objekt auf der rechten oder linken Seite das Ursprungsobjektes ausgegeben wird. In diesem Fall bedeutet `-1` oder `LEFT` „auf der linken Seite“ und `1` oder `RIGHT` „auf der rechten Seite“. `0` oder `CENTER` bedeutet „benutze Standardausrichtung“.

Diese Ausrichtungsanzeigen wirken sich auf alle Noten aus, bis sie rückgängig gemacht werden:

```
c2( c)
\slurDown
c2( c)
c2( c)
\slurNeutral
c2( c)
```



5.4.3 Reihenfolge des Kontextlayouts

Kontexte werden normalerweise in einer Notensystemgruppe dargestellt, von oben nach unten in der Reihenfolge, in der sie in der Eingabedatei auftreten. Wenn Kontexte verschachtelt sind, enthält der äußere Kontext die inneren geschachtelten Kontexte, wie in der Eingabedatei angegeben, vorausgesetzt die inneren Kontexte befinden sich in der „accepts“-Liste des äußeren Kontextes. Verschachtelte Kontexte, die nicht in dieser Liste auftauchen, werden neu unter den äußeren Kontext angeordnet, anstatt dass sie innerhalb dieses Kontextes gesetzt werden.

Es ist wichtig zu erinnern, dass ein Kontext implizit erstellt werden kann, wenn ein Befehl vorkommt und kein passender Kontext zur Verfügung steht, um den Befehl auszuführen. Dadurch können unerwartet neue Systeme oder Partituren erstellt werden.

Die Standardreihenfolge, in der die Kontexte gesetzt werden und die „accepts“-Liste können geändert werden, siehe auch ⟨undefined⟩ [Aligning contexts], Seite ⟨undefined⟩.

Siehe auch

Handbuch zum Lernen Abschnitt “An extra staff appears” in *Handbuch zum Lernen*.

5.4.4 Abstände und Maße

In LilyPond gibt es zwei Arten von Abständen: absolute und skalierte.

Absolute Abstände werden benutzt, um Ränder, Einzüge und andere Einzelheiten des Seitenlayouts zu bestimmen. Sie sind in den Standardeinstellungen in Millimetern definiert. Abstände können auch in anderen Einheiten definiert werden, indem folgende Befehle auf die Zahl folgen: `\mm`, `\cm`, `\in` (Zoll=2,54 cm) und `\pt` (Punkte, 1/72.27 eines Zolls). Abstände des Seitenlayouts können auch in skalierbaren Einheiten (siehe folgenden Absatz) definiert werden, indem man den Befehl `\staff-space` an die Zahl hängt. Das Seitenlayout ist genauer beschrieben in Abschnitt 4.1 [Seitenlayout], Seite 461.

Skalierbare Abstände werden immer in Einheiten von Notenlinienabständen angegeben, oder seltener in halben Notenlinienabständen. Ein Notenlinienabstand ist der Abstand zwischen zwei benachbarten Linien eines Notensystems. Der Standardwert dieser Einheit kann global geändert werden, indem man die globale Notensystemgröße ändert, oder sie kann lokal geändert werden, indem man die Eigenschaft `staff-space` des `StaffSymbol`-Objekts mit `\override` verändert. Skalierte Abstände verändern sich automatisch entsprechend, wenn der Notenlinienabstand entweder global oder lokal verändert wird, aber Schriftarten verändern ihre Größe nur, wenn der Notenlinienabstand global verändert wird. Mit dem globalen Notenlinienabstand kann man also auf einfach Art und Weise die gesamte Größe einer Partitur verändern. Zu Methoden, wie der globale Notenlinienabstand verändert werden kann, siehe Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472.

Wenn nur eine ein Abschnitt einer Partitur in einer anderen Größe erscheinen soll, etwa ein Ossia-Abschnitt in einer Fußnote, kann die globale Notensystemgröße nicht einfach geändert werden, weil sich diese Änderung auf die gesamte Partitur auswirken würde. In derartigen Fällen muss die Größenänderung vorgenommen werden, indem man sowohl die `staff-space`-Eigenschaft von `StaffSymbol` als auch die Größe der Schriftarten verändert. Eine Scheme-Funktion, `magstep`, kann von einer Schriftartveränderung zu der entsprechenden Veränderung in `staff-space` (Notenlinienabständen) konvertieren. Zu einer Erklärung und Beispielen zu ihrer Verwendung siehe Abschnitt “Länge und Dicke von Objekten” in *Handbuch zum Lernen*.

Siehe auch

Handbuch zum Lernen: Abschnitt “Länge und Dicke von Objekten” in *Handbuch zum Lernen*.

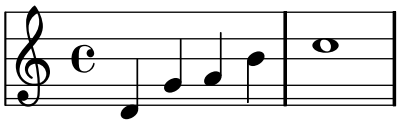
Notationsreferenz: Abschnitt 4.1 [Seitenlayout], Seite 461, Abschnitt 4.2.2 [Die Notensystemgröße einstellen], Seite 472.

5.4.5 Eigenschaften des Staff-Symbols

Die vertikale Position der Notenlinien und die Anzahl der Notenlinien kann gleichzeitig definiert werden. Wie das folgende Beispiel zeigt, werden Notenpositionen nicht durch die Position der Notenlinien verändert:
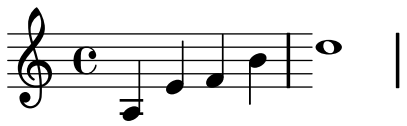
Achtung: Die `'line-positions`-Eigenschaft verändert die `'line-count`-Eigenschaft. Die Anzahl der Notenlinien wird implizit definiert durch die Anzahl der Elemente in der Liste der Werte von `'line-positions`.

```
\new Staff \with {
  \override StaffSymbol #'line-positions = #'(7 3 0 -4 -6 -7)
}
{ a4 e' f b | d1 }
```



Die Breite eines Notensystems kann verändert werden. Die Einheit ist in Notenlinienabständen. Die Abstände von Objekten in diesem Notensystem wird durch diese Einstellung nicht beeinflusst.

```
\new Staff \with {
  \override StaffSymbol #'width = #23
}
{ a4 e' f b | d1 }
```



5.4.6 Strecker

Viele Objekte der Musiknotation erstrecken sich über mehrere Objekte oder gar mehrere Takte. Beispiele hierfür sind etwa Bögen, Balken, Triolenklammern, Volta-Klamern in Wiederholungen, Crescendo, Triller und Glissando. Derartige Objekte werden als „Strecker“ bezeichnet. Sie haben spezielle Eigenschaften, mit welchen ihre Eigenschaften und ihr Verhalten beeinflusst werden kann. Einige dieser Eigenschaften gelten für alle Strecker, andere beschränken sich auf eine Untergruppe der Strecker.

Alle Strecker unterstützen das `spanner-interface` (Strecker-Schnittstelle). Ein paar, insbesondere die, die zwischen zwei Objekten eine gerade Linie ziehen, unterstützen auch das `line-spanner-interface` (Strecker-Linienschnittstelle).
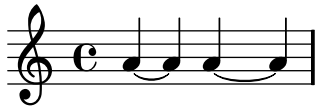
Das `spanner-interface` benutzen

Diese Schnittstelle stellt zwei Eigenschaften zur Verfügung, die sich auf mehrere Strecker auswirken:

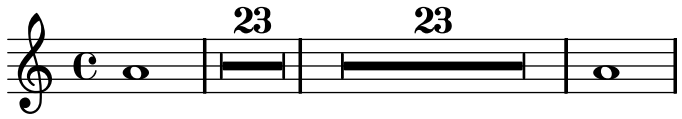
Die `minimum-length`*-Eigenschaft*

Die Mindestlänge eines Streckers wird durch die `minimum-length`-Eigenschaft definiert. Wenn diese Eigenschaft vergrößert wird, muss in den meisten Fällen auch der Abstand der Noten zwischen den zwei Endpunkten eines Streckers verändert werden. Eine Veränderung dieser Eigenschaft hat jedoch auf die meisten Strecker keine Auswirkung, weil ihre Länge aus anderen Berechnungen hervorgeht. Einige Beispiele, wo die Eigenschaft benutzt wird, sind unten dargestellt.
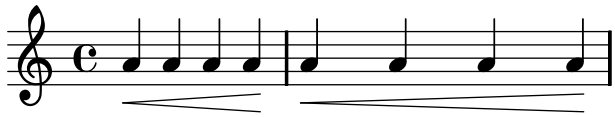
```
a~a
a
% increase the length of the tie
-\tweak #'minimum-length #5
~a
```



```
a1
\compressFullBarRests
R1*23
% increase the length of the rest bar
\once \override MultiMeasureRest #'minimum-length = #20
R1*23
a1
```




```
a \< a a a \!
% increase the length of the hairpin
\override Hairpin #'minimum-length = #20
a \< a a a \!
```



Diese Veränderung kann auch eingesetzt werden, um die Länge von Legato- und Phrasierungsbögen zu verändern:

```
a( a)
a
-\tweak #'minimum-length #5
( a)

a\( a\)
a
-\tweak #'minimum-length #5
\( a\)
```
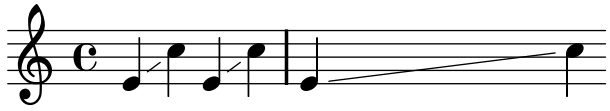


Im Falle einiger Layoutobjekte wirkt sich die `minimum-length`-Eigenschaft erst dann aus, wenn die `set-spacing-rods`-Prozedur explizit aufgerufen wird. Um das zu tun, sollte die `springs-and-rods`-Eigenschaft auf `ly:spanner::set-spacing-rods` gesetzt werden. Die Mindestlänge eines Glissandos etwa wird erst aktiv, wenn die `springs-and-rods`-Eigenschaft gesetzt ist:

```
% default
e \glissando c'

% not effective alone
\once \override Glissando #'minimum-length = #20
e, \glissando c'
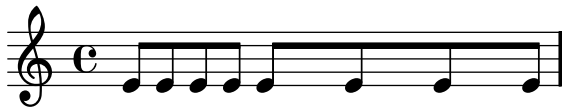
% effective only when both overrides are present
\once \override Glissando #'minimum-length = #20
\once \override Glissando #'springs-and-rods = #ly:spanner::set-spacing-rods
e, \glissando c'
```



Das gilt auch für das `Beam`-(Balken-)Objekt:

```
% not effective alone
\once \override Beam #'minimum-length = #20
e8 e e e
```

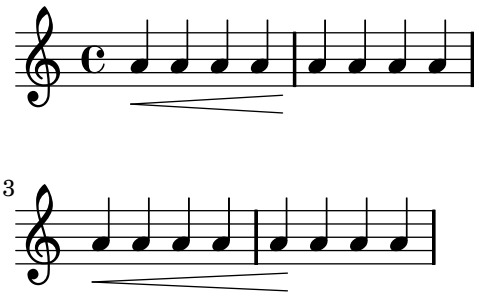
```
% effective only when both overrides are present
\once \override Beam #'minimum-length = #20
\once \override Beam #'springs-and-rods = #ly:spanner::set-spacing-rods
e8 e e e
```



Die `to-barline`*-Eigenschaft*

Die zweite nützliche Eigenschaft des `spanner-interface` ist `to-barline` (bis zum Taktstrich). In den Standardeinstellungen ist diese Eigenschaft auf „wahr“ gesetzt, was bedeutet, dass ein Strecker, etwa eine Crescendo-Klammer, der an der ersten Noten eines Taktes beendet wird, sich nur bis zum vorhergehenden Taktstrich erstreckt. Wenn die Eigenschaft auf „falsch“ gesetzt wird, erstrecken sich die Strecker entsprechend über die Taktlinie hinüber und enden erst an der entsprechenden Note:

```
a \< a a a a \! a a a \break
\override Hairpin #'to-barline = ##f
a \< a a a a \! a a a
```



Diese Eigenschaft wirkt sich nicht auf alle Strecker aus. Im Falle von Legato- oder Phrasierungsbögen etwa hat diese Eigenschaft keinen Effekt. Das gilt auch für alle anderen Streckern, bei denen es nicht sinnvoll währe, sie an einer Taktlinie abzuschließen.
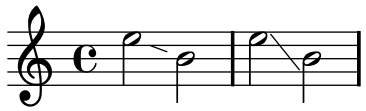
Das `line-spanner-interface` benutzen

Objekte, die das `line-spanner-interface` unterstützen, sind unter Anderem:

- `DynamicTextSpanner`
- `Glissando`
- `TextSpanner`
- `TrillSpanner`
- `VoiceFollower`

Die Routine, die das Setzen der Matrizen dieser Strecker hervorruft, ist `ly:line-interface::print`. Diese Routine bestimmt die exakte Position der zwei Endpunkte und zeichnet eine Linie zwischen ihnen, in dem erforderlichen Stil. Die Position der zwei Endpunkte des Streckers wird in Echtzeit errechnet, aber es ist möglich, ihre Y-Koordinaten zu verändern. Die Eigenschaften, die angegeben werden müssen, sind zwei Ebenen in der Objekthierarchie tiefer angeordnet, aber die Syntax des `\override`-Befehls ist ziemlich einfach:

```
e2 \glissando b
\once \override Glissando #'(bound-details left Y) = #3
\once \override Glissando #'(bound-details right Y) = #-2
e2 \glissando b
```

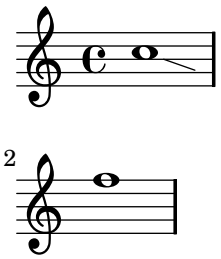


Die Einheiten für die `Y`-Eigenschaft werden in Notenlinienabständen angegeben, wobei die Mittellinie des Notensystems die Null darstellt. Für das Glissando ist der Wert von `Y` am entsprechenden X-Koordinatenpunkt entsprechend dem Mittelpunkt des Notenkopfes, wenn die Linie bis in die Noten hinein weitergeführt werden würde.

Wenn `Y` nicht gesetzt wird, wird der Wert aus der vertikalen Position des entsprechenden Anknüpfpunkts des Streckers errechnet.

Im Fall eines Zeilenumbruchs werden die Werte der Endpunkte in den Unterlisten `left-broken` bzw. `right-broken` von `bound-details` abgelegt. Zum Beispiel:

```
\override Glissando #'breakable = ##t
\override Glissando #'(bound-details right-broken Y) = #-3
c1 \glissando \break
f1
```



Eine Anzahl weitere Eigenschaft der `left-` und `right`-Unterlisten der `bound-details`-Eigenschaft kann auf gleiche Weise wie `Y` verändert werden:

`Y` Hiermit wird der Y-Koordinationspunkt des Endpunktes in Notenlinienabständen vom Mittelpunkt des Notensystems ausgehend angegeben. Der Endpunkt ist normalerweise der Mittelpunkt des Elternobjektes, sodass Glissandos vertikal auf den Mittelpunkt eines Notenkopfes weist.

Für horizontale Strecker, wie Textstrecker und Trillerstrecker ist sein Wert mit 0 definiert.
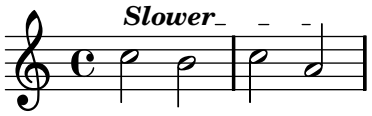
`attach-dir`
Das entscheidet, wo die Linie auf der X-Achse beginnt und endet, relativ zum Elternobjekt. Ein Wert `-1` (oder `LEFT`) lässt die Linie an der linken Seite der Noten beginnen/enden, mit der sie verknüpft ist.

`X` Das ist der absolute X-Koordinatenpunkt des Endpunktes. Der Wert wird normalerweise in Echtzeit errechnet, und ihn zu verändern ist normalerweise nicht nützlich.

`stencil` Linienstrecker können Symbole am Ende oder zu Anfang des Streckers haben, die in dieser Untereigenschaft definiert werden. Die Eigenschaft ist für interne Benutzung, es wird empfohlen, die Eigenschaft `text` zu benutzen.

`text` Das ist eine Textbeschriftung, die ausgewertet wird und die `stencil`-Eigenschaft überschreibt. Sie wird eingesetzt, um *cresc.*, *tr* oder andere Texte an horizontale Strecker zu setzen.

```
\override TextSpanner #'(bound-details left text)
   = \markup { \small \bold Slower }
c2\startTextSpan b c a\stopTextSpan
```
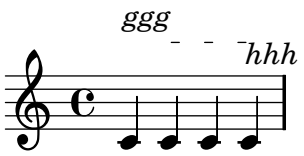


`stencil-align-dir-y`
`stencil-offset`

Wenn keine dieser beiden Eigenschaften gesetzt wird, wird die Matrize (engl. stencil) einfach am Endpunkt des Streckers, auf seiner Mittellinie (wie durch `X` und `Y` definiert) zentriert, ausgegeben. Wenn entweder `stencil-align-dir-y` oder `stencil-offset` gesetzt werden, wird das Symbol am Rand vertikal entsprechend des Endpunktes der Linie verschoben:

```
\override TextSpanner
  #'(bound-details left stencil-align-dir-y) = #-2
\override TextSpanner
  #'(bound-details right stencil-align-dir-y) = #UP

\override TextSpanner
  #'(bound-details left text) = #"ggg"
\override TextSpanner
  #'(bound-details right text) = #"hhh"
c4^\startTextSpan c c c \stopTextSpan
```
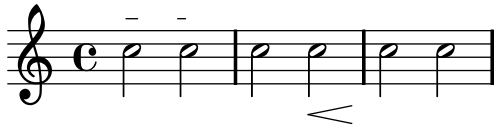


Dabei sollte beachtet werden, dass negative Werte das Objekt nach *oben* verschieben, anders als man erwarten könnte, weil der Wert `-1` oder `DOWN` bedeutet, dass die *Unterkante* des Textes mit der Streckerlinie ausgerichtet wird. Ein Wert `1` oder `UP` richtet die Oberkante des Textes mit der Streckerlinie aus.

`arrow` Wenn diese Untereigenschaft auf `#t` gesetzt wird, wird ein Pfeilkopf am Ende der Linie erstellt.

`padding` Diese Eigenschaft kontrolliert den Abstand zwischen dem angegebenen Endpunkt der Linie und dem wirklichen Ende. Ohne Füllung (engl. padding) würde ein Glissando in der Mitte eines Notenkopfes beginnen und enden.

Die musikalische Funktion `\endSpanners` beschließt den Strecker, der an der direkt folgenden Note beginnt, bevor er eigentlich zu ende wäre. Er wird exakt nach einer Note beendet, oder am nächsten Taktstrich, wenn `to-barline` auf wahr gesetzt ist und eine Taktlinie vor der nächsten Note erscheint.

```
\endSpanners
c2 \startTextSpan c2 c2
\endSpanners
c2 \< c2 c2
```



Wenn man `\endSpanners` benutzt, ist es nicht nötig, den Befehl \startTextSpan mit \stopTextSpan zu beenden, und es ist auch nicht nötig, Crescendo-Klammern mit `\!` zu beenden.

Siehe auch

Referenz der Interna: Abschnitt "TextSpanner" in *Referenz der Interna*, Abschnitt "Glissando" in *Referenz der Interna*, Abschnitt "VoiceFollower" in *Referenz der Interna*, Abschnitt "TrillSpanner" in *Referenz der Interna*, Abschnitt "line-spanner-interface" in *Referenz der Interna*.

5.4.7 Sichtbarkeit von Objekten

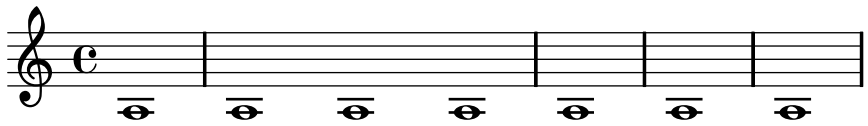
Die Sichtbarkeit von Layout-Objekten kann auf vier Arten kontrolliert werden: Ihre Matrizen (engl stencil) können entfernt werden, sie können unsichtbar gemacht werden, sie können weiß eingefärbt werden und ihre `break-visibility`-Eigenschaft kann verändert werden. Die ersten drei Möglichkeiten beziehen sich auf alle Layout-Objekte, die letzte nur auf einige wenige, nämlich die *zerteilbaren* Objekte. Das Handbuch zum Lernen führt in alle vier Möglichkeiten ein, siehe Abschnitt "Sichtbarkeit und Farbe von Objekten" in *Handbuch zum Lernen*.

Es gibt auch einige weitere Techniken, die sich nur auf bestimmte Layout-Objekte beziehen. Sie werden im letzten Abschnitt behandelt.

Einen `stencil` entfernen

Jedes Layout-Objekt hat eine Matrizen-(stencil)-Eigenschaft. Sie ist normalerweise definiert als die Funktion, die das entsprechende Objekt zeichnet. Wenn die Eigenschaft mit `\override` auf `#f` gesetzt wird, wird keine Funktion aufgerufen und also auch kein Objekt gezeichnet. Das Standardverhalten kann mit dem Befehl `\revert` wieder hergestellt werden.

```
a1 a
\override Score.BarLine #'stencil = ##f
a a
\revert Score.BarLine #'stencil
a a a
```



Objekten unsichtbar machen

Jedes Layout-Objekt hat eine Durchsichtigkeits-Eigenschaft (`'transparent`), die normalerweise auf den Wert `#f` gesetzt ist. Wenn sie auf `#t` gesetzt wird, nimmt das Objekt immer noch den entsprechenden Platz ein, ist aber unsichtbar.

```
a4 a
\once \override NoteHead #'transparent = ##t
a a
```



Objekte weiß malen

Alle Layout-Objekte haben eine Farb-(color)-Eigenschaft, die normalerweise schwarz (`black`) definiert ist. Wenn sie nach weiß (`white`) verändert wird, kann man das Objekt nicht mehr vom weißen Hintergrund unterscheiden. Wenn das Objekt jedoch andere Objekte überschneidet, wird die Farbe der Überschneidungen von der Reihenfolge entschieden, in welcher die Objekte gesetzt werden. Es kann also vorkommen, dass man die Umrisse des weißen Objektes erahnen kann, wie in diesem Beispiel:

```
\override Staff.Clef #'color = #white
a1
```



Das kann man vermeiden, indem man die Satzreihenfolge der Objekte verändert. Alle Layout-Objekte haben eine `layer`-Eigenschaft, die auf eine ganze Zahl gesetzt sein muss. Objekte mit der niedrigsten Zahl in der `layer`-Eigenschaft werden zuerst gesetzt, dann die nächsten Objekte in ansteigender Ordnung. Objekte mit höheren Werten überschneiden also Objekte mit niedrigeren Werten. Die meisten Objekte bekommen den Wert `1` zugewiesen, einige wenige Objekte, unter die auch `StaffSymbol` (die Notenlinien) gehört, jedoch den Wert `0`. Die Reihenfolge, in der Objekte mit demselben Wert gesetzt werden, ist nicht definiert.

Im oberen Beispiel wird der weiße Schlüssel, der einen Wert von `1` für `layer` hat, nach den Notenlinien gesetzt (die einen Wert von `0` für `layer` haben) und überschneidet sie also. Um das zu ändern, muss dem `Clef`-Objekt (Notenschlüssel) ein niedrigerer Wert, etwa `-1`, gegeben werden, sodass es früher gesetzt wird:

```
\override Staff.Clef #'color = #white
\override Staff.Clef #'layer = #-1
a1
```



`break-visibility` (unsichtbar machen) benutzen

Die meisten Layout-Objekte werden nur einmal gesetzt, aber einige, wie Taktstriche, Schlüssel, Taktartbezeichnung und Tonartvorzeichen, müssen mehrmals gesetzt werden, wenn die Zeile gewechselt wird: einmal am Ende des oberen Systems und ein zweites Mal zu Beginn des nächsten Systems. Derartige Objekte werden als *trennbar* bezeichnet und haben eine Eigenschaft, die `break-visibility`-Eigenschaft, mit der ihre Sichtbarkeit an allen drei Positionen, an denen sie auftreten können, kontrolliert werden kann: zu Beginn einer Zeile, innerhalb einer Zeile, wenn sie verändert werden, und am Ende einer Zeile, wenn die Änderung hier stattfindet.

Die Taktart wird beispielsweise standardmäßig nur zu Beginn des ersten Systems gesetzt, aber an anderen Stellen nur, wenn sie sich ändert. Wenn diese Änderung am Ende eines Systems auftritt, wird die neue Taktart am Ende des aktuellen Systems als auch zu Beginn des nächsten Systems gesetzt.

Dieses Verhalten wird von der `break-visibility`-Eigenschaft kontrolliert, die erklärt wird in Abschnitt “Sichtbarkeit und Farbe von Objekten” in *Handbuch zum Lernen*. Die Eigenschaft braucht einen Vektor von drei Booleschen Werten, die in ihrer Reihenfolge bestimmte, ob das Objekt a) zu Ende der Zeile, b) innerhalb einer Zeile oder c) zu Beginn einer Zeile gesetzt wird. Oder, genauer gesagt, vor einem Zeilenumbruch, an Stellen, wo kein Zeilenumbruch auftritt oder nach einem Zeilenumbruch.

Die acht möglichen Kombinationen können auch durch vordefinierte Funktionen bestimmt werden, welche in der Datei '`scm/output-lib.scm`' definiert sind. Die letzten drei Spalten der folgenden Tabelle zeigen an, ob das Layout-Objekt an einer bestimmten Position sichtbar sein wird oder nicht:
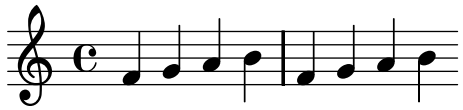
Funktion Form	**Vektor Form**	**Vor Umbruch**	**kein Umbruch**	**Nach Umbruch**
`all-visible`	`'#(#t #t #t)`	ja	ja	ja
`begin-of-line-visible`	`'#(#f #f #t)`	nein	nein	ja
`center-visible`	`'#(#f #t #f)`	nein	ja	nein
`end-of-line-visible`	`'#(#t #f #f)`	ja	nein	nein
`begin-of-line-invisible`	`'#(#t #t #f)`	ja	ja	nein
`center-invisible`	`'#(#t #f #t)`	ja	nein	ja
`end-of-line-invisible`	`'#(#f #t #t)`	nein	ja	ja
`all-invisible`	`'#(#f #f #f)`	nein	nein	nein
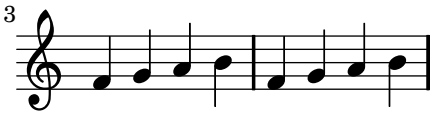
Die Standardeinstellungen von `break-visibility` hängen vom Layout-Objekt ab. Die folgende Tabelle zeigt alle wichtigen Layout-Objekte, die mit `break-visibility` verändert werden können und die jeweiligen Standardeinstellungen der Eigenschaft:

Layout-Objekt	**Normaler Kontext**	**Standardeinstellung**
`BarLine` (Taktstrich)	`Score`	calculated
`BarNumber` (Taktzahl)	`Score`	`begin-of-line-visible`
`BreathingSign` (Atemzeichen)	`Voice`	`begin-of-line-invisible`
`Clef` (Schlüssel)	`Staff`	`begin-of-line-visible`
`Custos`	`Staff`	`end-of-line-visible`
`DoublePercentRepeat` (Doppel-Prozent-Wiederholung	`Voice`	`begin-of-line-invisible`
`KeySignature` (Tonart)	`Staff`	`begin-of-line-visible`
`OctavateEight` (Oktavierungs-Acht)	`Staff`	`begin-of-line-visible`
`RehearsalMark` (Übungszeichen)	`Score`	`end-of-line-invisible`
`TimeSignature` (Taktart)	`Staff`	`all-visible`

Das Beispiel unten zeigt die Verwendung der Vektor-Form um die Sichtbarkeit von Taktlinien zu bestimmten:

```
f4 g a b
f4 g a b
% Remove bar line at the end of the current line
\once \override Score.BarLine #'break-visibility = #'#(#f #t #t)
\break
f4 g a b
f4 g a b
```





Obwohl alle drei Bestandteile des Vektors, mit denen `break-visibility` definiert wird, vorhanden sein müssen, haben nicht alle eine Auswirkung auf jedes Layout-Objekt, und einige Kombinationen können sogar Fehler hervorrufen. Es gelten die folgenden Einschränkungen:
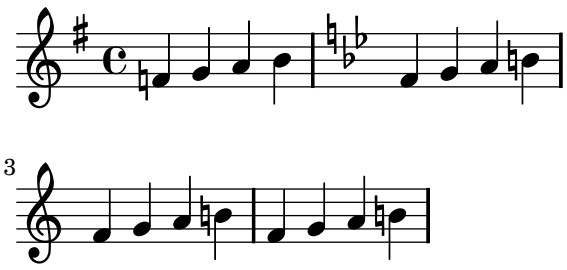
- Taktstriche können nicht zu Beginn einer Zeile gesetzt werden.
- Eine Taktzahl kann nicht zu Beginn der ersten Zeile gesetzt werden, außer wenn er nicht 1 ist.
- Schlüssel – siehe unten.
- Doppel-Prozent-Wiederholungen werden entweder alle ausgegeben oder alle unterdrückt. Mit `begin-of-line-invisible` werden sie ausgegeben, mit `all-invisible` unterdrückt.
- Tonart – siehe unten.
- Oktavierungs-Acht – siehe unten.

Besonderheiten

Sichtbarkeit nach expliziten Änderungen
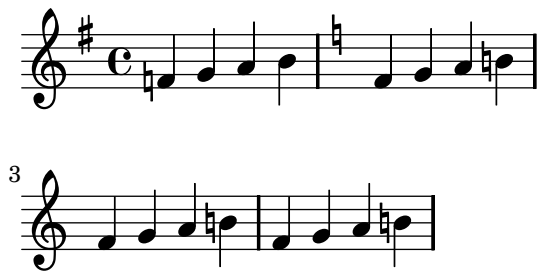
Die `break-visibility`-Eigenschaft kontrolliert die Sichtbarkeit von Tonarten und Schlüsseländerungen nur zu Beginn einer Zeile, d.h. nach einem Zeilenumbruch. Sie hat keinen Einfluss auf die Sichtbarkeit von Tonarten bzw. Schlüsseln, die nach einer expliziten Tonart- oder Schlüsseländerung in oder am Ende einer Zeile angezeigt werden. Im nächsten Beispiel ist die Tonartangabe nach dem expliziten Wechsel zu B-Dur immer noch sichtbar, obwohl `all-invisible` eingesetzt wurde:

```
\key g \major
f4 g a b
% Try to remove all key signatures
\override Staff.KeySignature #'break-visibility = #all-invisible
\key bes \major
f4 g a b
\break
f4 g a b
f4 g a b
```



Die Sichtbarkeit derartiger expliziter Tonart- und Schlüsseländerungen wird von den `explicitKeySignatureVisibility`- und `explicitClefVisibility`-Eigenschaften kontrolliert. Sie entsprechen der `break-visibility`-Eigenschaft und beide brauchen drei Boolesche Werte bzw. die oben aufgelisteten vordefinierten Funktionen als Argument, genau wie `break-visibility`. Beide sind Eigenschaft des `Staff`-Kontextes, nicht der Layout-Objekte selber, weshalb sie mit dem Befehl `\set` eingesetzt werden. Beide sind standardmäßig auf die Funktion `all-visible` gesetzt. Diese Eigenschaften kontrollieren nur die Sichtbarkeit von Tonarten bzw. Schlüssel, die von expliziten Änderungen herrühren, und haben keinen Einfluss auf Tonarten und Schlüssel zu Beginn einer Zeile – um diese zu beeinflussen, muss `break-visibility` benutzt werden.
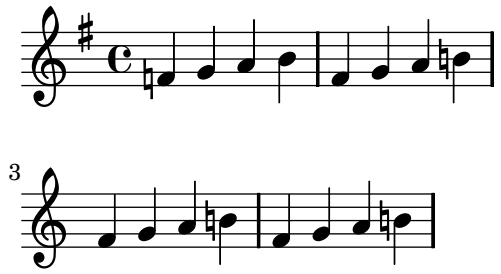
```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Staff.KeySignature #'break-visibility = #all-invisible
\key bes \major
f4 g a b \break
f4 g a b
f4 g a b
```



Sichtbarkeit von erinnernden Versetzungszeichen

Um erinnernde Versetzungszeichen zu entfernen, die nach einer expliziten Tonartänderung auftreten, muss die `Staff`-Eigenschaft `printKeyCancellation` auf `#f` gesetzt werden:

```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.printKeyCancellation = ##f
\override Staff.KeySignature #'break-visibility = #all-invisible
\key bes \major
f4 g a b \break
f4 g a b
f4 g a b
```



Mit diesen Veränderungen bleiben nur noch die Versetzungszeichen vor den Noten übrig um den Wechsel der Tonart anzuzeigen.

Automatische Takte

Ein Sonderfall sind die automatischen Taktstriche, die mit der Eigenschaft `automaticBars` im `Score`-Kontext ausgeschaltet werden können. Wenn sie auf `#f` gesetzt ist, werden Taktstrich nicht automatisch ausgegeben sondern müssen explizit mit dem `\bar`-Befehl eingegeben werden. Anders als bei dem `\cadenzaOn`-Befehl werden die Takte allerdings immer noch gezählt. Takterstellung wird später wieder mit diesem Zahl aufgenommen, wenn die Eigenschaft wieder auf `#t` gesetzt wird. Wenn sie den Wert `#f` hat, können Zeilenumbrüche nur an expliziten `\bar`-Befehlen auftreten.

Oktavierte Schlüssel

Das kleine Oktavierungssymbol von oktavierten Notenschlüsseln wird durch das `OctavateEight`-Layout-Objekt erstellt. Seine Sichtbarkeit wird automatisch vom `Clef`-Objekt geerbt, sodass Veränderungen von `break-visibility` des `OctavateEight`-Layout-Objekts nicht auch noch für unsichtbare Schlüssel zusätzlich vorgenommen werden müssen.

Bei expliziten Schlüsseländerungn kontrolliert die `explicitClefVisibility`-Eigenschaft wohl das Schlüsselsymbol als auch das damit verknüpfte Oktavierungssymbol.

Siehe auch

Handbuch zum Lernen: Abschnitt “Sichtbarkeit und Farbe von Objekten” in *Handbuch zum Lernen*
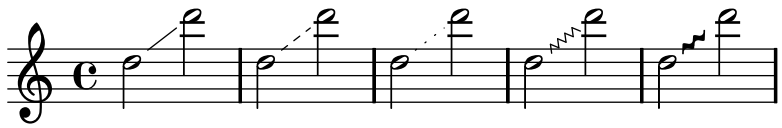
5.4.8 Linienstile

Einige Aufführungsanweisungen (z. B. *rallentando* und *accelerando* oder Triller werden als Text gesetzt und möglicherweise über mehrere Takte mit Linien fortgeführt, die teilweise gestrichelt oder gewellt sind.

Alle benutzen die gleichen Routinen wie das Glissando, um Text und Linien zu produzieren, weshalb auch eine Veränderungen der Erscheinung auf gleiche Weise vonstatten geht. Die Ausgabe erfolgt durch einen Strecker (engl. spanner), und die Routine, die ihn setzt, heißt `ly:line-interface::print`. Diese Routine bestimmt die exakte Position von zwei *Strecker-Punkten* und zeichnet eine Linie zwischen sie im gewünschten Linienstil.
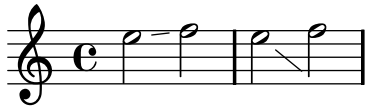
Hier einige Beispiele, welche Linienstile möglich sind und wie sie verändert werden können:

```
d2 \glissando d'2
\once \override Glissando #'style = #'dashed-line
d,2 \glissando d'2
\override Glissando #'style = #'dotted-line
d,2 \glissando d'2
\override Glissando #'style = #'zigzag
d,2 \glissando d'2
\override Glissando #'style = #'trill
d,2 \glissando d'2
```



Die Position der Endpunkte des Streckers werden in Realzeit für jedes graphische Objekt errechnet, aber es ist möglich, sie manuell vorzugeben:

```
e2 \glissando f
\once \override Glissando #'(bound-details right Y) = #-2
e2 \glissando f
```



Der Wert von `Y` wird für den rechten Endpunkt auf `-2` gesetzt. Die linke Seite kann ähnlich angepasst werden, indem man `left` anstelle von `right` angibt.

Wenn `Y` nicht gesetzt ist, wird der Wert ausgehend von der vertikalen Position der linken und rechten Anbindepunkte des Streckers errechnet.

Andere Anpassungen der Strecker sind auch möglich, für Einzelheiten siehe Abschnitt 5.4.6 [Strecker], Seite 541.

5.4.9 Drehen von Objekten

Layout-Objekte und Textbeschriftungselemente können zu einem beliebigen Winkel um einen beliebigen Punkt herum gedreht werden, aber die Methode, mit der die Änderung vorgenommen werden muss, unterscheidet sich je nach Objekt.
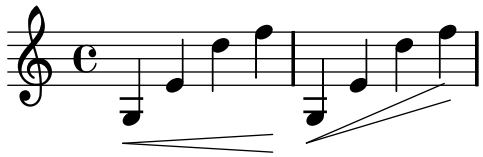
Drehen von Objekten

Alle Layout-Objekte, die das `grob-interface` unterstützen, können gedreht werden, indem man ihre `rotation`-Eigenschaft einstellt. Sie erhält eine Liste mit drei Einträgen: den Winkel der Drehung gegen den Uhrzeiger sowie die X- und Y-Koordinaten des Punktes relativ zum Referenzpunkt des Objekts, um welchen herum die Drehung stattfinden soll. Der Winkel der Drehung wird in Grad angegeben, die Koordinaten in Notenlinienzwischenräumen.

Der Winkel der Drehung und die Koordinaten des Drehpunktes müssen durch Ausprobieren herausgefunden werden.

Es gibt nur wenige Situationen, in welchen die Drehung eines Layout-Objektes sinnvoll ist. Das folgende Beispiel zeigt eine sinnvolle Anwendung:
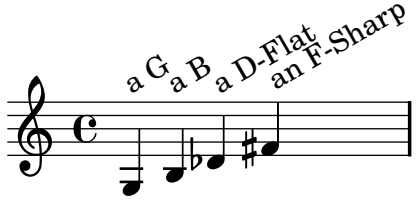
```
g4\< e' d' f\!
\override Hairpin #'rotation = #'(20 -1 0)
g,,4\< e' d' f\!
```



Textbeschriftung drehen

Jede Textbeschriftung kann gedreht werden, indem vor die Anweisung der Befehl `\rotate` gesetzt wird. Der Befehl hat zwei Argumente: Den Winkel der Drehung in Grad gegen den Uhrzeiger und der Text, der gedreht dargestllt werden soll. Die Ausdehnung des Textes wird nicht gedreht, sie erhält ihren Wert von den Extrempunkten der x- und y-Koordinaten des gedrehten Textes. Im folgenden Beispiel wird die `outside-staff-priority`-Eigenschaft auf `#f` gesetzt, damit automatische Zusammenstöße nicht verhindert werden, wodurch andernfalls einige der Texte zu hoch geschoben werden würden.

```
\override TextScript #'outside-staff-priority = ##f
g4^\markup { \rotate #30 "a G" }
b^\markup { \rotate #30 "a B" }
des^\markup { \rotate #30 "a D-Flat" }
fis^\markup { \rotate #30 "an F-Sharp" }
```



5.5 Fortgeschrittene Optimierungen

Dieser Abschnitt behandelt verschiedene Möglichkeiten, das Aussehen des Notenbildes zu polieren.

Siehe auch

Handbuch zum Lernen: Abschnitt “Die Ausgabe verändern” in *Handbuch zum Lernen*, Abschnitt “Mehr Information” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 5.2 [Die Referenz der Programminterna erklärt], Seite 525, Abschnitt 5.3 [Eigenschaften verändern], Seite 528, ⟨undefined⟩ [Schnittstellen für Programmierer], Seite ⟨undefined⟩.

Installierte Dateien: ‘`scm/define-grobs.scm`’.

Schnipsel: Abschnitt “Tweaks and overrides” in *Schnipsel*.

Referenz der Interna: Abschnitt “All layout objects” in *Referenz der Interna*.

5.5.1 Objekte ausrichten

Graphische Objekte, die das `self-alignment-interface` und/obder das `side-position-interface` unterstützen, können an einem vorher gesetzten Objekt auf verschiedene Weise ausgerichtet werden. Eine Liste derartiger Objekte findet sich in Abschnitt “self-alignment-interface” in *Referenz der Interna* und Abschnitt “side-position-interface” in *Referenz der Interna*.

Alle graphischen Objekte haben einen Referenzpunkt, eine horizontale Ausdehnung und eine vertikale Ausdehnung. Die horizontale Ausdehnung ist ein Zahlenpaar, mit dem die Verschiebung der rechten und linken Ecken ausgehend vom Referenzpunkt angegeben werden, wobei Verschiebungen nach links mit negativen Zahlen notiert werden. Die vertikale Ausdehnung ist ein Zahlenpaar, das die Verschiebung der unteren und oberen Ränder vom Referenzpunkt ausgehend angibt, wobei Verschiebungen nach unten mit negativen Zahlen notiert werden.

Die Position eines Objektes auf dem Notensystem wird mit Werten von `X-offset` und `Y-offset` angegeben. Der Wert von `X-offset` gibt die Verschiebung von der X-Koordinate des Referenzpunkts des Elternobjektes an, der Wert von `Y-offset` die Verschiebung ausgehend von der Mittellinie des Notensystemes. Die Werte von `X-offset` und `Y-offset` können direkt bestimmt werden oder durch Prozeduren errechnet werden, sodass eine Ausrichtung mit dem Elternobjekt erreicht werden kann.

Achtung: Viele Objekte brauchen besondere Überlegungen zu ihrer Position, weshalb in manchen Fällen manuell gesetzte Werte von `X-offset` oder `Y-offset` ignoriert oder verändert werden können, obwohl das Objekt das `self-alignment-interface` unterstützt. Wenn man `X-offset` oder `Y-offset` auf einen festen Wert setzt, wird die entsprechende `self-alignment`-Eigenschaft ignoriert.

Ein Versetzungszeichen beispielsweise kann vertikal durch Veränderung von `Y-offset` verschoben werden, aber Änderungen von `X-offset` haben keine Auswirkung.

Übungszeichen können an trennbaren Objekten (wie Taktstrichen, Schlüsseln, Taktarten und Tonartvorzeichen) ausgerichtet werden. In `break-aligned-interface` finden sich besondere Eigenschaften, mit denen Übungszeichen an derartigen Objekten ausgerichtet werden können.

Siehe auch

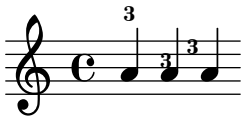
Notationshandbuch: [Benutzung des break-alignable-interface], Seite 556.

Erweitern: Abschnitt “Callback functions” in *Extending*.

`X-offset` und `Y-offset` direkt setzen

Numerische Werte können den `X-offset`- und `Y-offset`-Eigenschaften vieler Objekte zugewiesen werden. Das folgende Beispiel zeigt drei Noten mit der Standardposition von Fingersatzanweisungen und die Positionen, wenn `X-offset` und `Y-offset` verändert werden.

```
a-3
a
-\tweak #'X-offset #0
-\tweak #'Y-offset #0
-3
a
-\tweak #'X-offset #-1
-\tweak #'Y-offset #1
-3
```



Das `side-position-interface` benutzen

Ein Objekt, das die `side-position-interface`-Schnittstelle unterstützt, kann neben sein Elternobjekt gesetzt werden, sodass zwei definierte Enden der Objekte sich berühren. Das Objekt kann über, unter, rechts oder links vom Ursprungsobjekt positioniert werden. Das Ursprungsobjekt kann nicht definiert werden: es ergibt sich aus der Reihenfolge der Objekte in der Eingabe. Die meisten Objekte haben einen Notenkopf als Ursprung assoziiert.

Die Werte von `side-axis` und `direction` bestimmen, wo das Objekt platziert werden soll, wie in der Tabelle zu sehen:

`side-axis`-Eigenschaft	`direction`-Eigenschaft	Platzierung
`0`	`-1`	links
`0`	`1`	rechts
`1`	`-1`	unten
`1`	`1`	oben

Wenn `side-axis` gleich `0` ist, sollte `X-offset` auf die Prozedur `ly:side-position-interface::x-aligned-side` gesetzt werden. Diese Prozedur errechnet den richtigen Wert für `X-offset`, sodass das Objekt auf der rechten oder linken Seite des Ursprungs angeordnet wird, entsprechend dem Wert der `direction`-Eigenschaft.

Wenn `side-axis` gleich `1` ist, sollte `Y-offset` auf die Prozedur `ly:side-position-interface::y-aligned-side` gesetzt werden. Diese Prozedur errechnet den richtigen Wert für `Y-offset`, sodass das Objekt über oder unter dem Ursprungsobjekt angeordnet wird, entsprechend dem Wert der `direction`-Eigenschaft.

Das `self-alignment-interface` benutzen

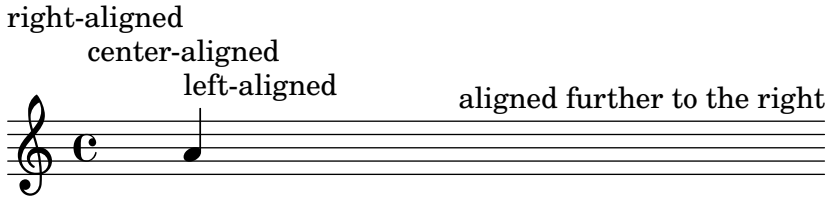
Selbstausrichtende Objekte horizontal

Die horizontale Ausrichtung eines Objektes, das die `self-alignment-interface`-(Selbstausrichtungs)-Schnittstelle unterstützt, wird durch den Wert von `self-alignment-X` kontrolliert, vorausgesetzt die Eigenschaft `X-offset` des Objektes ist auf `ly:self-alignment-interface::x-aligned-on-self` gesetzt. `self-alignment-X` kann eine beliebige reale Zahl zugewiesen werden, in Einheiten der Hälfte der X-Gesamtausdehnung des Objekts. Negative Werte verschieben das Objekt nach rechts, positive nach links. Ein Wert von `0` zentriert das Objekt auf dem Referenzpunkt des Ursprungs, ein Wert von `-1` richtet die linke Ecke des Objekts am Referenzpunkt des Ursprungsobjektes aus,

ein Wert von `1` richtet die rechte Ecke des Objektes am Referenzpunkt des Ursprungsobjektes aus. Die Symbole `LEFT`, `CENTER` und `RIGHT` können anstelle von `-1, 0, 1` eingesetzt werden.

Normalerweise würde der `\override`-Befehl benutzt werden, um die Werte von `self-alignment-X` zu verändern, aber der `\tweak`-Befehl kann benutzen, um verschiedene Anmerkungen an einer einzigen Note auszurichten:

```
a'
-\tweak #'self-alignment-X #-1
^"left-aligned"
-\tweak #'self-alignment-X #0
^"center-aligned"
-\tweak #'self-alignment-X #RIGHT
^"right-aligned"
-\tweak #'self-alignment-X #-2.5
^"aligned further to the right"
```



Objekte vertikal automatisch ausrichten
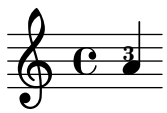
Objekte können auf ähnliche Weise auch vertikal aneinander ausgerichtet werden, wenn ihre `Y-offset`-Eigenschaft auf `ly:self-alignment-interface::y-aligned-on-self` gesetzt ist. Oft greifen jedoch auch andere Mechanismen bei der vertikalen Ausrichtung ein: Der Wert von `Y-offset` ist nur eine der Variablen, die für die Berechnung benutzt werden. Darum ist es kompliziert, den Wert für einige Objekte richtig anzupassen. Die Einheiten sind Halbe der vertikalen Ausdehnung des Objektes, welche normalerweise recht klein ist, sodass ziemlich große Werte erforderlich sein können. Der Wert `-1` richtet die untere Kante des Objekts am Referenzpunkt des Ursprungsobjektes aus, der Wert `0` richtet die Mitte des Objekts am Referenzpunkt des Ursprungsobjektes aus und der Wert `1` richtet die Oberkante des Objektes am Referenzpunkt des Ursprungsobjektes aus. Die Symbole `DOWN`, `CENTER` und `UP` können anstelle von `-1, 0, 1` benutzt werden.

Automatische Ausrichtung in beide Richtungen

Indem sowohl `X-offset` als auch `Y-offset` eingestellt werden, kann ein Objekt gleichzeitig in beiden Richtungen ausgerichtet werden.

Das folgende Beispiel zeigt, wie man eine Fingersatzanweisung so ausrichtet, dass sie nah am Notenkopf bleibt.

```
a
-\tweak #'self-alignment-X #0.5  % move horizontally left
-\tweak #'Y-offset #ly:self-alignment-interface::y-aligned-on-self
-\tweak #'self-alignment-Y #-1  % move vertically up
-3  % third finger
```

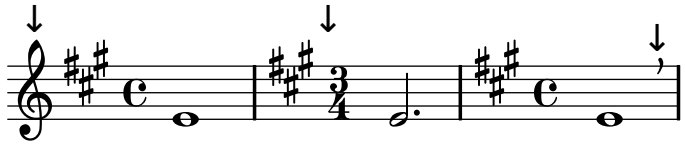


Benutzung des `break-alignable-interface`

Übungszeichen und Taktzahlen können an Notationsobjekten (ausschließlich Taktstriche) ausgerichtet werden. Zu diesen Objekten gehören `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature` und `time-signature`.

Standardmäßig werden Übungszeichen und Taktzahlen horizontal über dem Objekt zentriert:

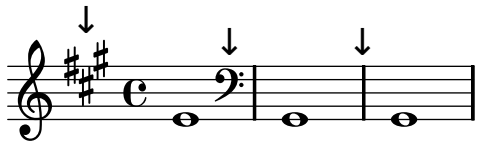
```
% The RehearsalMark will be centered above the Clef
\override Score.RehearsalMark #'break-align-symbols = #'(clef)
\key a \major
\clef treble
\mark ""
e1
% The RehearsalMark will be centered above the TimeSignature
\override Score.RehearsalMark #'break-align-symbols = #'(time-signature)
\key a \major
\clef treble
\time 3/4
\mark ""
e2.
% The rehearsal mark will be centered above the Breath Mark
\override Score.RehearsalMark #'break-align-symbols = #'(breathing-sign)
\key a \major
\clef treble
\time 4/4
e1
\breathe
\mark ""
```



Eine Liste von möglichen Objekten zur Ausrichtung kann definiert werden. Wenn eins dieser Objekte an der aktuellen Stelle unsichtbar ist (etwa durch Einstellung von `break-visibility` oder die expliziten Sichtbarkeitseinstellungen von Taktart und Vorzeichen), werden Übungszeichen und Taktzahlen an dem ersten Objekt in der Liste ausgerichtet, dass sichtbar ist. Wenn keine Objekte in der Liste sichtbar sind, wird das Objekt am Taktstrich ausgerichtet. Wenn der Taktstrich unsichtbar ist, wird das Objekt an der Stelle ausgerichtet, an der sich der Taktstrich befinden würde.

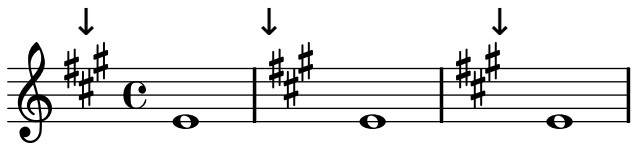
```
% The RehearsalMark will be centered above the Key Signature
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark ""
e1
% The RehearsalMark will be centered above the Clef
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature clef)
\key a \major
\clef bass
\mark ""
```

```
gis,,1
% The rehearsal mark will be centered above the Bar Line
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark ""
e''1
```



Die Ausrichtung des Übungszeichen relativ zum Notationsobjekt kann verändert werden, wie das nächste Beispiel zeigt. In einer Partitur mit vielen Systemen würde man diese Einstellung für alle Systeme vornehmen.

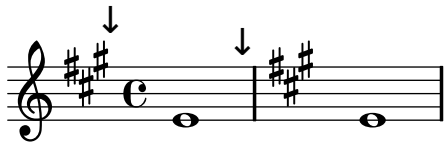
```
% The RehearsalMark will be centered above the KeySignature
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
\key a \major
\clef treble
\time 4/4
\mark ""
e1
% The RehearsalMark will be aligned with the left edge of the KeySignature
\once \override Score.KeySignature #'break-align-anchor-alignment = #LEFT
\mark ""
\key a \major
e1
% The RehearsalMark will be aligned with the right edge of the KeySignature
\once \override Score.KeySignature #'break-align-anchor-alignment = #RIGHT
\key a \major
\mark ""
e1
```



Das Übungszeichen kann auch nach rechts oder links um einen beliebigen Wert verschoben werden. Die Einheiten sind in Notenlinienzwischenräumen:

```
% The RehearsalMark will be aligned with the left edge of the KeySignature
% and then shifted right by 3.5 staff-spaces
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
\once \override Score.KeySignature #'break-align-anchor = #3.5
\key a \major
\mark ""
e1
% The RehearsalMark will be aligned with the left edge of the KeySignature
```

```
% and then shifted left by 2 staff-spaces
\once \override Score.KeySignature #'break-align-anchor = #-2
\key a \major
\mark ""
e1
```



5.5.2 Vertikale Gruppierung der grafischen Objekte („grob“s)

Die graphischen Objekte `VerticalAlignment` und `VerticalAxisGroup` funktionieren zusammen. `VerticalAxisGroup` gruppiert unterschiedliche Objekte wie Notensysteme, Gesangstext usw. zusammen. `VerticalAlignment` richtet die unterschiedlichen Objektgruppen dann aneinander aus. Es gibt normalerweise nur ein `VerticalAlignment` in einer Partitur, aber jedes Notensystem, Gesangstext usw. hat eine eigene `VerticalAxisGroup`.

5.5.3 stencils verändern
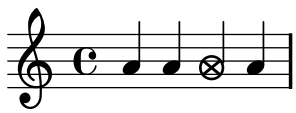
Alle Layout-Objekte haben eine `stencil`-(Stempel-)Eigenschaft, die ein Teil von `grob-interface` ist. Diese Eigenschaft ist normalerweise als eine Funktion definiert, die auf das jeweilige Objekt angepasst ist und das Symbol erstellt, dass dann im Druckbild erscheint. Beispielsweise die Standardeinstellung für die `stencil`-Eigenschaft von `MultiMeasureRest` (Ganztaktpausenobjekt) ist `ly:multi-measure-rest::print`.

Das Standardsymbol für jedes Objekt kann ersetzt werden, indem man die `stencil`-Eigenschaft verändert, sodass sie auf eine andere, speziell geschriebene Prozedur verweist. Das erfordert einen hohen Grad an Kenntnis der LilyPond-Interna, aber es gibt einen einfacheren Weg, mit dem man oft vergleichbarere Ergebnisse erzielen kann.

Dieser Weg besteht darin, die `stencil`-Eigenschaft auf die Prozedur zu verweisen, die Text ausgibt: `ly:text-interface::print` und eine `text`-Eigenschaft zu dem Objekt hinzuzufügen, in welcher dann die Textbeschriftung definiert wird, mit der das entsprechende Symbol dargestellt wird. Aufgrund der Flexibilität der Textbeschriftung ist hier sehr viel möglich. Siehe zu Details insbesondere [Graphische Notation innerhalb einer Textbeschriftung], Seite 217.

Das folgende Beispiel zeigt diese Methode, indem das Symbol der Notenköpfe in ein Kreuz innerhalb eines Kreises umgewandelt wird.

```
XinO = {
  \once \override NoteHead  #'stencil = #ly:text-interface::print
  \once \override NoteHead #'text = \markup {
    \combine
      \halign #-0.7 \draw-circle #0.85 #0.2 ##f
      \musicglyph #"noteheads.s2cross"
  }
}
\relative c'' {
  a a \XinO a a
}
```



Alle Schriftzeichen in der feta-Schriftart können mit dem `\musicglyph`-Befehl erreicht werden. Siehe auch Abschnitt A.7 [Die Feta-Schriftart], Seite 578.

Siehe auch

Notationsreferenz: [Graphische Notation innerhalb einer Textbeschriftung], Seite 217, Abschnitt 1.8.2 [Text formatieren], Seite 210, ⟨undefined⟩ [Text markup commands], Seite ⟨undefined⟩, Abschnitt A.7 [Die Feta-Schriftart], Seite 578.

5.5.4 Formen verändern
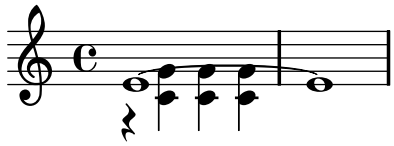
Bögen verändern

Binde-, Legato- und Phrasierungsbögen werden als Bézierkurven dritter Ordnung gezeichnet. Wenn die Form eines automatischen Bogens nicht optimal ist, kann sie manuell verändert werdne, indem man die vier erforderlichen Kontrollpunkte angibt.

Bézierkurven dritter Ordnung (auch als quadratische Bézierkurven bezeichnet) werden durch vier Kontrollpunkte definiert. Der erste und vierte Kontrollpunkt geben Beginn und Ende der Kurve an. Die zwei Punkte dazwischen werden benutzt, um die Form der Kurve zu bestimmen. Im Internet gibt es Animationen, die illustrieren, wie eine derartige Kurve gezeichnet wird, aber die folgende Beschreibung kann hilfreich sein. Die Kurve beginnt am ersten Kontrollpunkt in Richtung des zweiten, wobei sie sich schrittweise krümmt um zum dritten Kontrollpunkt zu gelangen, von wo aus sie sich weiter zum vierten Punkt hin krümmt. Die Form der Kurve wird vollständig von den vier Punkten definiert.

Hier ein Beispiel eines Falles, in dem der Bogen nicht optimal erscheint, und wo auch `\tieDown` das Problem nicht lösen würde.

```
<<
  { e1 ~ e }
\\
  { r4 <g c,> <g c,> <g c,> }
>>
```
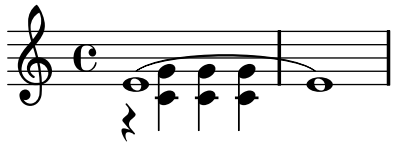


Eine Möglichkeit, diesen Bogen zu verbessern, ist es, seine Kontrollpunkte manuell zu verändern:

Die Koordinaten von Bézierkontrollpunkten werden in Notenlinienzwischenräumen angegeben. Die X-Achse ist relativ zum Referenzpunkt der Note, an die der Bogen angefügt wird, und die Y-Achse relativ zur Mittellinie des Notensystems. Die Koordinaten werden als eine Liste von vier Paaren an realen Dezimalzahlen eingegeben. Eine Möglichkeit ist es, die Koordinaten der zwei Endpunkte zu schätzen und dann die zwei Zwischenpunkte zu erraten. Die optimalen Werte können nur durch Ausprobieren gefunden werden.

Es lohnt sich daran zu denken, dass eine symmetrische Kurve symmetrische Kontrollpunkte benötigt, und dass Bézierkurven die nützliche Eigenschaft haben, dass eine Transformation der Kurve wie eine Übersetzung, Drehung oder Skalierung der Kurve erreicht werden kann, indem man die gleiche Skalierung auf die Kontrollpunkte anwendet.

In dem obigen Beispiel geben folgende Werte einen zufriedenstellenden Bogen – Achtung: der Befehl muss direkt vor dem Beginn der Note gesetzt werden, an die der (Binde-)Bogen angehängt wird.

```
<<
  {
  \once \override Tie
    #'control-points = #'((1 . -1) (3 . 0.6) (12.5 . 0.6) (14.5 . -1))
  e1 ~ e1
  }
\\
  { r4 <g c,> <g c,> <g c,>4  }
>>
```



Bekannte Probleme und Warnungen

Es ist nicht möglich, die Form von Bögen anhand ihrer `control-points`-Eigenschaft zu verändern, wenn mehr als ein Bogen zum gleichen musikalischen Moment auftritt, nicht einmal mit dem `\tweak`-Befehl.

5.6 Musikfunktionen benutzen

Wenn Optimierungen von unterschiedlichen musikalischen Ausdrücken wiederverwendet werden sollen, bietet es sich oft an, den „Optimierungsanteil“ einer *musikalischen Funktion* zu erstellen. In diesem Abschnitt sollen nur *Ersetzungen* erklärt werden, wo es darum geht, eine Variable mit einem Stück LilyPond-Code zu ersetzen. Andere komplexere Funktionen werden beschrieben in Abschnitt “Musikalische Funktionen” in *Extending*.

5.6.1 Syntax der Ersetzungsfunktion

Es ist einfach eine Funktion zu erstellen, die eine Variable in LilyPond-Code umwandelt. Die generelle Form dieser Funktionen ist:

```
Funktion =
#(define-music-function
     (parser location Arg1 Arg2 ...)
     (Typ1? Typ2? ...)
   #{
     ...Noten...
   #})
```

wobei

ArgN	*n*tes Argument
TypN?	ein Scheme *Typenprädikat*, für das *ArgN* den Wert `#t` ausgibt.
...Noten...	normale LilyPond-Eingab, wobei `$` benutzt wird, um Argumente zu referenzieren (etwa ‘`$Arg1`’).

Die `parser` und `location`-Argumente sind zwingend und werden in einigen fortgeschrittenen Situationen benutzt, wie sie im „Erweitern“-Handbuch beschrieben werden (siehe Abschnitt “Musikalische Funktionen” in *Extending*). In Ersetzungsfunktionen gehen Sie einfach sicher, dass sie die beiden Wörter auch mit aufnehmen.

Die Liste der Typenprädikate ist auch notwendig. Einige der häufigsten Typenprädikate, die in musikalischen Funktionen benutzt werden, sind:

```
boolean?
cheap-list?  (benutze anstelle von ,list¿
 für schnelleres Kompilieren)
ly:music?
markup?
number?
pair?
string?
symbol?
```

Eine Liste aller Typprädikate findet sich unter ⟨undefined⟩ [Vordefinierte Typprädikate], Seite ⟨undefined⟩. Eigene Typprädikate sind auch erlaubt.

Siehe auch

Notationsreferenz: ⟨undefined⟩ [Vordefinierte Typprädikate], Seite ⟨undefined⟩.

Erweitern: Abschnitt “Musikalische Funktionen” in *Extending*.

Installierte Dateien: ‘`lily/music-scheme.cc`’, ‘`scm/c++.scm`’, ‘`scm/lily.scm`’.

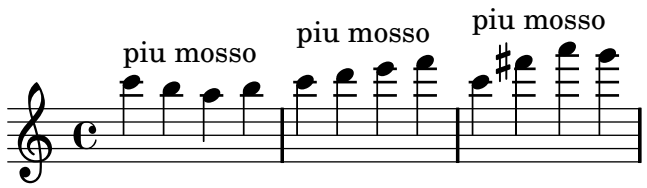
5.6.2 Beispiele der Ersetzungsfunktionen

Dieser Abschnitt zeigt einige Beispiele von Ersetzungsfunktionen. Sie sind nicht vollständig, sondern sollen einige der Möglichkeiten von einfachen Ersetzungsfunktionen aufzeigen.

Im ersten Beispiel wird eine Funktione definiert, die das Verschieben von `TextScript` erleichtert:

```
padText =
#(define-music-function
     (parser location padding)
     (number?)
   #{
     \once \override TextScript #'padding = $padding
   #})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}
```
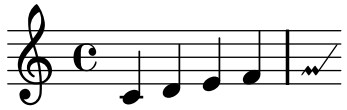


Neben Zahlen können auch musikalische Ausdrücke wie Noten als Argumente für musikalische Funktionen eingesetzt werden:

```
custosNote =
#(define-music-function
     (parser location note)
```

```
    (ly:music?)
  #{
    \once \override Voice.NoteHead #'stencil =
      #ly:text-interface::print
    \once \override Voice.NoteHead #'text =
      \markup \musicglyph #"custodes.mensural.u0"
    \once \override Voice.Stem #'stencil = ##f
    $note
  #})

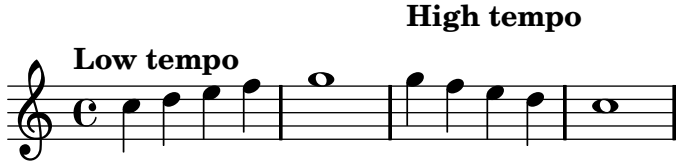
\relative c' { c4 d e f \custosNote g }
```



Ersetzungsfunktionen mit mehrfachen Argumenten können definiert werden:

```
tempoPadded =
#(define-music-function
    (parser location padding tempotext)
    (number? string?)
  #{
    \once \override Score.MetronomeMark #'padding = $padding
    \tempo \markup { \bold $tempotext }
  #})

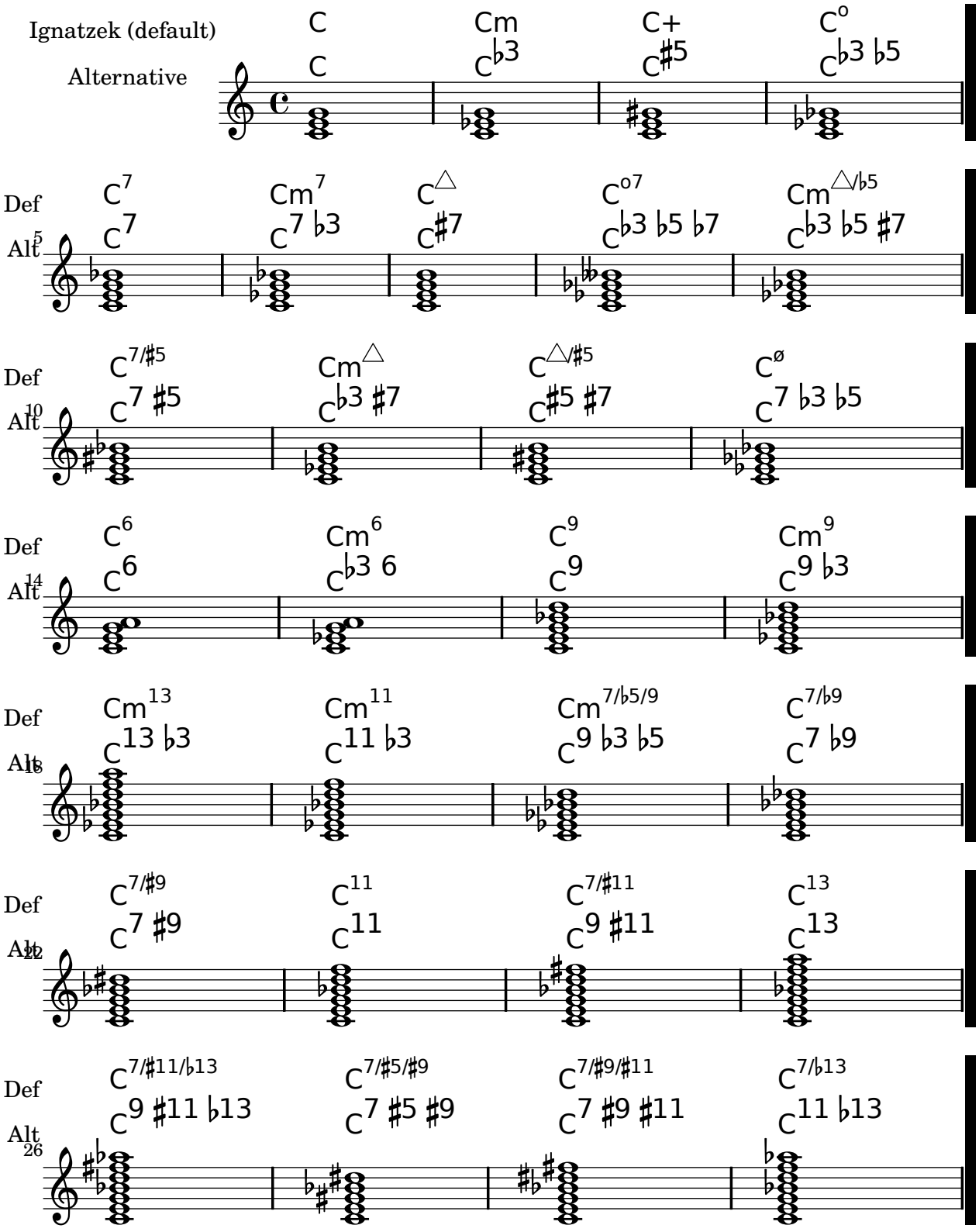
\relative c'' {
  \tempo \markup { "Low tempo" }
  c4 d e f g1
  \tempoPadded #4.0 #"High tempo"
  g4 f e d c1
}
```

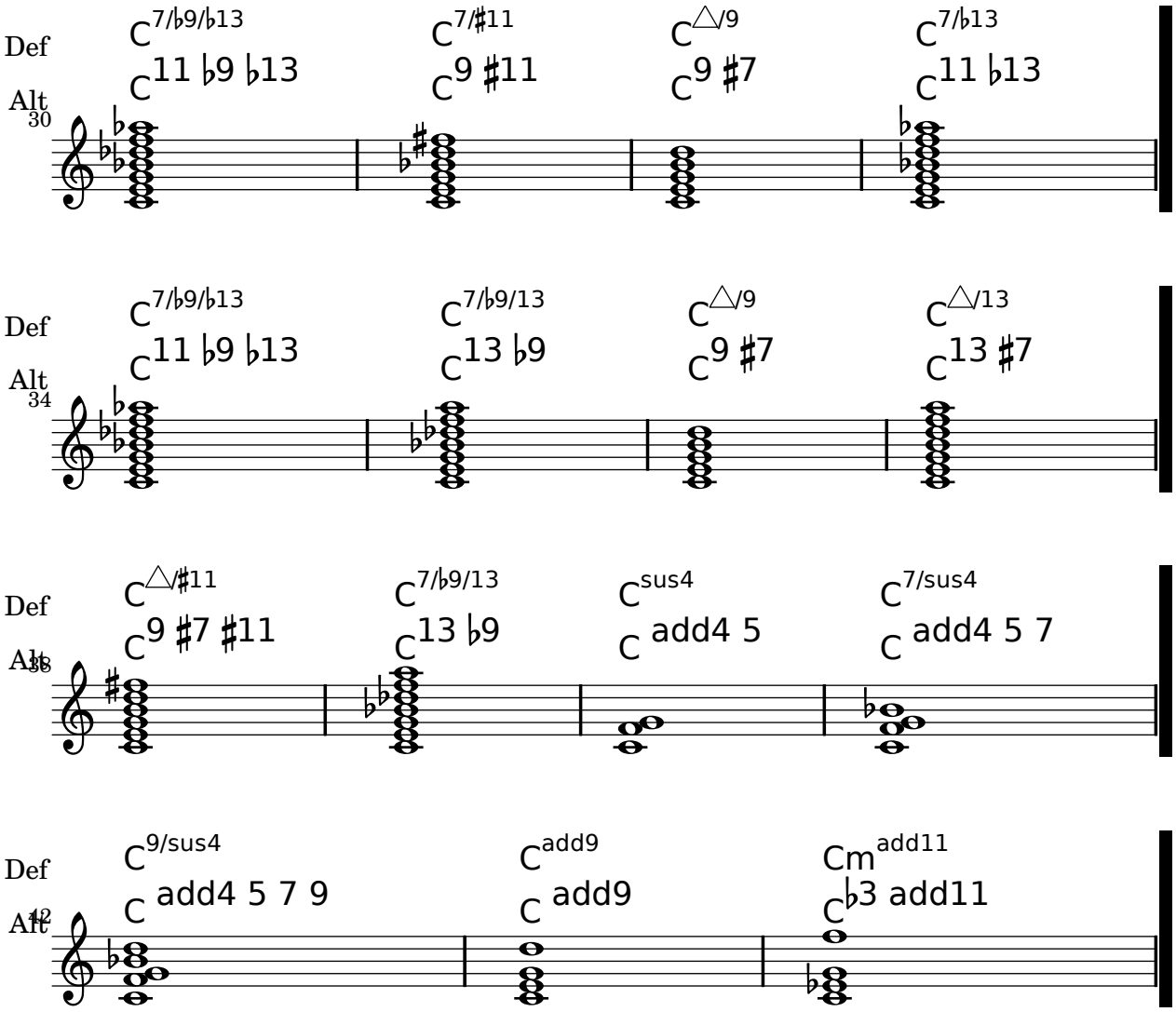


Anhang A Notationsübersicht

A.1 Liste der Akkordbezeichnungen

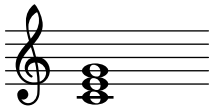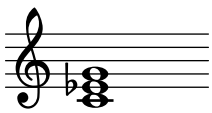
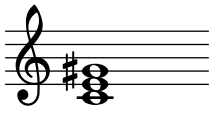
Die Tabelle zeigt die zwei üblichen Möglichkeiten, wie Akkordbezeichnungen ausgegeben werden. Es wird auch die entsprechende Note ausgegeben.
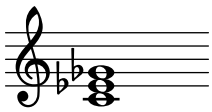
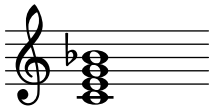
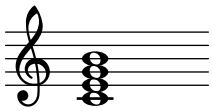
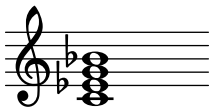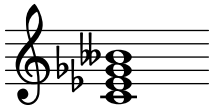
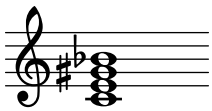
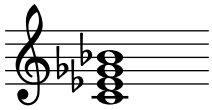
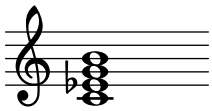
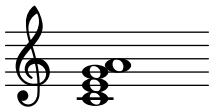
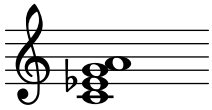


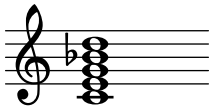
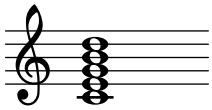
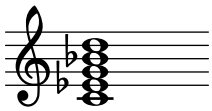
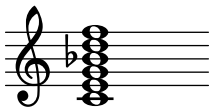
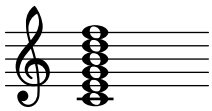
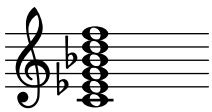
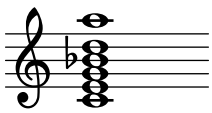
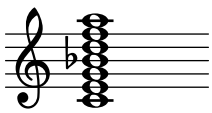
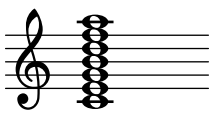
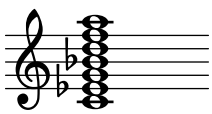


A.2 Übliche Akkord-Variablen

Die Tabelle zeigt Modifikatoren für Akkorde, die im `\chordmode`-Modus benutzt werden können, um übliche Akkordkonstrukte zu notieren.

Akkordtyp	**Intervalle**	**Modifikator(en)**	**Beispiel**
Dur	große Terz, Quinte	`5` oder nichts	
Moll	kleine Terz, Quinte	`m` oder `m5`	
Übermäßig	Große Terz, übermäßige Quinte	`aug`	

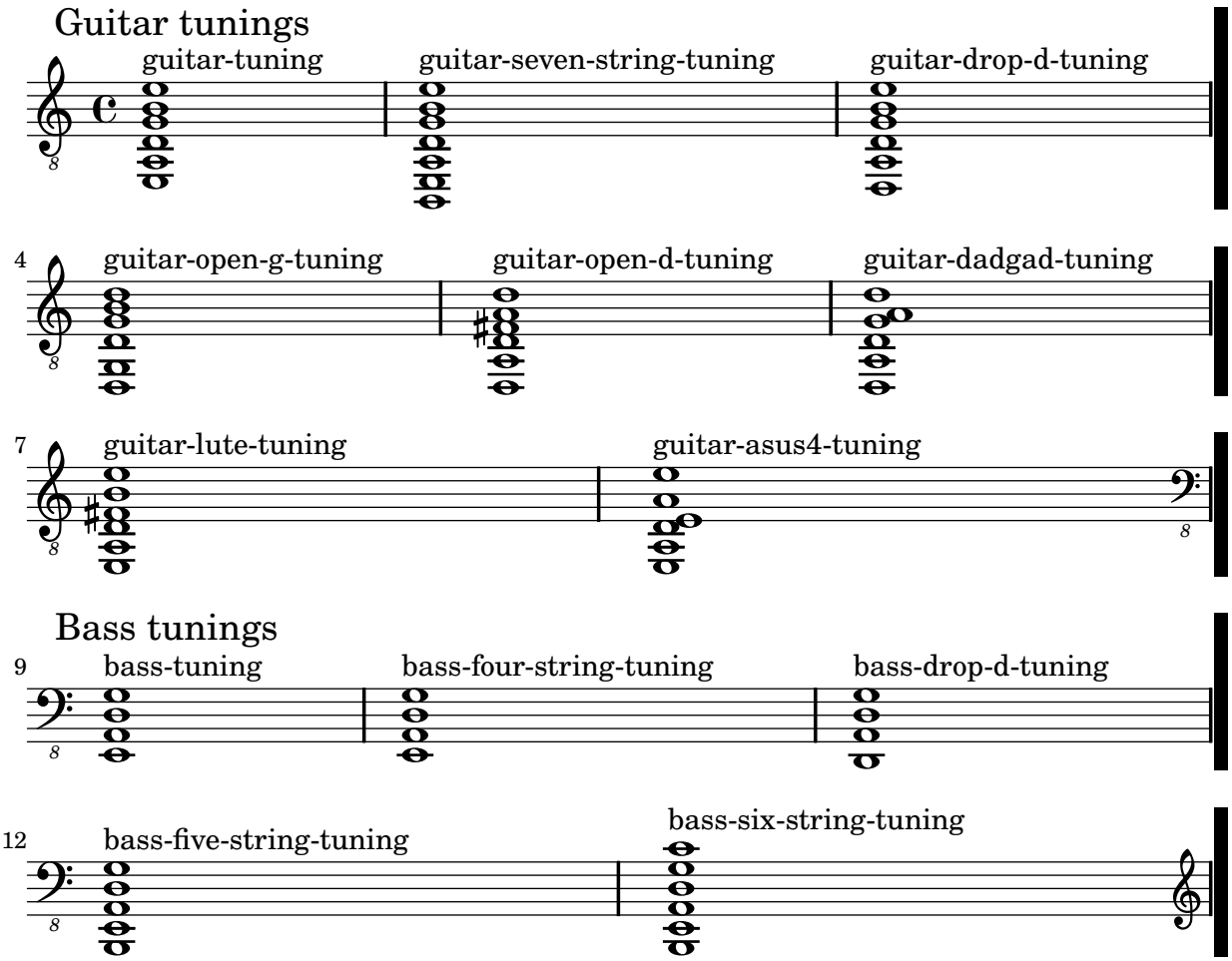
Vermindert	Kleine Terz, verminderte Quinte	`dim`	
Dominantsieben	Durdreiklang, kleine Septime	`7`	
Große Septime	Durdreiklang, große Septime	`maj7` oder `maj`	
Kleine Septime	Molldreiklang, kleine Septime	`m7`	
Verminderte Septime	Verminderter Dreiklang, verminderte Septime	`dim7`	
Übermäßige Septime	Übermäßiger Dreiklang, kleine Septime	`aug7`	
halbverminderte Septime	Verminderter Dreklang, kleine Sept	`m7.5-`	
Kleine MollSept	Molldreiklang, große Septime	`m7+`	
Große Sexte	Durdreiklang, Sexte	`6`	
Kleine Sexte	Molldreiklang, Sexte	`m6`	

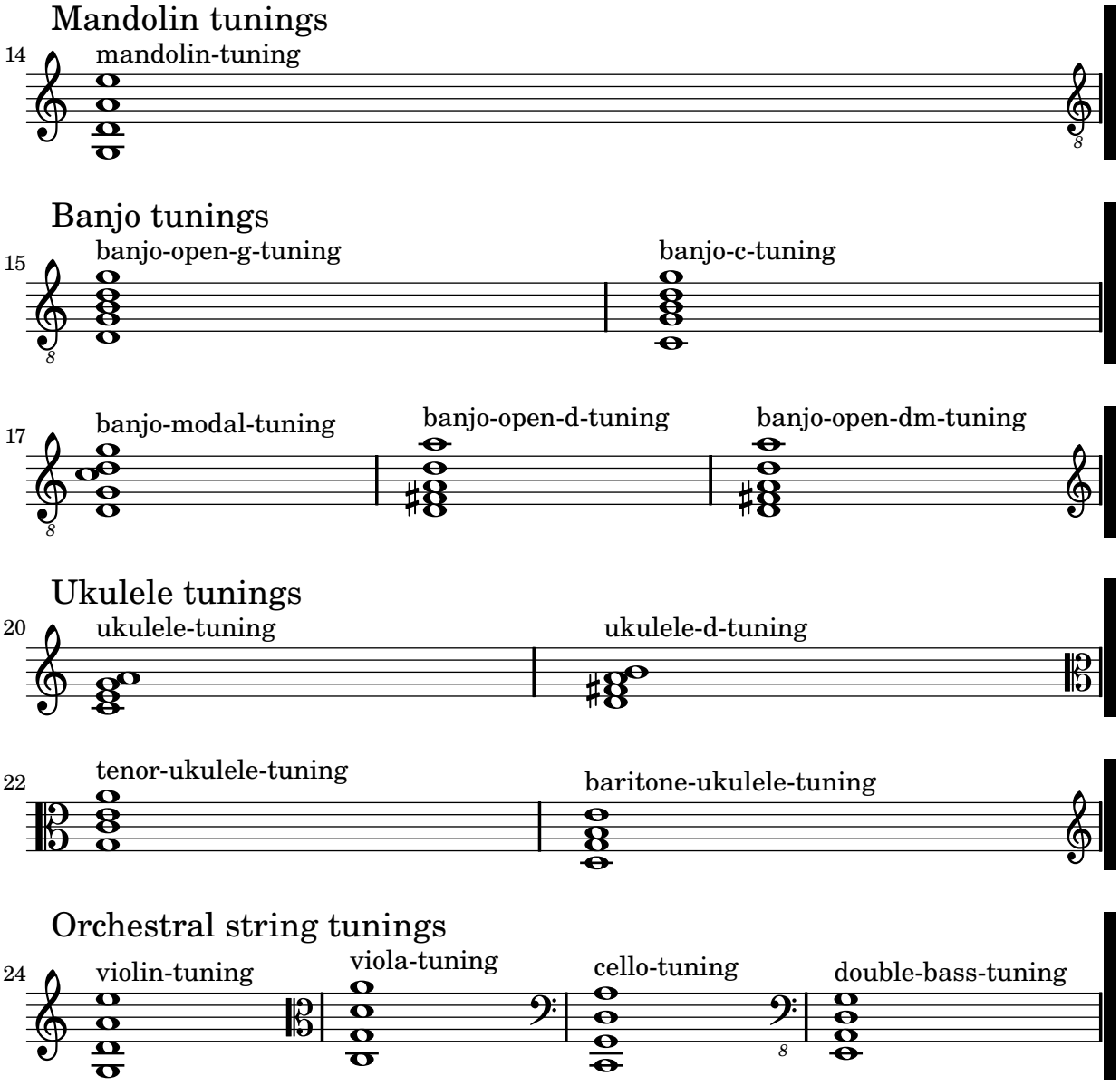
Dominantnone	Dominantsept, große None	`9`	
Dur-None	Große None, große Septime	`maj9`	
Moll-None	Große None, kleine Septime	`m9`	
Dominantundezime	Dominantnone, perfekte Undezime	`11`	
Durundezime	Große None, perfekte Undezime	`maj11`	
Mollundezime	Kleine None, perfekte Undezime	`m11`	
Dominant-13	Dominantnone, große 13	`13`	
Dominant-13	Dominant-Undezime, große 13	`13.11`	
Dur-13	Große Undezime, große 13	`maj13.11`	
Moll-13	Kleine Undezime, große 13	`m13.11`	

Sekundakkord	große Sekunde, perfekte Quinte	`sus2`
Quartakkord	perfekte Quarte, perfekte Quinte	`sus4`
Powerakkord (zweistimmig)	Perfekte Quinte	`1.5`
Powerakkord (dreistimmig)	Perfekte Quinte, Oktave	`1.5.8`

A.3 Vordefinierte Saitenstimmungen

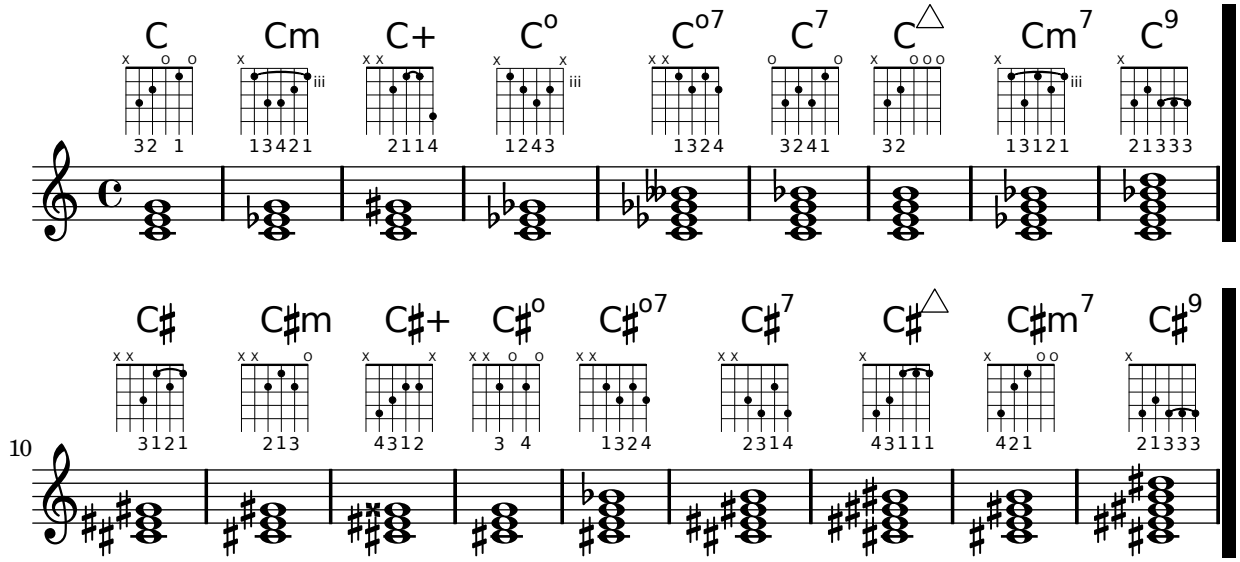
Die folgende Tabelle zeigt die vordefinierten Saitenstimmungen:

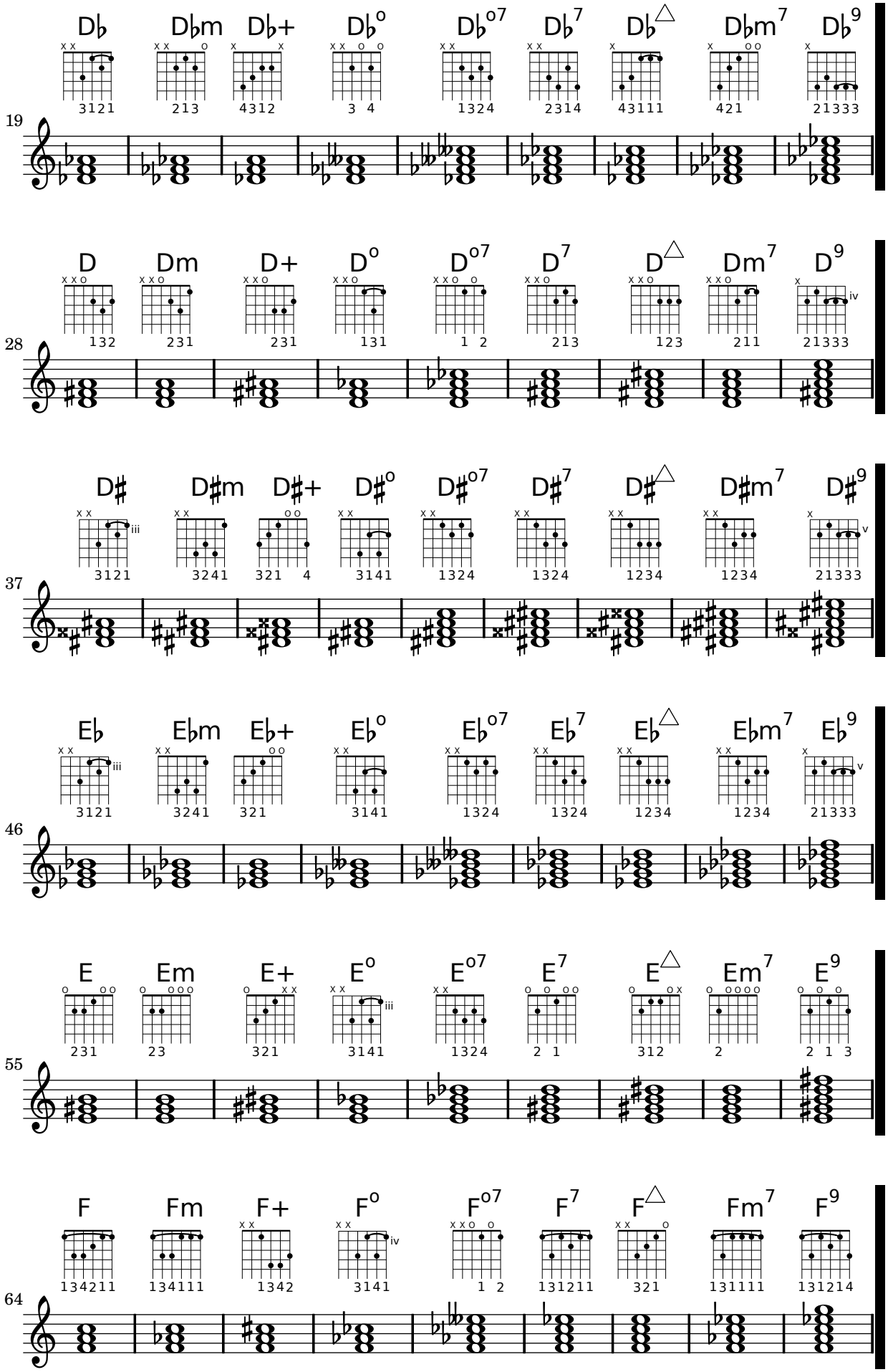


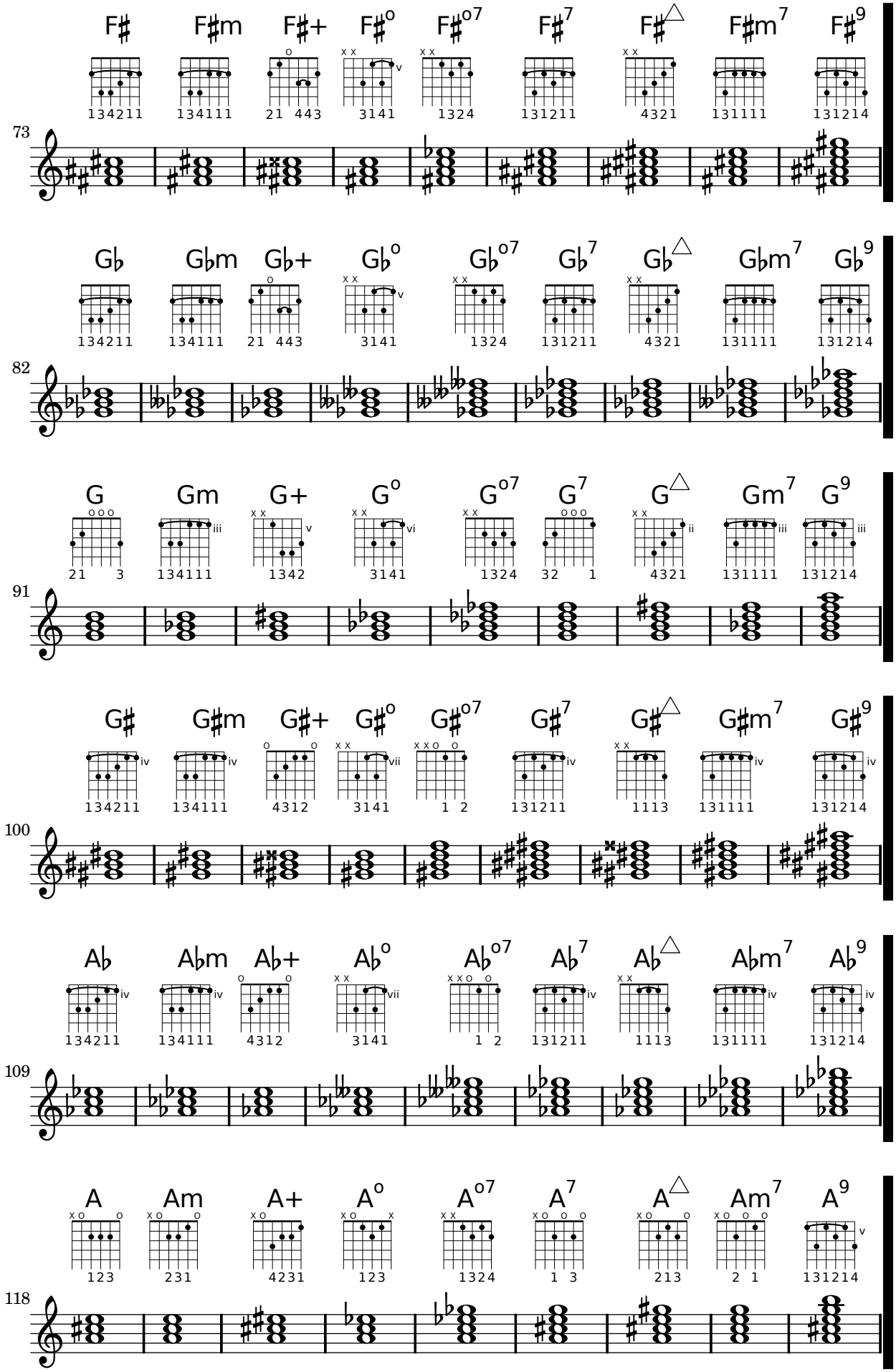


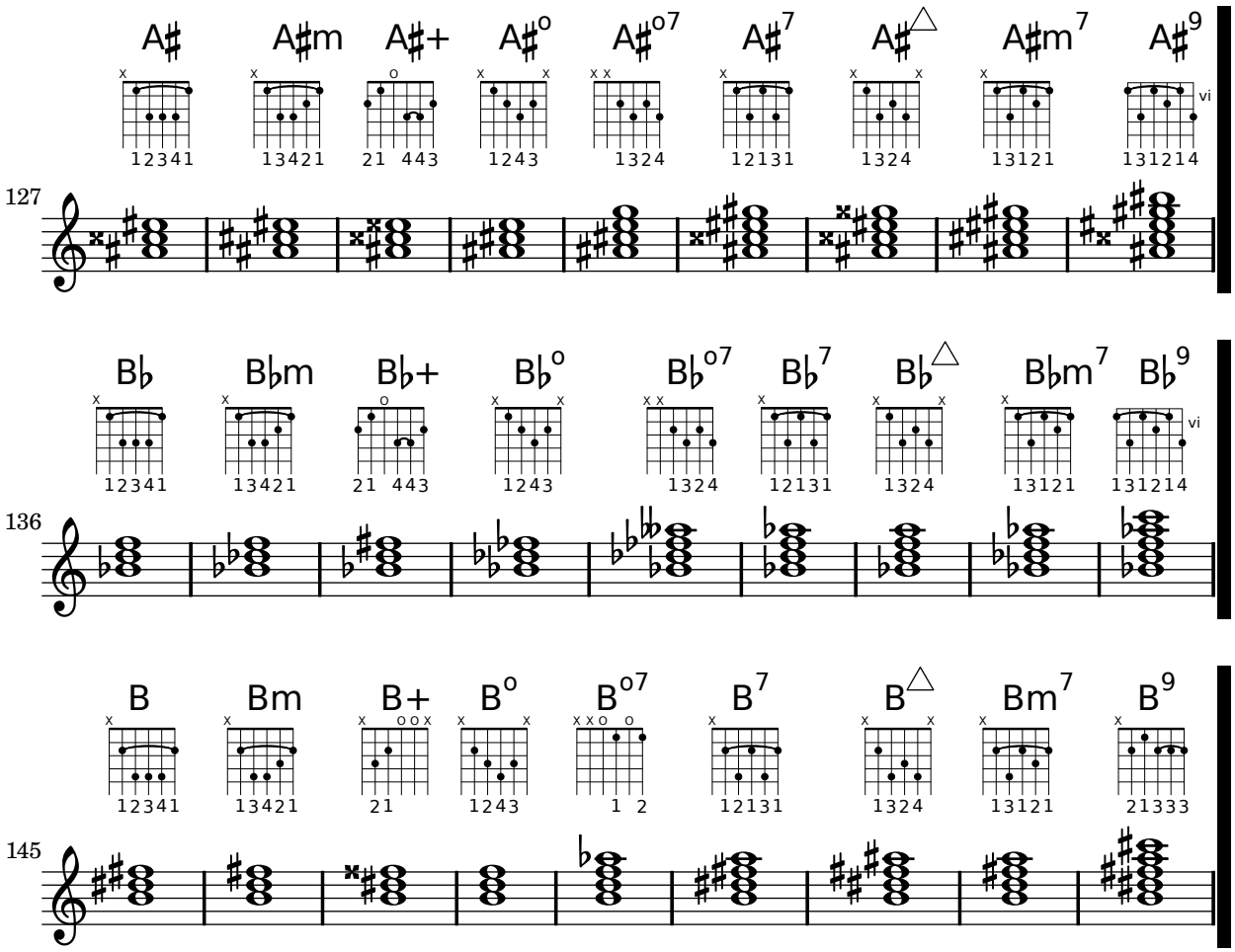
A.4 Die vordefinierten Bund-Diagramme

Die Tabelle zeigt alle vordefinierten Bunddiagramme für Gitarre.

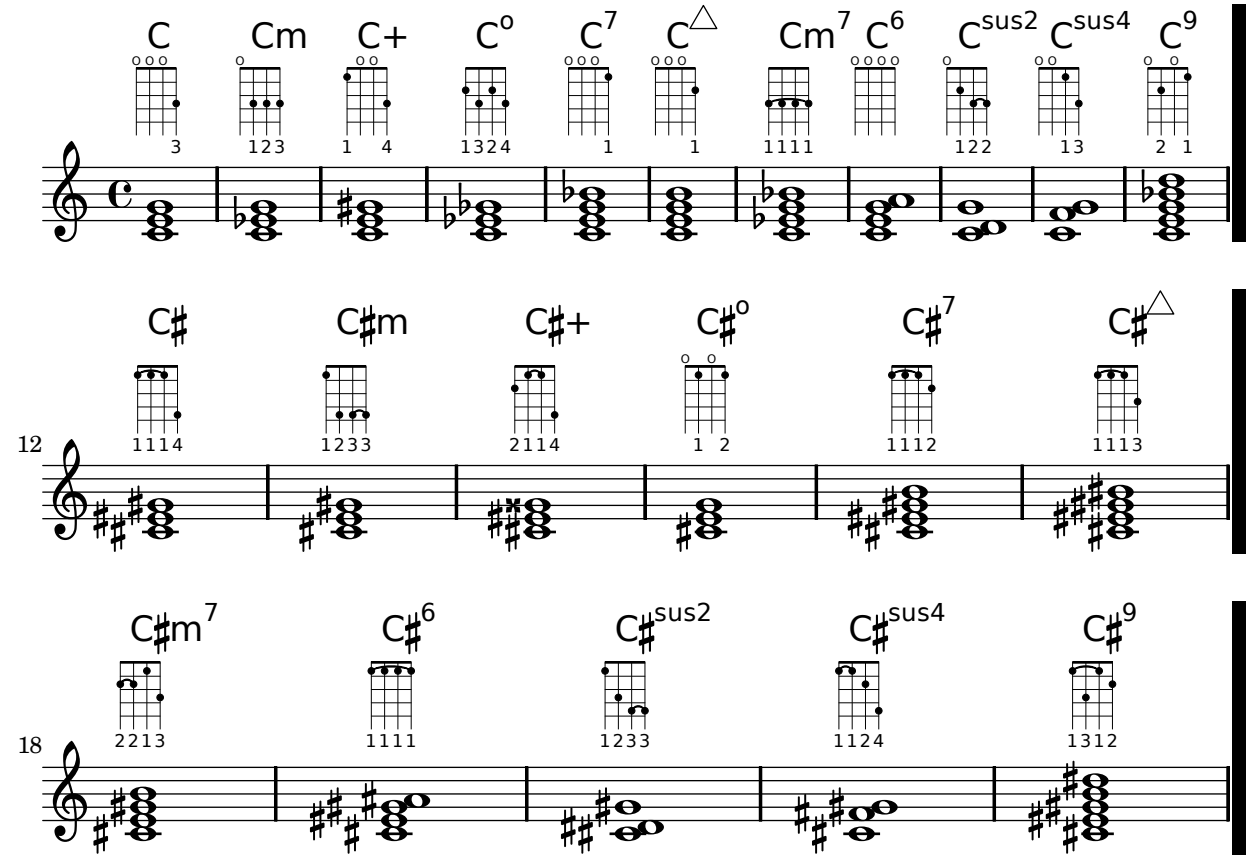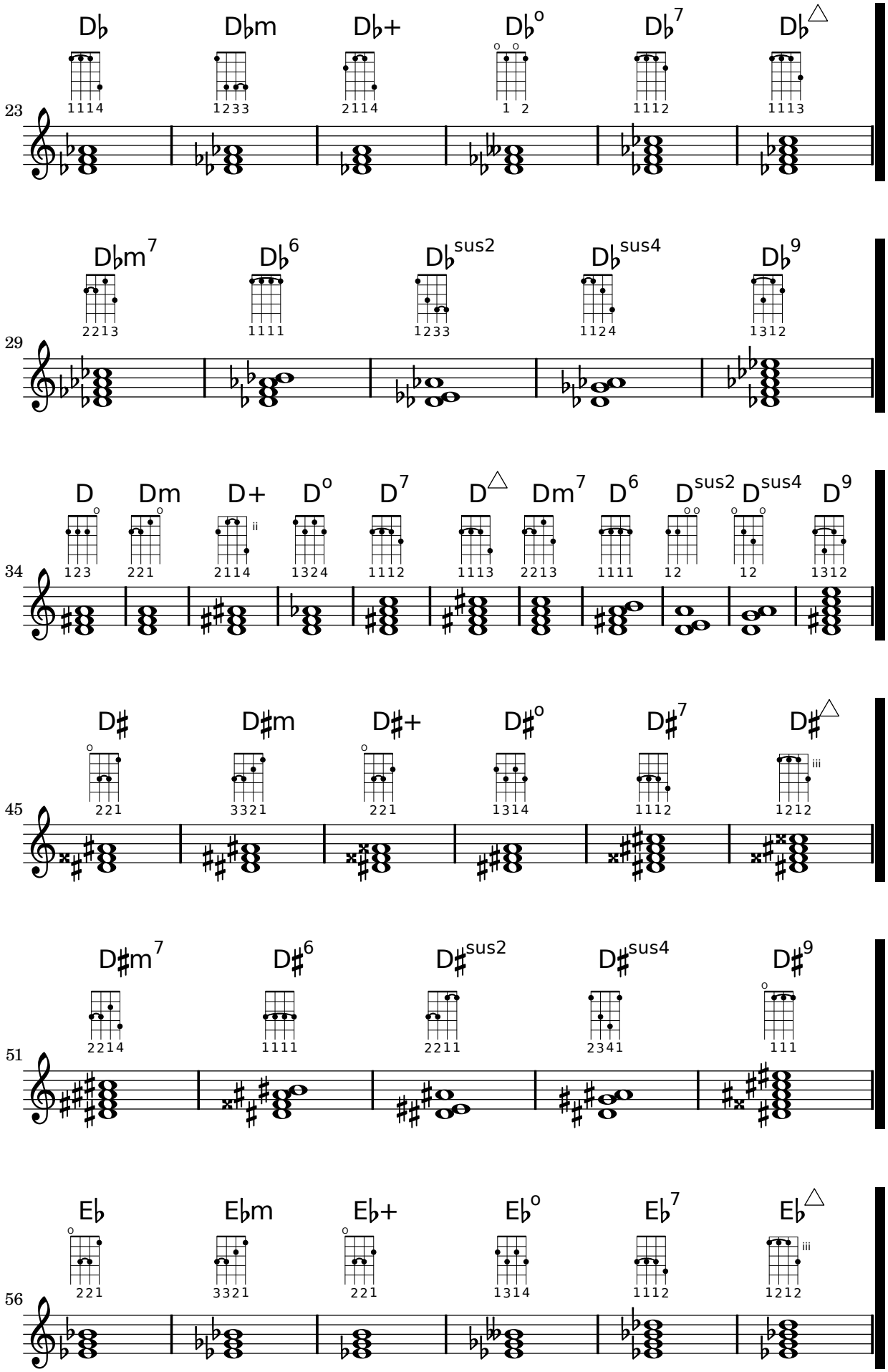



D♭ D♭m D♭+ D♭° D♭°7 D♭7 D♭△ D♭m7 D♭9
3121 213 4312 3 4 1324 2314 43111 421 21333
19
D Dm D+ D° D°7 D7 D△ Dm7 D9
132 231 231 131 1 2 213 123 211 21333
28
D♯ D♯m D♯+ D♯° D♯°7 D♯7 D♯△ D♯m7 D♯9
3121 3241 321 4 3141 1324 1324 1234 1234 21333
37
E♭ E♭m E♭+ E♭° E♭°7 E♭7 E♭△ E♭m7 E♭9
3121 3241 321 3141 1324 1324 1234 1234 21333
46
E Em E+ E° E°7 E7 E△ Em7 E9
231 23 321 3141 1324 2 1 312 2 2 1 3
55
F Fm F+ F° F°7 F7 F△ Fm7 F9
134211 134111 1342 3141 1 2 131211 321 131111 131214
64

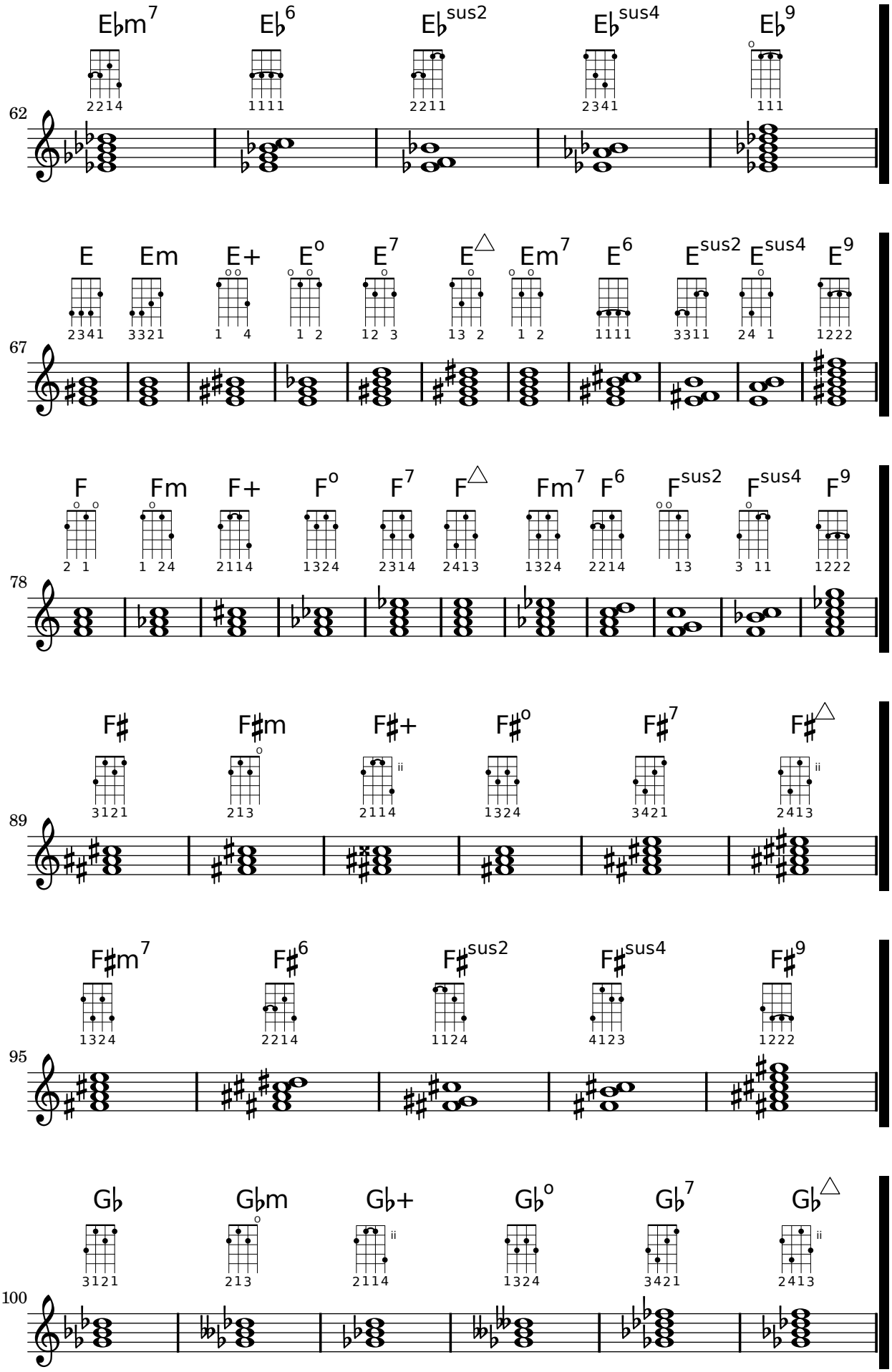

F♯ F♯m F♯+ F♯° F♯°7 F♯7 F♯△ F♯m7 F♯9
134211 134111 21 443 3141 1324 131211 4321 131111 131214
73
G♭ G♭m G♭+ G♭° G♭°7 G♭7 G♭△ G♭m7 G♭9
134211 134111 21 443 3141 1324 131211 4321 131111 131214
82
G Gm G+ G° G°7 G7 G△ Gm7 G9
21 3 134111 1342 3141 1324 32 1 4321 131111 131214
91
G♯ G♯m G♯+ G♯° G♯°7 G♯7 G♯△ G♯m7 G♯9
134211 134111 4312 3141 1 2 131211 1113 131111 131214
100
A♭ A♭m A♭+ A♭° A♭°7 A♭7 A♭△ A♭m7 A♭9
134211 134111 4312 3141 1 2 131211 1113 131111 131214
109
A Am A+ A° A°7 A7 A△ Am7 A9
123 231 4231 123 1324 1 3 213 2 1 131214
118

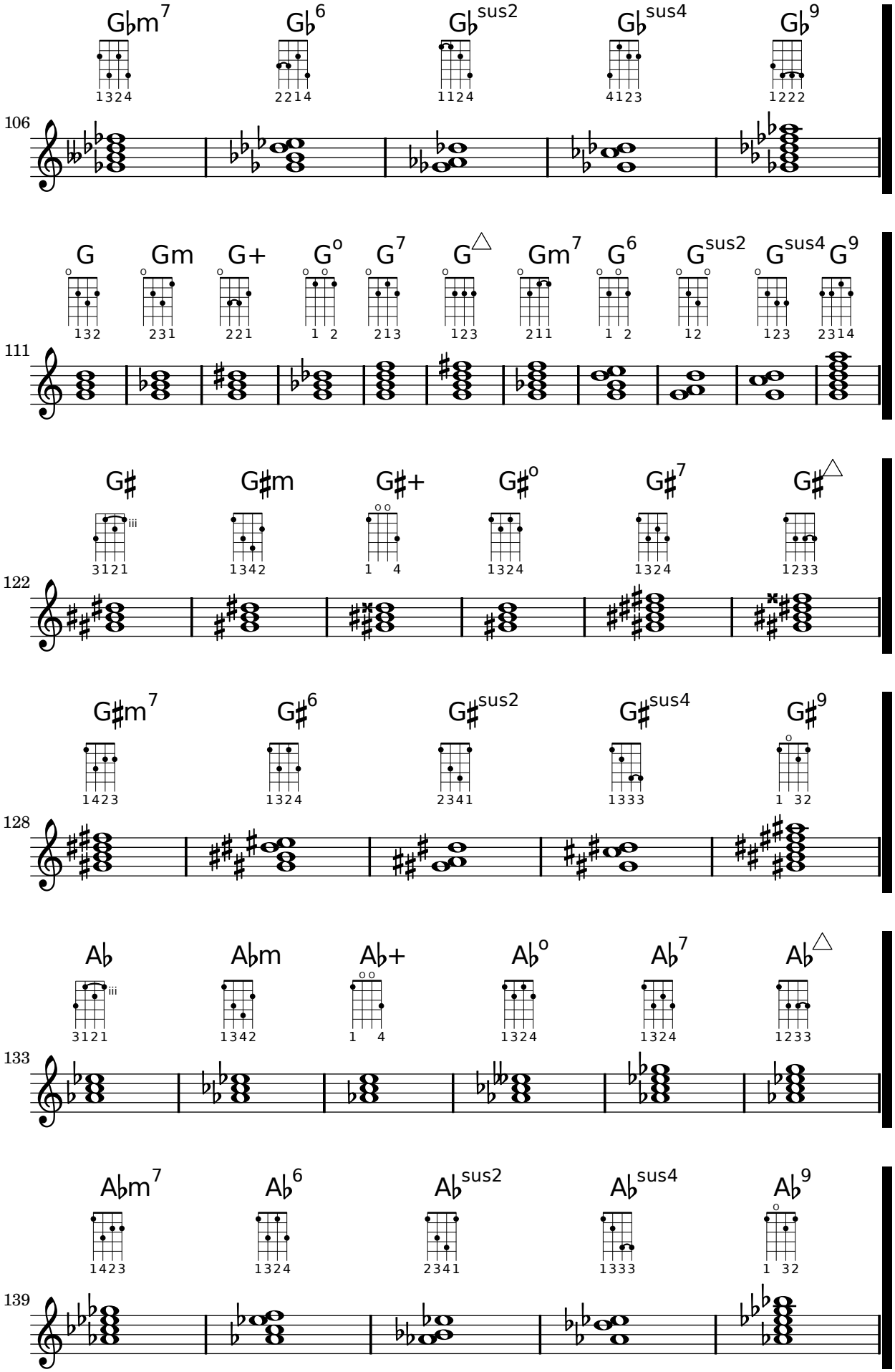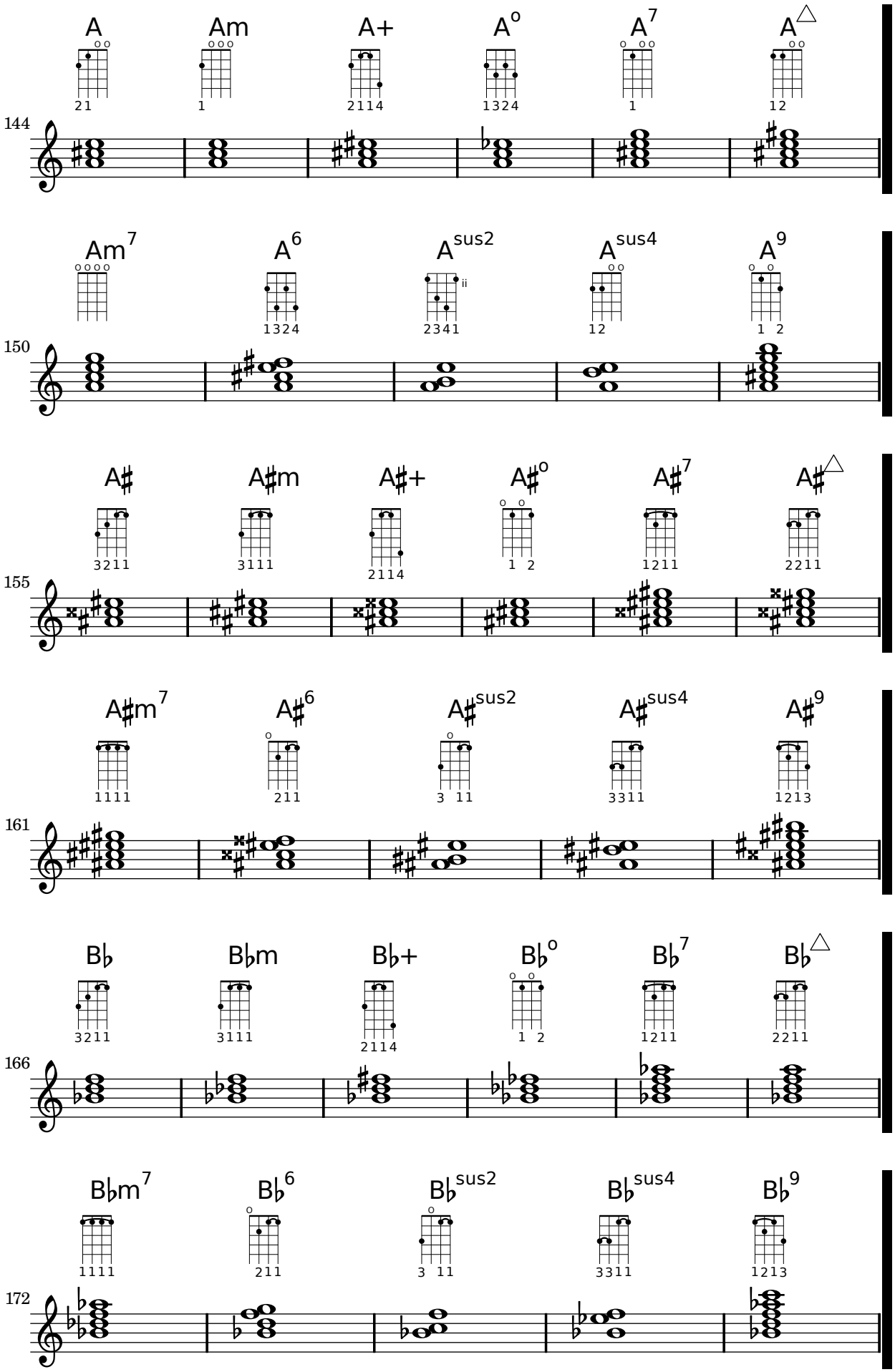


Die folgende Tabelle zeigt vordefinierte Bunddiagramme für Ukulele.

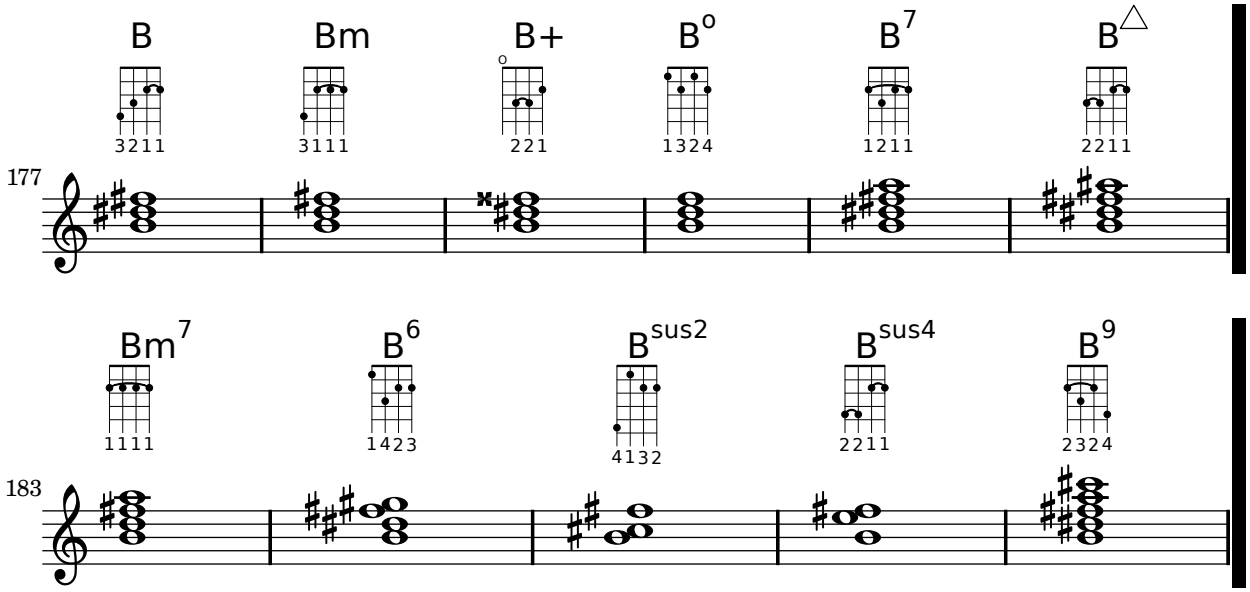



D♭ D♭m D♭+ D♭° D♭7 D♭△
1114 1233 2114 1 2 1112 1113
23
D♭m7 D♭6 D♭sus2 D♭sus4 D♭9
2213 1111 1233 1124 1312
29
D Dm D+ D° D7 D△ Dm7 D6 Dsus2 Dsus4 D9
123 221 2114 1324 1112 1113 2213 1111 12 12 1312
34
D♯ D♯m D♯+ D♯° D♯7 D♯△
221 3321 221 1314 1112 1212
45
D♯m7 D♯6 D♯sus2 D♯sus4 D♯9
2214 1111 2211 2341 111
51
E♭ E♭m E♭+ E♭° E♭7 E♭△
221 3321 221 1314 1112 1212
56


E♭m7 E♭6 E♭sus2 E♭sus4 E♭9
2214 1111 2211 2341 111
62
E Em E+ Eo E7 E△ Em7 E6 Esus2 Esus4 E9
2341 3321 1 4 1 2 12 3 13 2 1 2 1111 3311 24 1 1222
67
F Fm F+ Fo F7 F△ Fm7 F6 Fsus2 Fsus4 F9
2 1 1 24 2114 1324 2314 2413 1324 2214 13 3 11 1222
78
F♯ F♯m F♯+ F♯o F♯7 F♯△
3121 213 2114 1324 3421 2413
89
F♯m7 F♯6 F♯sus2 F♯sus4 F♯9
1324 2214 1124 4123 1222
95
G♭ G♭m G♭+ G♭o G♭7 G♭△
3121 213 2114 1324 3421 2413
100


G♭m7 G♭6 G♭sus2 G♭sus4 G♭9
1324 2214 1124 4123 1222
106
G Gm G+ Go G7 G△ Gm7 G6 Gsus2 Gsus4 G9
132 231 221 1 2 213 123 211 1 2 12 123 2314
111
G♯ G♯m G♯+ G♯o G♯7 G♯△
iii
3121 1342 1 4 1324 1324 1233
122
G♯m7 G♯6 G♯sus2 G♯sus4 G♯9
1423 1324 2341 1333 1 32
128
A♭ A♭m A♭+ A♭o A♭7 A♭△
iii
3121 1342 1 4 1324 1324 1233
133
A♭m7 A♭6 A♭sus2 A♭sus4 A♭9
1423 1324 2341 1333 1 32
139


A
Am
A+
A°
A7
A△
144
Am7
A6
Asus2
Asus4
A9
150
A♯
A♯m
A♯+
A♯°
A♯7
A♯△
155
A♯m7
A♯6
A♯sus2
A♯sus4
A♯9
161
B♭
B♭m
B♭+
B♭°
B♭7
B♭△
166
B♭m7
B♭6
B♭sus2
B♭sus4
B♭9
172



A.5 MIDI-Instrumente

Hier eine Liste von Musikinstrumentenbezeichnungen, die als Name für `midiInstrument` benutzt werden können. Die Anordnung der Instrumente entspricht den 128 Prgrammnummern des MIDI-Standards.

```
acoustic grand
bright acoustic
electric grand
honky-tonk
electric piano 1
electric piano 2
harpsichord
clav
celesta
glockenspiel
music box
vibraphone
marimba
xylophone
tubular bells
dulcimer
drawbar organ
percussive organ
rock organ
church organ
reed organ
accordion
harmonica
concertina
acoustic guitar (nylon)
acoustic guitar (steel)
electric guitar (jazz)
electric guitar (clean)
electric guitar (muted)
overdriven guitar
distorted guitar
guitar harmonics
contrabass
tremolo strings
pizzicato strings
orchestral harp
timpani
string ensemble 1
string ensemble 2
synthstrings 1
synthstrings 2
choir aahs
voice oohs
synth voice
orchestra hit
trumpet
trombone
tuba
muted trumpet
french horn
brass section
synthbrass 1
synthbrass 2
soprano sax
alto sax
tenor sax
baritone sax
oboe
english horn
bassoon
clarinet
piccolo
flute
recorder
lead 7 (fifths)
lead 8 (bass+lead)
pad 1 (new age)
pad 2 (warm)
pad 3 (polysynth)
pad 4 (choir)
pad 5 (bowed)
pad 6 (metallic)
pad 7 (halo)
pad 8 (sweep)
fx 1 (rain)
fx 2 (soundtrack)
fx 3 (crystal)
fx 4 (atmosphere)
fx 5 (brightness)
fx 6 (goblins)
fx 7 (echoes)
fx 8 (sci-fi)
sitar
banjo
shamisen
koto
kalimba
bagpipe
fiddle
shanai
tinkle bell
agogo
steel drums
woodblock
taiko drum
melodic tom
```

```
acoustic bass              pan flute             synth drum
electric bass (finger)     blown bottle          reverse cymbal
electric bass (pick)       shakuhachi            guitar fret noise
fretless bass              whistle               breath noise
slap bass 1                ocarina               seashore
slap bass 2                lead 1 (square)       bird tweet
synth bass 1               lead 2 (sawtooth)     telephone ring
synth bass 2               lead 3 (calliope)     helicopter
violin                     lead 4 (chiff)        applause
viola                      lead 5 (charang)      gunshot
cello                      lead 6 (voice)
```

A.6 Liste der Farben

Normale Farben

Die Syntax zur Benutzung findet sich im Abschnitt [Farbige Objekte], Seite 194.

```
black       white          red         green
blue        cyan           magenta     yellow
grey        darkred        darkgreen   darkblue
darkcyan    darkmagenta    darkyellow
```

X-Farbbezeichnungen

X-Farbbezeichnungen haben verschiedene Varianten:

Alle Bezeichnungen, die als einziges Wort mit Großbuchstaben geschrieben werden (bspw. ‚LightSlateBlue‘), können auch von Leerzeichen getrennt geschrieben werden (also ‚light slate blue‘).

Das Wort ‚grey‘ kann in jedem Fall auch ‚gray‘ geschrieben werden (bspw. ‚DarkSlateGray‘).

Manche Bezeichnungen können auch ein numerales Suffix tragen (etwa ‚LightSalmon4‘).

Farben ohne eine numerale Endung

```
snow              GhostWhite        WhiteSmoke        gainsboro         FloralWhite
OldLace           linen             AntiqueWhite      PapayaWhip        BlanchedAlmond
bisque            PeachPuff         NavajoWhite       moccasin          cornsilk
ivory             LemonChiffon      seashell          honeydew          MintCream
azure             AliceBlue         lavender          LavenderBlush     MistyRose
white             black             DarkSlateGrey     DimGrey           SlateGrey
LightSlateGrey    grey              LightGrey         MidnightBlue      navy
NavyBlue          CornflowerBlue    DarkSlateBlue     SlateBlue         MediumSlateBlue
LightSlateBlue    MediumBlue        RoyalBlue         blue              DodgerBlue
DeepSkyBlue       SkyBlue           LightSkyBlue      SteelBlue         LightSteelBlue
LightBlue         PowderBlue        PaleTurquoise     DarkTurquoise     MediumTurquoise
turquoise         cyan              LightCyan         CadetBlue         MediumAquamarine
aquamarine        DarkGreen         DarkOliveGreen    DarkSeaGreen      SeaGreen
MediumSeaGreen    LightSeaGreen     PaleGreen         SpringGreen       LawnGreen
green             chartreuse        MediumSpringGreen       GreenYellow     LimeGreen
YellowGreen       ForestGreen       OliveDrab         DarkKhaki         khaki
PaleGoldenrod     LightGoldenrodYellow    LightYellow     yellow    gold
LightGoldenrod    goldenrod         DarkGoldenrod     RosyBrown         IndianRed
SaddleBrown       sienna            peru              burlywood         beige
wheat             SandyBrown        tan               chocolate         firebrick
```

```
brown           DarkSalmon      salmon          LightSalmon     orange
DarkOrange      coral           LightCoral      tomato          OrangeRed
red             HotPink         DeepPink        pink            LightPink
PaleVioletRed   maroon          MediumVioletRed VioletRed       magenta
violet          plum            orchid          MediumOrchid    DarkOrchid
DarkViolet      BlueViolet      purple          MediumPurple    thistle
DarkGrey        DarkBlue        DarkCyan        DarkMagenta     DarkRed
LightGreen
```

Farben mit einer numeralen Endung

Für die folgenden Bezeichnungen kann das Suffix N durch eine Zahl von 1–4 ersetzt werden.

```
snowN           seashellN       AntiqueWhiteN   bisqueN         PeachPuffN
NavajoWhiteN    LemonChiffonN   cornsilkN       ivoryN          honeydewN
LavenderBlushN  MistyRoseN      azureN          SlateBlueN      RoyalBlueN
blueN           DodgerBlueN     SteelBlueN      DeepSkyBlueN    SkyBlueN
LightSkyBlueN   LightSteelBlueN LightBlueN      LightCyanN      PaleTurquoiseN
CadetBlueN      turquoiseN      cyanN           aquamarineN     DarkSeaGreenN
SeaGreenN       PaleGreenN      SpringGreenN    greenN          chartreuseN
OliveDrabN      DarkOliveGreenN khakiN          LightGoldenrodN LightYellowN
yellowN         goldN           goldenrodN      DarkGoldenrodN  RosyBrownN
IndianRedN      siennaN         burlywoodN      wheatN          tanN
chocolateN      firebrickN      brownN          salmonN         LightSalmonN
orangeN         DarkOrangeN     coralN          tomatoN         OrangeRedN
redN            DeepPinkN       HotPinkN        pinkN           LightPinkN
PaleVioletRedN  maroonN         VioletRedN      magentaN        orchidN
plumN           MediumOrchidN   DarkOrchidN     purpleN         MediumPurpleN
thistleN
```

Grauskala

Eine Grauskala kann mit der Bezeichnung

```
greyN
```

erstellt werden, wobei N eine Zahl von 0–100 darstellt.

A.7 Die Feta-Schriftart

Die folgenden Symbole sind als Emmentaler-Schriftart verfügbar; auf sie kann direkt zugegriffen werden, indem man die übliche Textbeschriftung benutzt. `\musicglyph` greift direkt auf die Notationsschriftart zu (bspw. `g^\markup { \musicglyph #"scripts.segno" }`). Siehe auch Abschnitt 1.8.2 [Text formatieren], Seite 210.

Notenschlüssel-Glyphen

`clefs.C`		`clefs.C_change`	
`clefs.F`		`clefs.F_change`	
`clefs.G`		`clefs.G_change`	

clefs.percussion		clefs.percussion_change	
clefs.tab		clefs.tab_change	

Taktart-Glyphen

timesig.C44		timesig.C22	

Zahlen-Glyphen

plus		comma	
hyphen		period	
zero		one	
two		three	
four		five	
six		seven	
eight		nine	

Versetzungszeichen-Glyphen

accidentals.sharp		accidentals .sharp.arrowup	
accidentals .sharp.arrowdown		accidentals .sharp.arrowboth	
accidentals.sharp .slashslash.stem		accidentals.sharp .slashslashslash.stemstem	
accidentals.sharp .slashslashslash.stem		accidentals.sharp .slashslash.stemstemstem	

accidentals.natural		accidentals.natural.arrowup	
accidentals.natural.arrowdown		accidentals.natural.arrowboth	
accidentals.flat		accidentals.flat.arrowup	
accidentals.flat.arrowdown		accidentals.flat.arrowboth	
accidentals.flat.slash		accidentals.flat.slashslash	
accidentals.mirroredflat.flat		accidentals.mirroredflat	
accidentals.mirroredflat.backslash		accidentals.flatflat	
accidentals.flatflat.slash		accidentals.doublesharp	
accidentals.rightparen	)	accidentals.leftparen	(

Standard-Notenkopf-Glyphen

noteheads.uM2		noteheads.dM2	
noteheads.sM1		noteheads.s0	
noteheads.s1		noteheads.s2	

Spezielle Notenkopf-Glyphen

noteheads.sM1double		noteheads.s0diamond	
noteheads.s1diamond		noteheads.s2diamond	
noteheads.s0triangle		noteheads.d1triangle	
noteheads.u1triangle		noteheads.u2triangle	
noteheads.d2triangle		noteheads.s0slash	
noteheads.s1slash		noteheads.s2slash	
noteheads.s0cross		noteheads.s1cross	
noteheads.s2cross		noteheads.s2xcircle	
noteheads.s0harmonic		noteheads.s2harmonic	

Geformte Notenkopf-Glyphen

noteheads.s0do		noteheads.d1do	
noteheads.u1do		noteheads.d2do	
noteheads.u2do		noteheads.s0doThin	
noteheads.d1doThin		noteheads.u1doThin	
noteheads.d2doThin		noteheads.u2doThin	

noteheads.s0re		noteheads.u1re	
noteheads.d1re		noteheads.u2re	
noteheads.d2re		noteheads.s0reThin	
noteheads.u1reThin		noteheads.d1reThin	
noteheads.u2reThin		noteheads.d2reThin	
noteheads.s0mi		noteheads.s1mi	
noteheads.s2mi		noteheads.s0miMirror	
noteheads.s1miMirror		noteheads.s2miMirror	
noteheads.s0miThin		noteheads.s1miThin	
noteheads.s2miThin		noteheads.u0fa	
noteheads.d0fa		noteheads.u1fa	
noteheads.d1fa		noteheads.u2fa	
noteheads.d2fa		noteheads.u0faThin	
noteheads.d0faThin		noteheads.u1faThin	
noteheads.d1faThin		noteheads.u2faThin	

noteheads.d2faThin		noteheads.s0sol	
noteheads.s1sol		noteheads.s2sol	
noteheads.s0la		noteheads.s1la	
noteheads.s2la		noteheads.s0laThin	
noteheads.s1laThin		noteheads.s2laThin	
noteheads.s0ti		noteheads.u1ti	
noteheads.d1ti		noteheads.u2ti	
noteheads.d2ti		noteheads.s0tiThin	
noteheads.u1tiThin		noteheads.d1tiThin	
noteheads.u2tiThin		noteheads.d2tiThin	
noteheads.u0doFunk		noteheads.d0doFunk	
noteheads.u1doFunk		noteheads.d1doFunk	
noteheads.u2doFunk		noteheads.d2doFunk	
noteheads.u0reFunk		noteheads.d0reFunk	
noteheads.u1reFunk		noteheads.d1reFunk	

noteheads.u2reFunk		noteheads.d2reFunk	
noteheads.u0miFunk		noteheads.d0miFunk	
noteheads.u1miFunk		noteheads.d1miFunk	
noteheads.s2miFunk		noteheads.u0faFunk	
noteheads.d0faFunk		noteheads.u1faFunk	
noteheads.d1faFunk		noteheads.u2faFunk	
noteheads.d2faFunk		noteheads.s0solFunk	
noteheads.s1solFunk		noteheads.s2solFunk	
noteheads.s0laFunk		noteheads.s1laFunk	
noteheads.s2laFunk		noteheads.u0tiFunk	
noteheads.d0tiFunk		noteheads.u1tiFunk	
noteheads.d1tiFunk		noteheads.u2tiFunk	
noteheads.d2tiFunk		noteheads.s0doWalker	
noteheads.u1doWalker		noteheads.d1doWalker	
noteheads.u2doWalker		noteheads.d2doWalker	

noteheads.s0reWalker		noteheads.u1reWalker	
noteheads.d1reWalker		noteheads.u2reWalker	
noteheads.d2reWalker		noteheads.s0miWalker	
noteheads.s1miWalker		noteheads.s2miWalker	
noteheads.s0faWalker		noteheads.u1faWalker	
noteheads.d1faWalker		noteheads.u2faWalker	
noteheads.d2faWalker		noteheads.s0laWalker	
noteheads.s1laWalker		noteheads.s2laWalker	
noteheads.s0tiWalker		noteheads.u1tiWalker	
noteheads.d1tiWalker		noteheads.u2tiWalker	
noteheads.d2tiWalker			

Pausen-Glyphen

rests.0		rests.1	
rests.0o		rests.1o	
rests.M3		rests.M2	
rests.M1		rests.2	

rests.2classical		rests.3	
rests.4		rests.5	
rests.6		rests.7	

Fähnchen-Glyphen

flags.u3		flags.u4	
flags.u5		flags.u6	
flags.u7		flags.d3	
flags.d4		flags.d5	
flags.d6		flags.d7	
flags.ugrace		flags.dgrace	

Punkt-Glyphen

dots.dot	

Dynamik-Glyphen

space		f	
m		p	
r		s	
z			

Schrift-Glyphen

scripts.ufermata		scripts.dfermata	
scripts.ushortfermata		scripts.dshortfermata	
scripts.ulongfermata		scripts.dlongfermata	
scripts.uverylongfermata		scripts.dverylongfermata	
scripts.thumb		scripts.sforzato	
scripts.espr		scripts.staccato	
scripts.ustaccatissimo		scripts.dstaccatissimo	
scripts.tenuto		scripts.uportato	
scripts.dportato		scripts.umarcato	
scripts.dmarcato		scripts.open	
scripts.halfopen		scripts.stopped	
scripts.upbow		scripts.downbow	
scripts.reverseturn		scripts.turn	
scripts.trill		scripts.upedalheel	

scripts.dpedalheel		scripts.upedaltoe	
scripts.dpedaltoe		scripts.flageolet	
scripts.segno		scripts.varsegno	
scripts.coda		scripts.varcoda	
scripts.rcomma		scripts.lcomma	
scripts.rvarcomma		scripts.lvarcomma	
scripts.arpeggio		scripts.trill_element	
scripts.arpeggio .arrow.M1		scripts.arpeggio.arrow.1	
scripts.trilelement		scripts.prall	
scripts.mordent		scripts.prallprall	
scripts.prallmordent		scripts.upprall	
scripts.upmordent		scripts.pralldown	
scripts.downprall		scripts.downmordent	
scripts.prallup		scripts.lineprall	
scripts.caesura.curved		scripts.caesura.straight	

scripts.snappizzicato		scripts.ictus	
scripts.uaccentus		scripts.daccentus	
scripts.usemicirculus		scripts.dsemicirculus	
scripts.circulus		scripts.augmentum	
scripts .usignumcongruentiae		scripts .dsignumcongruentiae	

Pfeilkopf-Glyphen

arrowheads.open.01		arrowheads.open.0M1	
arrowheads.open.11		arrowheads.open.1M1	
arrowheads.close.01		arrowheads.close.0M1	
arrowheads.close.11		arrowheads.close.1M1	

Klammerspitzen-Glyphen

brackettips.up		brackettips.down	

Pedal-Glyphen

pedal.*		pedal.M	
pedal..		pedal.P	
pedal.d		pedal.e	
pedal.Ped			

Akkordeon-Glyphen

`accordion.discant`		`accordion.dot`	
`accordion.freebass`		`accordion.stdbass`	
`accordion.bayanbass`		`accordion.oldEE`	
`accordion.push`		`accordion.pull`	

Vaticana-Glyphen

`noteheads .svaticana.punctum`		`noteheads.svaticana .punctum.cavum`	
`noteheads.svaticana .linea.punctum`		`noteheads.svaticana .linea.punctum.cavum`	
`noteheads.svaticana .inclinatum`		`noteheads.svaticana.lpes`	
`noteheads .svaticana.vlpes`		`noteheads.svaticana.upes`	
`noteheads .svaticana.vupes`		`noteheads .svaticana.plica`	
`noteheads .svaticana.vplica`		`noteheads .svaticana.epiphonus`	
`noteheads.svaticana .vepiphonus`		`noteheads.svaticana .reverse.plica`	
`noteheads.svaticana .reverse.vplica`		`noteheads.svaticana .inner.cephalicus`	
`noteheads.svaticana .cephalicus`		`noteheads .svaticana.quilisma`	

clefs.vaticana.do

clefs.vaticana.do_change

clefs.vaticana.fa 

clefs.vaticana.fa_change 

custodes.vaticana.u0

custodes.vaticana.u1

custodes.vaticana.u2

custodes.vaticana.d0

custodes.vaticana.d1

custodes.vaticana.d2

accidentals.vaticanaM1

accidentals.vaticana0

dots.dotvaticana .

Medicaea-Glyphen

noteheads.smedicaea .inclinatum

noteheads .smedicaea.punctum

noteheads .smedicaea.rvirga

noteheads .smedicaea.virga

clefs.medicaea.do

clefs.medicaea.do_change

clefs.medicaea.fa

clefs.medicaea.fa_change

custodes.medicaea.u0

custodes.medicaea.u1

custodes.medicaea.u2

custodes.medicaea.d0

custodes.medicaea.d1

custodes.medicaea.d2

accidentals.medicaeaM1

Hufnagel-Glyphen

noteheads .shufnagel.punctum

noteheads .shufnagel.virga

noteheads.shufnagel.lpes

clefs.hufnagel.do

clefs.hufnagel.do_change

clefs.hufnagel.fa

clefs.hufnagel.fa_change

clefs.hufnagel.do.fa

clefs.hufnagel .do.fa_change

custodes.hufnagel.u0

custodes.hufnagel.u1

custodes.hufnagel.u2

custodes.hufnagel.d0

custodes.hufnagel.d1

custodes.hufnagel.d2

accidentals.hufnagelM1

Mensural-Glyphen

rests.M3mensural

rests.M2mensural

rests.M1mensural

rests.0mensural

rests.1mensural

rests.2mensural

rests.3mensural

rests.4mensural

noteheads.slmensural

noteheads.sM3mensural

noteheads.sM3ligmensural

noteheads.sM2mensural

noteheads.sM1mensural 

noteheads .sM3blackmensural 

noteheads .sM3blackligmensural 

noteheads .sM2blackmensural 

noteheads .sM1blackmensural 

noteheads .sM3semimensural 

noteheads .sM3semiligmensural 

noteheads .sM2semimensural 

noteheads .sM1semimensural 

noteheads.s0mensural

noteheads.s1mensural

noteheads.s2mensural

noteheads .s0blackmensural

clefs.mensural.c 

clefs.mensural.c_change 

clefs.mensural.f 

clefs.mensural.f_change 

clefs.mensural.g 

clefs.mensural.g_change 

custodes.mensural.u0 

custodes.mensural.u1 

custodes.mensural.u2 

custodes.mensural.d0 

custodes.mensural.d1 

custodes.mensural.d2 

accidentals.mensural1

accidentals.mensuralM1

flags.mensuralu03 

flags.mensuralu13

flags.mensuralu23

flags.mensurald03 

flags.mensurald13

flags.mensurald23

flags.mensuralu04 

flags.mensuralu14

flags.mensuralu24

flags.mensurald04 

flags.mensurald14

flags.mensurald24

flags.mensuralu05 

flags.mensuralu15

flags.mensuralu25

flags.mensurald05 

flags.mensurald15

flags.mensurald25

flags.mensuralu06 

flags.mensuralu16

flags.mensuralu26

flags.mensurald06 

flags.mensurald16

flags.mensurald26

timesig.mensural44 

timesig.mensural22 

timesig.mensural32 

timesig.mensural64 

timesig.mensural94 

timesig.mensural34 

timesig.mensural68 

timesig.mensural98		timesig.mensural48	
timesig.mensural68alt		timesig.mensural24	

Neomensural-Glyphen

rests.M3neomensural		rests.M2neomensural	
rests.M1neomensural		rests.0neomensural	
rests.1neomensural		rests.2neomensural	
rests.3neomensural		rests.4neomensural	
noteheads.slneomensural		noteheads.sM3neomensural	
noteheads.sM2neomensural		noteheads.sM1neomensural	
noteheads.s0neomensural		noteheads.s1neomensural	
noteheads.s2neomensural		clefs.neomensural.c	
clefs.neomensural .c_change		timesig.neomensural44	
timesig.neomensural22		timesig.neomensural32	
timesig.neomensural64		timesig.neomensural94	
timesig.neomensural34		timesig.neomensural68	
timesig.neomensural98		timesig.neomensural48	
timesig.neomensural68alt		timesig.neomensural24	

Petrucci-Glyphen

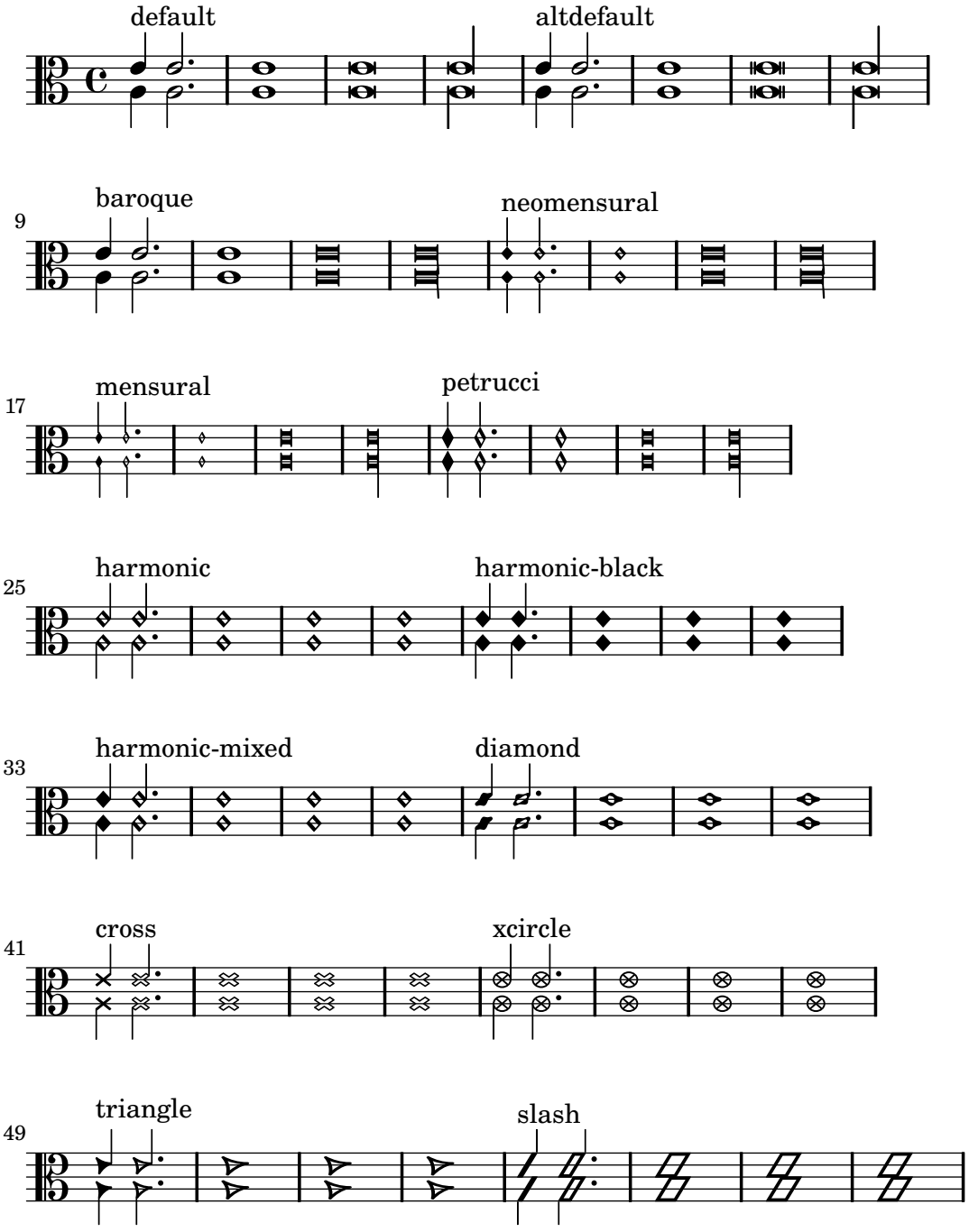
noteheads.s0petrucci		noteheads.s1petrucci	
noteheads.s2petrucci		noteheads .s0blackpetrucci	
noteheads .s1blackpetrucci		noteheads .s2blackpetrucci	
clefs.petrucci.c1		clefs.petrucci.c1_change	
clefs.petrucci.c2		clefs.petrucci.c2_change	
clefs.petrucci.c3		clefs.petrucci.c3_change	
clefs.petrucci.c4		clefs.petrucci.c4_change	
clefs.petrucci.c5		clefs.petrucci.c5_change	
clefs.petrucci.f		clefs.petrucci.f_change	
clefs.petrucci.g		clefs.petrucci.g_change	

Solesmes-Glyphen

noteheads.ssolesmes .incl.parvum		noteheads .ssolesmes.auct.asc	
noteheads .ssolesmes.auct.desc		noteheads.ssolesmes .incl.auctum	
noteheads .ssolesmes.stropha		noteheads.ssolesmes .stropha.aucta	
noteheads .ssolesmes.oriscus			

A.8 Notenkopfstile

Folgende Stile können zur Darstellung der Notenköpfe verwendet werden:



A.9 Textbeschriftungsbefehle

The following commands can all be used inside `\markup { }`.

A.9.1 Font

`\abs-fontsize` *size* (number) *arg* (markup)

Use *size* as the absolute font size to display *arg*. Adjusts `baseline-skip` and `word-space` accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```

default text font size text font size 16 text font size 12

`\bold` *arg* (markup)
Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```

default **bold**

`\box` *arg* (markup)
Draw a box round *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \override #'(box-padding . 0.5)
  \box
  \line { V. S. }
}
```

V. S.

Used properties:

- `box-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\caps` *arg* (markup)
Copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\dynamic` *arg* (markup)
Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like ‚più **f**', the normal words (like ‚più') should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

sfzp

`\finger` *arg* (markup)
Set *arg* as small numbers.

```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```

1 2 3 4 5

`\fontCaps` *arg* (markup)
Set `font-shape` to `caps`

Note: `\fontCaps` requires the installation and selection of fonts which support the `caps` font shape.

`\fontsize` *increment* (number) *arg* (markup)
Add *increment* to the font-size. Adjusts `baseline-skip` accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5
  smaller
}
```

default smaller

Used properties:

- `baseline-skip` (2)
- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)
Set font size to +2.

```
\markup {
  default
  \hspace #2
  \huge
  huge
}
```

default huge

`\italic` *arg* (markup)
Use italic `font-shape` for *arg*.

```
\markup {
  default
  \hspace #2
  \italic
  italic
}
```

default *italic*

`\large` *arg* (markup)
Set font size to +1.

```
\markup {
  default
  \hspace #2
  \large
  large
}
```

default large

`\larger` *arg* (markup)
Increase the font size relative to the current setting.

```
\markup {
  default
  \hspace #2
  \larger
  larger
}
```

default larger

`\magnify` *sz* (number) *arg* (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:

```
A \magnify #1.1 { A } A
```

Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

default 50% larger

`\medium` *arg* (markup)
Switch to medium font-series (in contrast to bold).

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}
```

some bold text medium font series **bold again**

`\normal-size-sub` *arg* (markup)
Set *arg* in subscript with a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default $_{\text{subscript in standard size}}$

Used properties:

- `baseline-skip`

`\normal-size-super` *arg* (markup)
Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

default $^{\text{superscript in standard size}}$

Used properties:

- `baseline-skip`

`\normal-text` *arg* (markup)
Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \caps {
    Some text with font overrides
    \hspace #2
    \normal-text {
      Default text, same font-size
    }
```

```
    \hspace #2
    More text as before
  }
}
```

SOME TEXT WITH FONT OVERRIDES Default text, same font-size **MORE**

`\normalsize` *arg* (markup)
Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

this is very small normal size teeny again

`\number` *arg* (markup)
Set font family to `number`, which yields the font used for time signatures and fingerings. This font contains numbers and some punctuation; it has no letters.

```
\markup {
  \number {
    0 1 2 3 4 5 6 7 8 9 . ,
  }
}
```

0123456789.,

`\roman` *arg* (markup)
Set font family to `roman`.

```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
    \hspace #2
    return to sans
  }
}
```

sans serif, bold text in roman font family return to sans

`\sans` *arg* (markup)
Switch to the sans serif font family.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```

default sans serif

`\simple` *str* (string)
: A simple text string; `\markup { foo }` is equivalent with `\markup { \simple #"foo" }`.

  Note: for creating standard text markup or defining new markup commands, the use of `\simple` is unnecessary.

```
\markup {
  \simple #"simple"
  \simple #"text"
  \simple #"strings"
}
```

simple text strings

`\small` *arg* (markup)
: Set font size to -1.

```
\markup {
  default
  \hspace #2
  \small
  small
}
```

default small

`\smallCaps` *arg* (markup)
: Emit *arg* as small caps.

  Note: `\smallCaps` does not support accented characters.

```
\markup {
  default
  \hspace #2
  \smallCaps {
    Text in small caps
  }
}
```

default Text in small caps

`\smaller` *arg* (markup)
: Decrease the font size relative to the current setting.

```
\markup {
  \fontsize #3.5 {
```

```
    some large text
    \hspace #2
    \smaller {
      a bit smaller
    }
    \hspace #2
    more large text
  }
}
```

some large text a bit smaller more large text

`\sub` *arg* (markup)
Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    O
  }
}
```

H_2O

Used properties:

- `baseline-skip`
- `font-size` (0)

`\super` *arg* (markup)
Set *arg* in superscript.

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

$E = mc^2$

Used properties:

- `baseline-skip`
- `font-size` (0)

`\teeny` *arg* (markup)
Set font size to -3.

```
\markup {
  default
```

```
  \hspace #2
  \teeny
  teeny
}
```

default teeny

`\text` *arg* (markup)
: Use a text font instead of music symbol or music alphabet font.

```
\markup {
  \number {
    1, 2,
    \text {
      three, four,
    }
    5
  }
}
```

1, 2, three, four, **5**

`\tiny` *arg* (markup)
: Set font size to -2.

```
\markup {
  default
  \hspace #2
  \tiny
  tiny
}
```

default tiny

`\typewriter` *arg* (markup)
: Use `font-family` typewriter for *arg*.

```
\markup {
  default
  \hspace #2
  \typewriter
  typewriter
}
```

default `typewriter`

`\underline` *arg* (markup)
: Underline *arg*. Looks at `thickness` to determine line thickness, and `offset` to determine line y-offset.

```
\markup \fill-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
```

```
  \override #'(thickness . 5)
  \underline "underlined"
}
```

underlined underlined underlined

Used properties:

- `offset` (2)
- `thickness` (1)

`\upright` *arg* (markup)
Set `font-shape` to `upright`. This is the opposite of `italic`.

```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
    }
    \hspace #2
    italic again
  }
}
```

italic text upright text *italic again*

A.9.2 Align

`\center-align` *arg* (markup)
Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```

one
two
three

`\center-column` *args* (markup list)
Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\column` *args* (markup list)
: Stack the markups in *args* vertically. The property `baseline-skip` determines the space between markups in *args*.

```
\markup {
  \column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\combine` *arg1* (markup) *arg2* (markup)
: Print two markups on top of each other.

Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; the follow example will not compile:

```
\combine { a list }
```

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}
```



`\concat` *args* (markup list)
: Concatenate *args* in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to `"fi"`.

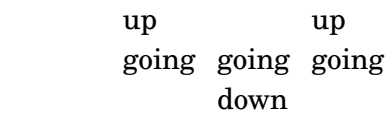
```
\markup {
  \concat {
    one
    two
    three
  }
}
```

onetwothree

`\dir-column` *args* (markup list)

Make a column of *args*, going up or down, depending on the setting of the `direction` layout property.

```
\markup {
  \override #`(direction . ,UP) {
    \dir-column {
      going up
    }
  }
  \hspace #1
  \dir-column {
    going down
  }
  \hspace #1
  \override #'(direction . 1) {
    \dir-column {
      going up
    }
  }
}
```



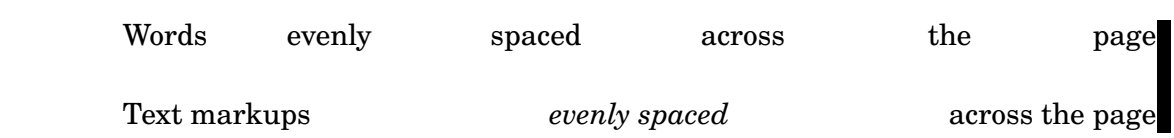
Used properties:

- `baseline-skip`
- `direction`

`\fill-line` *args* (markup list)

Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

```
\markup {
  \column {
    \fill-line {
      Words evenly spaced across the page
    }
    \null
    \fill-line {
      \line { Text markups }
      \line {
        \italic { evenly spaced }
      }
      \line { across the page }
    }
  }
}
```



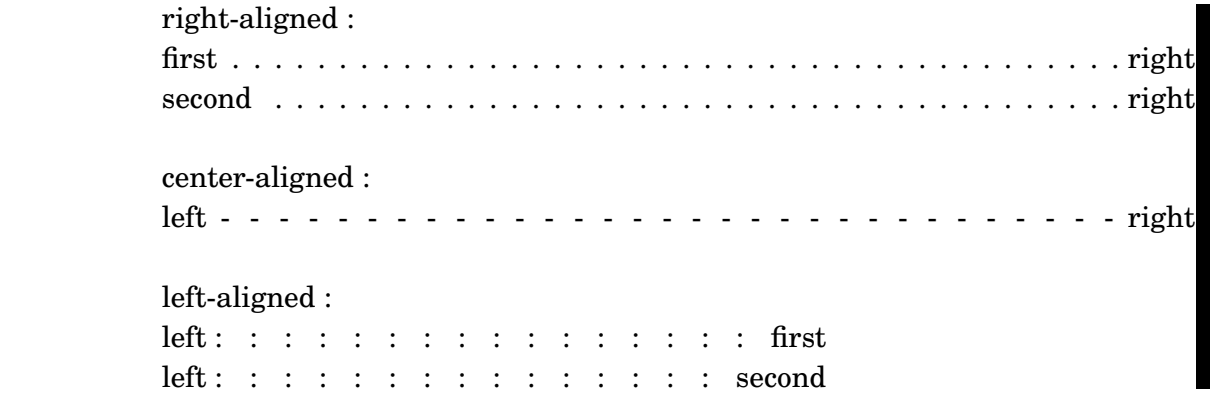
Used properties:

- `line-width` (#f)
- `word-space` (0.6)
- `text-direction` (1)

`\fill-with-pattern` *space* (number) *dir* (direction) *pattern* (markup) *left* (markup) *right* (markup)

Put *left* and *right* in a horizontal line of width `line-width` with a line of markups *pattern* in between. Patterns are spaced apart by *space*. Patterns are aligned to the *dir* markup.

```
\markup \column {
  "right-aligned :"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  \null
  "center-aligned :"
  \fill-with-pattern #1.5 #CENTER - left right
  \null
  "left-aligned :"
  \override #'(line-width . 50) \fill-with-pattern #2 #LEFT : left first
  \override #'(line-width . 50) \fill-with-pattern #2 #LEFT : left second
}
```



Used properties:

- `line-width`
- `word-space`

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)

Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT
    two
    three
    \null
    one
    \general-align #X #CENTER
    two
    three
    \null
```

```
      \line {
        one
        \general-align #Y #UP
        two
        three
      }
      \null
      \line {
        one
        \general-align #Y #3.2
        two
        three
      }
    }
  }
```

one
two
three

one
two
three

one two three

one three
two

`\halign` *dir* (number) *arg* (markup)
: Set horizontal alignment. If *dir* is `-1`, then it is left-aligned, while `+1` is right. Values in between interpolate alignment accordingly.

```
\markup {
  \column {
    one
    \halign #LEFT
    two
    three
    \null
    one
    \halign #CENTER
    two
    three
    \null
    one
    \halign #RIGHT
    two
    three
    \null
    one
    \halign #-5
```

```
    two
    three
  }
}
```

one
two
three
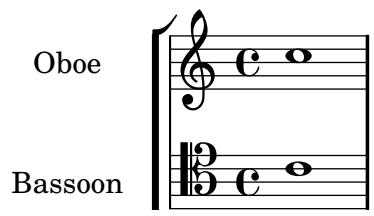
one
two
three

one
two
three

one
two
three

`\hcenter-in` *length* (number) *arg* (markup)
: Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Bassoon
    }
    \clef tenor
    c'1
  }
>>
```



`\hspace` *amount* (number)
: Create an invisible object taking up horizontal space *amount*.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one two three

Used properties:

- `word-space`

`\justify-field` *symbol* (symbol)

Justify the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et dolore magna
    aliqua.  Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\justify` *args* (markup list)

Like `\wordwrap`, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; X is the number of staff spaces.

```
\markup {
  \justify {
```

```
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\justify-string` *arg* (string)

Justify a string. Paragraphs may be separated with double newlines

```
\markup {
  \override #'(line-width . 40)
  \justify-string #"Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut labore
      et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat.


      Excepteur sint occaecat cupidatat non proident, sunt in culpa
      qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`

- `baseline-skip`

`\left-align` *arg* (markup)
Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align
    two
    three
  }
}
```

one
two
three

`\left-column` *args* (markup list)
Put `args` in a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\line` *args* (markup list)
Put *args* in a horizontal line. The property `word-space` determines the space between markups in *args*.

```
\markup {
  \line {
    one two three
  }
}
```

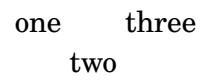
one two three

Used properties:

- `text-direction` (1)
- `word-space`

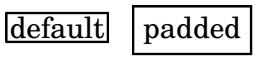
`\lower` *amount* (number) *arg* (markup)
Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.

```
\markup {
  one
  \lower #3
  two
  three
}
```



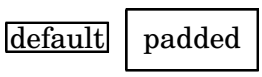
`\pad-around` *amount* (number) *arg* (markup)
Add padding *amount* all around *arg*.

```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```



`\pad-markup` *amount* (number) *arg* (markup)
Add space around a markup object.

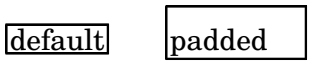
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-markup #1 {
      padded
    }
  }
}
```



`\pad-to-box` *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)
Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
```

```
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```



`\pad-x` *amount* (number) *arg* (markup)
: Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```



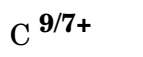
`\put-adjacent` *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)
: Put *arg2* next to *arg1*, without moving *arg1*.

`\raise` *amount* (number) *arg* (markup)
: Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

  The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

  If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

```
\markup {
  C
  \small
  \bold
  \raise #1.0
  9/7+
}
```



`\right-align` *arg* (markup)
: Align *arg* on its right edge.

```
\markup {
  \column {
```

```
    one
    \right-align
    two
    three
  }
}
```

one
two
three

`\right-column` *args* (markup list)
: Put `args` in a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\rotate` *ang* (number) *arg* (markup)
: Rotate object with *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
  \line {
    rotated 45°
  }
}
```

default rotated 45°

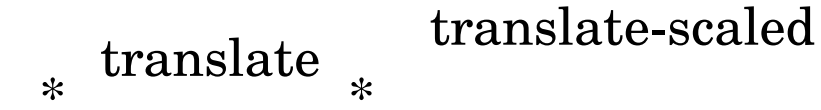
`\translate` *offset* (pair of numbers) *arg* (markup)
: Translate *arg* relative to its surroundings. *offset* is a pair of numbers representing the displacement in the X and Y axis.

```
\markup {
  *
  \translate #'(2 . 3)
  \line { translated two spaces right, three up }
}
```

translated two spaces right, three up
*

`\translate-scaled` *offset* (pair of numbers) *arg* (markup)
Translate *arg* by *offset*, scaling the offset by the `font-size`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```



Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)
Align `arg` to its Y center.

```
\markup {
  one
  \vcenter
  two
  three
}
```

one two three

`\vspace` *amount* (number)
Create an invisible object taking up vertical space of *amount* multiplied by 3.

```
\markup {
    \center-column {
    one
    \vspace #2
    two
    \vspace #5
    three
  }
}
```

one

two

three

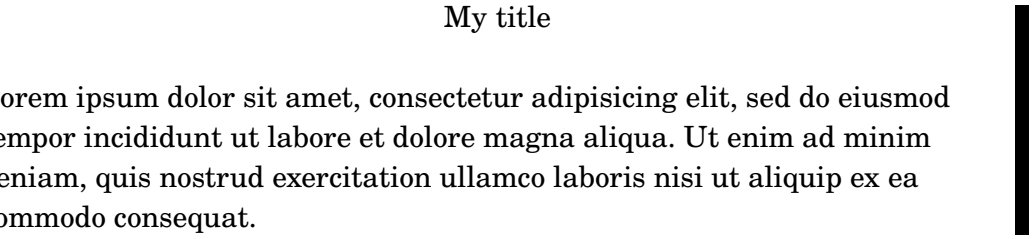
`\wordwrap-field` *symbol* (symbol)

Wordwrap the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et dolore magna
    aliqua.  Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:myText
    }
  }
}

\markup {
  \null
}
```



`\wordwrap` *args* (markup list)

Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string` *arg* (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string #"Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut labore
      et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat.


      Excepteur sint occaecat cupidatat non proident, sunt in culpa
      qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore
et dolore magna aliqua.
Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo
consequat.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia
deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

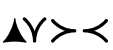
A.9.3 Graphic

`\arrow-head` *axis* (integer) *dir* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

```
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
```

```
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}
```



`\beam` *width* (number) *slope* (number) *thickness* (number)
: Create a beam with the specified parameters.

```
\markup {
  \beam #5 #1 #2
}
```



`\bracket` *arg* (markup)
: Draw vertical brackets around *arg*.

```
\markup {
  \bracket {
    \note #"2." #UP
  }
}
```



`\circle` *arg* (markup)
: Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \circle {
    Hi
  }
}
```



Used properties:

- `circle-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\draw-circle` *radius* (number) *thickness* (number) *filled* (boolean)
: A circle of radius *radius* and thickness *thickness*, optionally filled.

```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
```

```
}
```



`\draw-hline`

Draws a line across a page, where the property `span-factor` controls what fraction of the page is taken up.

```
\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}
```



Used properties:

- `span-factor` (1)
- `line-width`
- `draw-line-markup`

`\draw-line` *dest* (pair of numbers)

A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```



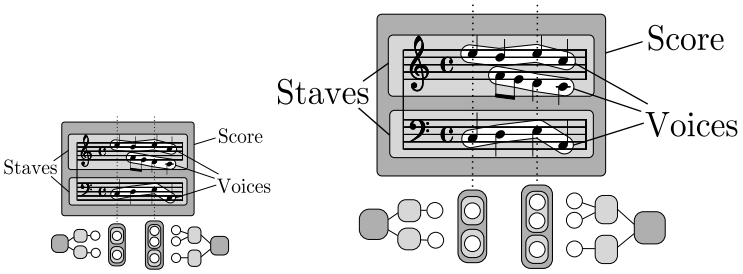
Used properties:

- `thickness` (1)

`\epsfile` *axis* (number) *size* (number) *file-name* (string)

Inline an EPS image. The image is scaled along *axis* to *size*.

```
\markup {
  \general-align #Y #DOWN {
    \epsfile #X #20 #"context-example.eps"
    \epsfile #Y #20 #"context-example.eps"
  }
}
```



`\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)
Draw a box with rounded corners of dimensions *xext* and *yext*. For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).

```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \filled-box #'(1 . 8) #'(0 . 7) #0.2
  \with-color #white
  \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
}
```



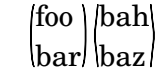
`\hbracket` *arg* (markup)
Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}
```

one two three

`\parenthesize` *arg* (markup)
Draw parentheses around *arg*. This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \line {
    \parenthesize {
      \column {
        foo
        bar
      }
    }
    \override #'(angularity . 2) {
      \parenthesize {
        \column {
          bah
          baz
        }
      }
    }
  }
}
```



Used properties:

- `width` (0.25)
- `thickness` (1)
- `size` (1)
- `padding`
- `angularity` (0)

`\path` *thickness* (number) *commands* (list)

Draws a path with line thickness *thickness* according to the directions given in *commands*. *commands* is a list of lists where the `car` of each sublist is a drawing command and the `cdr` comprises the associated arguments for each command.

Line-cap styles and line-join styles may be customized by overriding the `line-cap-style` and `line-join-style` properties, respectively. Available line-cap styles are `'butt`, `'round`, and `'square`. Available line-join styles are `'miter`, `'round`, and `'bevel`.

The property `filled` specifies whether or not the path is filled with color.

There are seven commands available to use in the list `commands`: `moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `rcurveto`, and `closepath`. Note that the commands that begin with *r* are the relative variants of the other three commands.

The commands `moveto`, `rmoveto`, `lineto`, and `rlineto` take 2 arguments; they are the X and Y coordinates for the destination point.

The commands `curveto` and `rcurveto` create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The `closepath` command takes zero arguments and closes the current subpath in the active path.

Note that a sequence of commands *must* begin with a `moveto` or `rmoveto` to work with the SVG output.

```
samplePath =
  #'((moveto 0 0)
     (lineto -1 1)
     (lineto 1 1)
     (lineto 1 -1)
     (curveto -5 -5 -5 5 -1 0)
     (closepath))

\markup {
  \path #0.25 #samplePath
}
```



Used properties:

- `filled` (#f)
- `line-join-style` (round)
- `line-cap-style` (round)
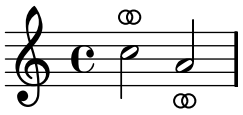
`\postscript` *str* (string)

This inserts *str* directly into the output as a PostScript command string.

```
ringsps = #"
  0.15 setlinewidth
  0.9 0.6 moveto
  0.4 0.6 0.5 0 361 arc
  stroke
  1.0 0.6 0.5 0 361 arc
  stroke
  "

rings = \markup {
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}
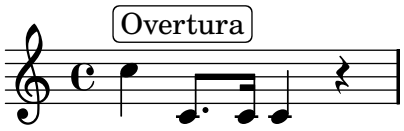
\relative c'' {
  c2^\rings
  a2_\rings
}
```



`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup; the `corner-radius` property makes it possible to define another shape for the corners (default is 1).

```
c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r
```



Used properties:

- `box-padding` (0.5)
- `font-size` (0)
- `corner-radius` (1)
- `thickness` (1)

`\scale` *factor-pair* (pair of numbers) *arg* (markup)

Scale *arg*. *factor-pair* is a pair of numbers representing the scaling-factor in the X and Y axes. Negative values may be used to produce mirror images.

```
\markup {
  \line {
    \scale #'(2 . 1)
    stretched
    \scale #'(1 . -1)
    mirrored
  }
}
```



`\triangle` *filled* (boolean)
A triangle, either filled or empty.

```
\markup {
  \triangle ##t
  \hspace #2
  \triangle ##f
}
```



Used properties:

- `baseline-skip` (2)
- `font-size` (0)
- `thickness` (0.1)

`\with-url` *url* (string) *arg* (markup)
Add a link to URL *url* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-url #"http://lilypond.org/web/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

A.9.4 Music

`\customTabClef` *num-strings* (integer) *staff-space* (number)
Draw a tab clef sans-serif style.

`\doubleflat`
Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```



`\doublesharp`
Draw a double sharp symbol.

```
\markup {
  \doublesharp
}
```


`\flat`

Draw a flat symbol.

```
\markup {
  \flat
}
```



`\musicglyph` *glyph-name* (string)

glyph-name is converted to a musical symbol; for example, `\musicglyph #"accidentals.natural"` selects the natural sign from the music font. See Abschnitt "The Feta font" in *Notationsreferenz* for a complete listing of the possible glyphs.

```
\markup {
  \musicglyph #"f"
  \musicglyph #"rests.2"
  \musicglyph #"clefs.G_change"
}
```



`\natural`

Draw a natural symbol.

```
\markup {
  \natural
}
```



`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)

Construct a note symbol, with stem. By using fractional values for *dir*, longer or shorter stems can be obtained.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```



Used properties:

- `style` ('())

- `font-size` (0)

`\note` *duration* (string) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note #"4." #-0.75` creates a dotted quarter note, with a shortened down stem.

```
\markup {
  \override #'(style . cross) {
    \note #"4.." #UP
  }
  \hspace #2
  \note #"breve" #0
}
```



Used properties:

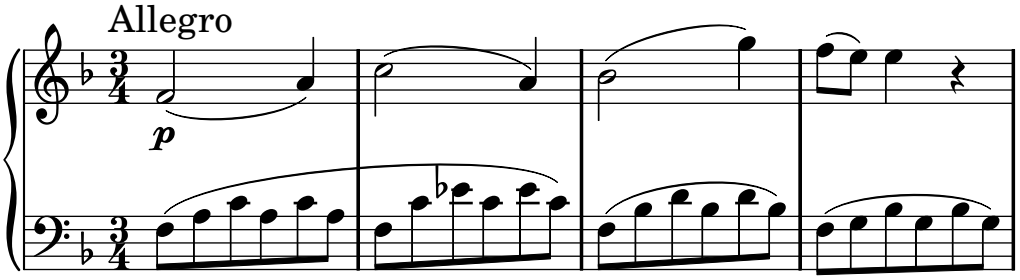
- `style` ('())
- `font-size` (0)

`\score` *score* (score)

Inline an image of music.

```
\markup {
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
        c2( a4)
        bes2( g'4)
        f8( e) e4 r
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
        f c' es c es c)
        f,( bes d bes d bes)
        f( g bes g bes g)
      }
    >>
    \layout {
      indent = 0.0\cm
      \context {
        \Score
        \override RehearsalMark #'break-align-symbols =
          #'(time-signature key-signature)
        \override RehearsalMark #'self-alignment-X = #LEFT
      }
```

```
        \context {
          \Staff
          \override TimeSignature #'break-align-anchor-alignment = #LEFT
        }
      }
    }
  }
```



Used properties:

- `baseline-skip`

`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```


`\semisharp`

Draw a semisharp symbol.

```
\markup {
  \semisharp
}
```



`\sesquiflat`

Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```



`\sesquisharp`

Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```



`\sharp`

Draw a sharp symbol.

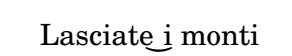
```
\markup {
  \sharp
}
```



`\tied-lyric` *str* (string)

Like simple-markup, but use tie characters for ‚~‘ tilde symbols.

```
\markup {
  \tied-lyric #"Lasciate~i monti"
}
```



A.9.5 Instrument Specific Markup

`\fret-diagram` *definition-string* (string)

Make a (guitar) fret diagram. For example, say

```
\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"
```

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
  - `s:`*number* – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - `t:`*number* – Set the line thickness (in staff spaces). Default: 0.05.
  - `h:`*number* – Set the height of the diagram in frets. Default: 4.
  - `w:`*number* – Set the width of the diagram in strings. Default: 6.
  - `f:`*number* – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
  - `d:`*number* – Set radius of dot, in terms of fret spacing. Default: 0.25.
  - `p:`*number* – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
  - `c:`*string1-string2-fret* – Include a barre mark from *string1* to *string2* on *fret*.
  - *string-fret* – Place a dot on *string* at *fret*. If *fret* is ‘`o`’, *string* is identified as open. If *fret* is ‘`x`’, *string* is identified as muted.
  - *string-fret-fingering* – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.
- Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by '`x`'.
- Open strings are indicated by '`o`'.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a `-`, followed by the finger indicator, e.g. '`3-2`' for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-verbose` *marking-list* (pair)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
  #'((mute 6) (mute 5) (open 4)
     (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute` *`string-number`*`)`

Place a small ‚x‘ at the top of string *string-number*.

`(open` *`string-number`*`)`

Place a small ‚o‘ at the top of string *string-number*.

`(barre` *`start-string end-string fret-number`*`)`

Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(capo` *`fret-number`*`)`

Place a capo indicator (a large solid bar) across the entire fretboard at fret location *fret-number*. Also, set fret *fret-number* to be the lowest fret on the fret diagram.

`(place-fret` *`string-number fret-number`* `[`*`finger-value`*
`[`*`color-modifier`*`]])`

Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*, and an optional color modifier *color-modifier*. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the variable *dot-color*. Setting *color-modifier* to `inverted` inverts the dot color

for a specific fingering. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\harp-pedal` *definition-string* (string)

Make a harp pedal diagram.

Possible elements in *definition-string*:

`^`	pedal is up
`-`	pedal is neutral
`v`	pedal is down
`|`	vertical divider line
`o`	the following pedal should be circled (indicating a change)

The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you'll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (`\override Voice.TextScript #'size = #0.3`) for the overall, the thickness property (`\override Voice.TextScript #'thickness = #3`) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the harp-pedal-details list of properties (`\override Voice.TextScript #'harp-pedal-details #'box-width = #1`). It contains the following settings: `box-offset` (vertical shift of the box center for up/down pedals), `box-width`, `box-height`, `space-before-divider` (the spacing between two boxes before the divider) and `space-after-divider` (box spacing after the divider).

```
\markup \harp-pedal #"^-v|--ov^"
```



Used properties:

- `thickness` (0.5)
- `harp-pedal-details` ('())
- `size` (1.2)

`\woodwind-diagram` *instrument* (symbol) *user-draw-commands* (list)

Make a woodwind-instrument diagram. For example, say

```
\markup \woodwind-diagram #'oboe #'((lh . (d ees)) (cc . (five3qT1q)) (rh . (gis))
```

for an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed while the five-hole of the central column effectuates a trill between 1/4 and 3/4 closed.

The following instruments are supported:

- piccolo
- flute
- oboe
- clarinet
- bass-clarinet
- saxophone
- bassoon
- contrabassoon

To see all of the callable keys for a given instrument, include the function `(print-keys 'instrument)` in your .ly file, where instrument is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

- 1q (1/4 covered)
- 1h (1/2 covered)
- 3q (3/4 covered)
- R (ring depressed)
- F (fully covered; the default if no state put)

Additionally, these configurations can be used in trills. So, for example, `three3qTR` effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, `threeRT` effectuates a trill between R and open, whereas `threeTR` effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke `(print-keys-verbose 'instrument)`.

Lastly, substituting an empty list for the pressed-key alist will result in a diagram with all of the keys drawn but none filled, for example:

```
\markup \woodwind-diagram #'oboe #'()
```

Used properties:

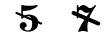
- `graphical` (#t)
- `thickness` (0.1)
- `size` (1)

A.9.6 Other

`\backslashed-digit` *num* (integer)

A feta number, with backslash. This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```



Used properties:

- `thickness` (1.6)

- `font-size` (0)

`\char` *num* (integer)
Produce a single character. Characters encoded in hexadecimal format require the prefix `#x`.

```
\markup {
  \char #65 \char ##x00a9
}
```

A ©

`\eyeglasses`
Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```



`\footnote` *mkup* (markup) *note* (markup)
Have footnote *note* act as an annotation to the markup *mkup*.

`\fraction` *arg1* (markup) *arg2* (markup)
Make a fraction of two markups.

```
\markup {

  \fraction 355 113
}
```

$$\pi \approx \frac{355}{113}$$

Used properties:

- `font-size` (0)

`\fromproperty` *symbol* (symbol)
Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

```
\header {
  myTitle = "myTitle"
  title = \markup {
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
  \null
}
```

from *myTitle*

`\left-brace` *size* (number)
A feta brace in point size *size*.

```
\markup {
  \left-brace #35
  \hspace #2
  \left-brace #45
}
```



`\lookup` *glyph-name* (string)

Lookup a glyph by name.

```
\markup {
  \override #'(font-encoding . fetaBraces) {
    \lookup #"brace200"
    \hspace #2
    \rotate #180
    \lookup #"brace180"
  }
}
```



`\markalphabet` *num* (integer)

Make a markup letter for *num*. The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

I AA

`\markletter` *num* (integer)

Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

J AB

`\null`

An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (symbol) *arg* (markup)
: Apply the *procedure* markup command to *arg*. *procedure* should take a single argument.

`\override` *new-prop* (pair) *arg* (markup)
: Add the argument *new-prop* to the property list. Properties may be any property supported by Abschnitt "font-interface" in *Referenz der Interna*, Abschnitt "text-interface" in *Referenz der Interna* and Abschnitt "instrument-specific-markup-interface" in *Referenz der Interna*.

```
\markup {
  \line {
    \column {
      default
      baseline-skip
    }
    \hspace #2
    \override #'(baseline-skip . 4) {
      \column {
        increased
        baseline-skip
      }
    }
  }
}
```

default
baseline-skip

increased
baseline-skip

`\page-link` *page-number* (number) *arg* (markup)
: Add a link to the page *page-number* around *arg*. This only works in the PDF backend.

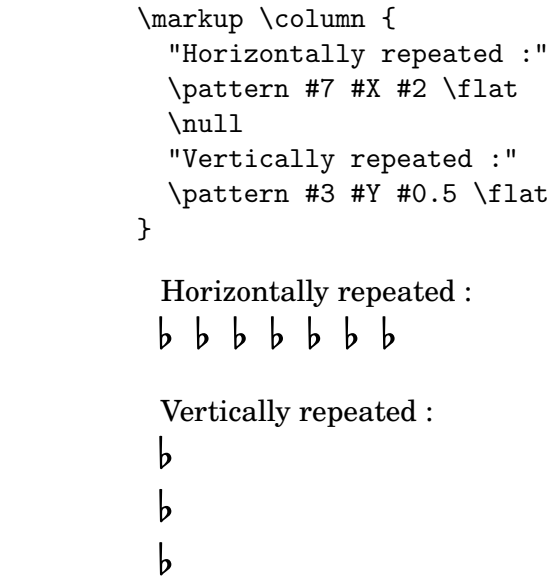
```
\markup {
  \page-link #2  { \italic { This links to page 2... } }
}
```

This links to page 2...

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)
: Reference to a page number. *label* is the label set on the referenced page (using the `\label` command), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

`\pattern` *count* (integer) *axis* (integer) *space* (number) *pattern* (markup)
: Prints *count* times a *pattern* markup. Patterns are spaced apart by *space*. Patterns are distributed on *axis*.

```
\markup \column {
  "Horizontally repeated :"
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated :"
  \pattern #3 #Y #0.5 \flat
}
```



`\right-brace` *size* (number)

A feta brace in point size *size*, rotated 180 degrees.

```
\markup {
  \right-brace #45
  \hspace #2
  \right-brace #35
}
```



`\slashed-digit` *num* (integer)

A feta number, with slash. This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \slashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\stencil` *stil* (stencil)

Use a stencil as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```



`\strut`

Create a box of the same height as the space in the current font.

`\transparent` *arg* (markup)

Make *arg* transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

`\verbatim-file` *name* (string)

Read the contents of file *name*, and include it verbatim.

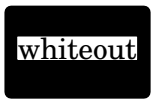
```
\markup {
  \verbatim-file #"simple.ly"
}
```

```
%% A simple piece in LilyPond, a scale.
\relative c' {
  c d e f g a b c
}
%% Optional helper for automatic updating by convert-ly.  May be omit
\version "2.14.0"
```

`\whiteout` *arg* (markup)

Provide a white background for *arg*.

```
\markup {
  \combine
    \filled-box #'(-1 . 10) #'(-3 . 4) #1
    \whiteout whiteout
}
```



`\with-color` *color* (color) *arg* (markup)

Draw *arg* in color specified by *color*.

```
\markup {
  \with-color #red
  red
  \hspace #2
  \with-color #green
  green
  \hspace #2
  \with-color #blue
  blue
}
```

red green blue

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
: Set the dimensions of *arg* to *x* and *y*.

`\with-link` *label* (symbol) *arg* (markup)
: Add a link to the page holding label *label* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-link #"label" { \italic { This links to the page containing the label... }
}
```

This links to the page containing the label...

A.10 Textbeschriftungslistenbefehle

Folgende Befehle können mit dem Befehl `\markuplines` zusammen benutzt werden:

`\column-lines` *args* (markup list)
: Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.

  Used properties:

  - `baseline-skip`

`\justified-lines` *args* (markup list)
: Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\override-lines` *new-prop* (pair) *args* (markup list)
: Like `\override`, for markup lists.

`\table-of-contents`

`\wordwrap-internal` *justify* (boolean) *args* (markup list)
: Internal markup list command used to define `\justify` and `\wordwrap`.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)

`\wordwrap-lines` *args* (markup list)
: Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\wordwrap-string-internal` *justify* (boolean) *arg* (string)
Internal markup list command used to define `\justify-string` and `\wordwrap-string`.

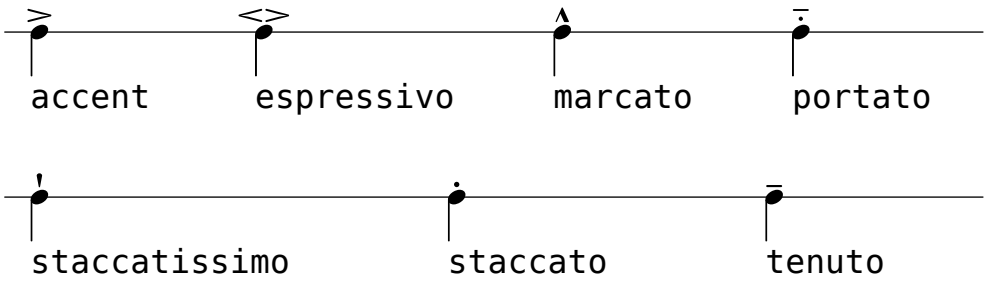
Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`

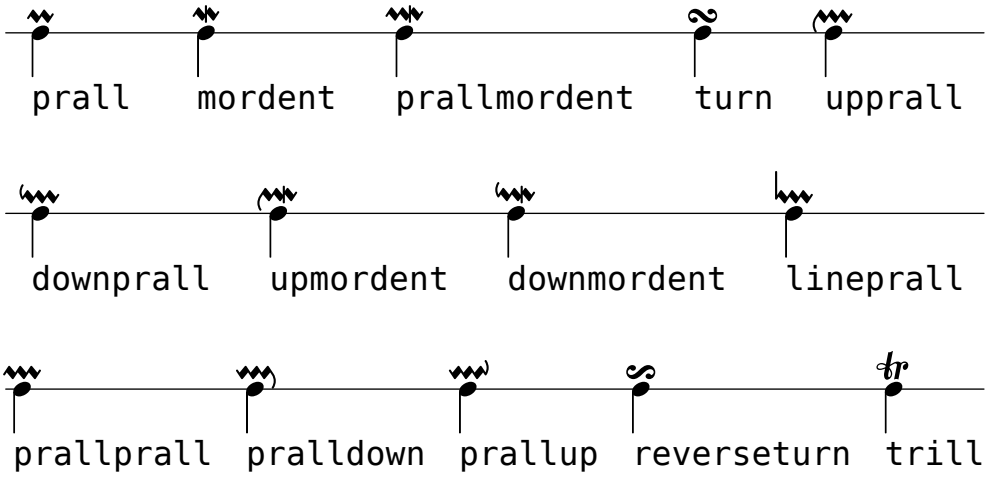
A.11 Liste der Artikulationszeichen

Die Skripte unten sind in der Feta-Schriftart definiert und können an Noten angehängt werden (etwa '`c\accent`').

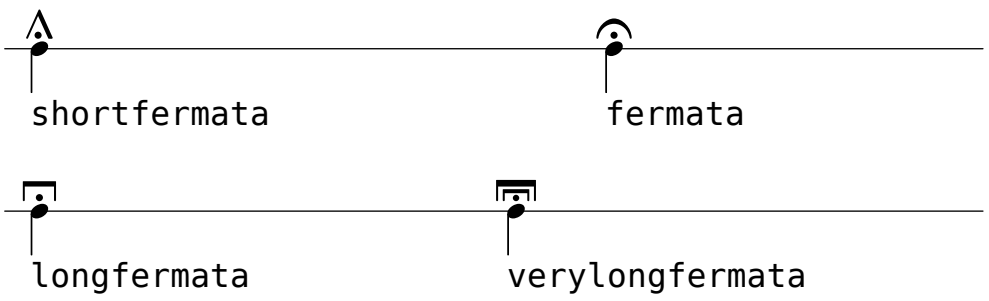
Artikulationsskripte



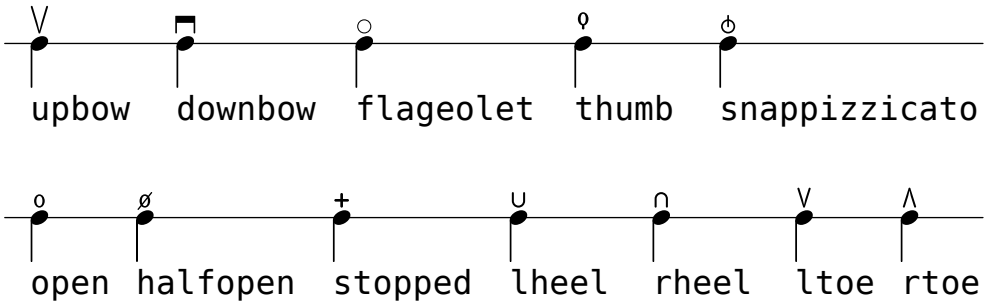
Ornamentale Skripte



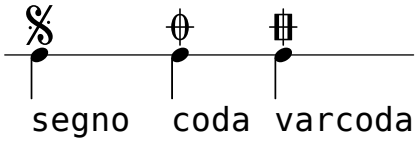
Fermatenskripte



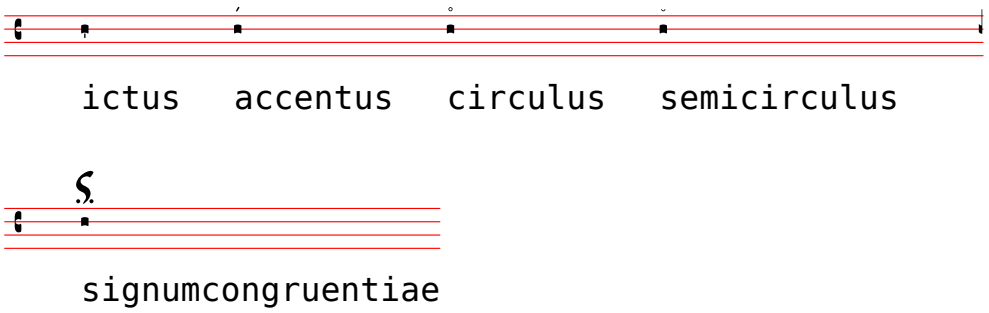
Instrumentenspezifische Skripte



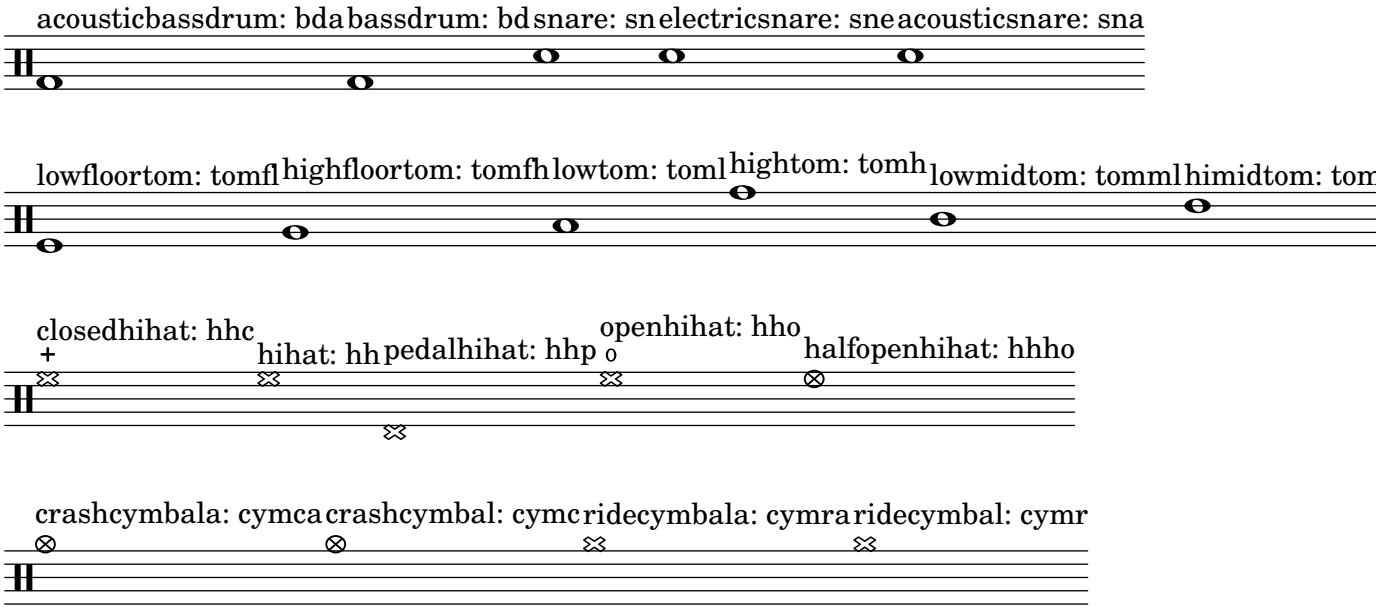
Wiederholungszeichensrkipte

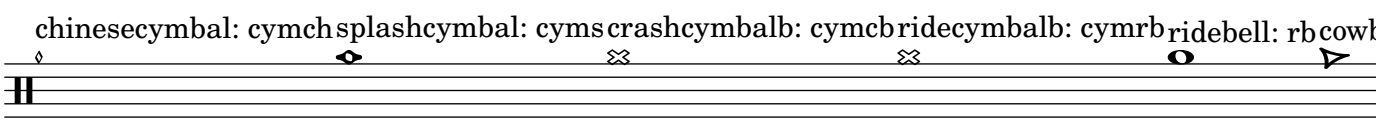


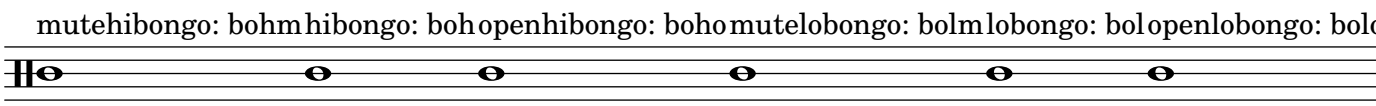
Ancient scripts

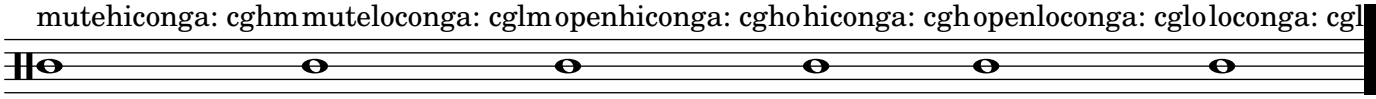


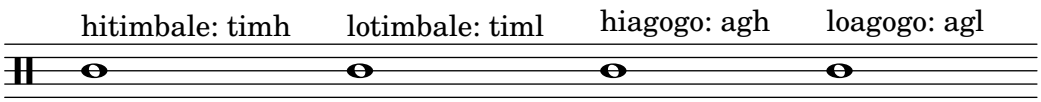
A.12 Schlagzeugnoten

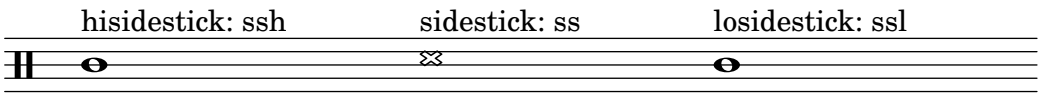


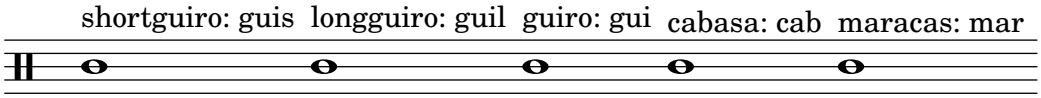

chinesecymbal: cymch
splashcymbal: cyms
crashcymbalb: cymcb
ridecymbalb: cymrb
ridebell: rb
cowb

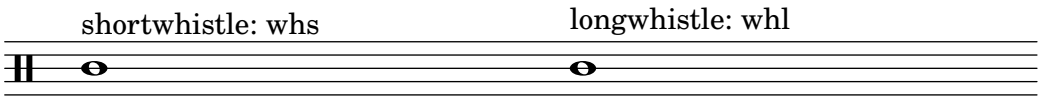

mutehibongo: bohm
hibongo: boh
openhibongo: boho
mutelobongo: bolm
lobongo: bol
openlobongo: bolo

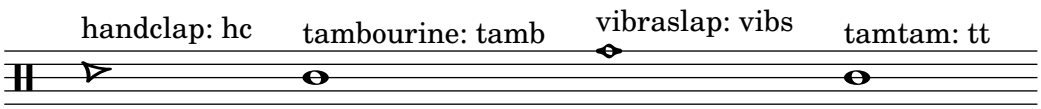

mutehiconga: cghm
muteloconga: cglm
openhiconga: cgho
hiconga: cgh
openloconga: cglo
loconga: cgl

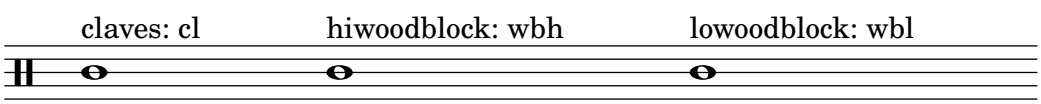

hitimbale: timh
lotimbale: timl
hiagogo: agh
loagogo: agl

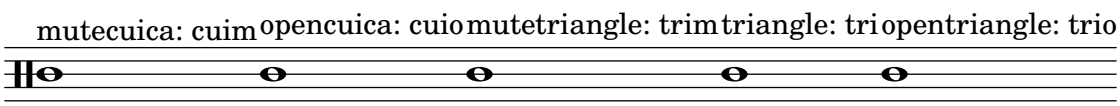

hisidestick: ssh
sidestick: ss
losidestick: ssl

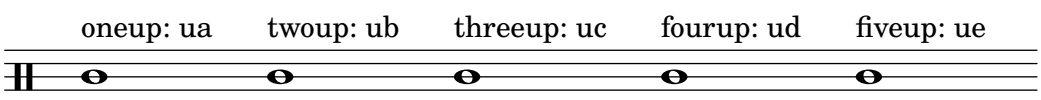

shortguiro: guis
longguiro: guil
guiro: gui
cabasa: cab
maracas: mar

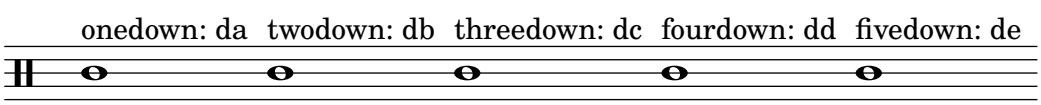

shortwhistle: whs
longwhistle: whl


handclap: hc
tambourine: tamb
vibraslap: vibs
tamtam: tt


claves: cl
hiwoodblock: wbh
lowoodblock: wbl


mutecuica: cuim
opencuica: cuio
mutetriangle: trim
triangle: tri
opentriangle: trio


oneup: ua
twoup: ub
threeup: uc
fourup: ud
fiveup: ue


onedown: da
twodown: db
threedown: dc
fourdown: dd
fivedown: de

A.13 Technisches Glossar

Ein Glossar der technischen Ausdrücke und Konzepte, die von LilyPond intern benutzt werden. Die Ausdrücke kommen in den Handbüchern, auf den Mailinglisten oder im Quellcode vor.

alist

Eine assoziative Liste oder **alist** in kurz ist ein Scheme-Paar, das einen Wert mit einem Schlüssel assoziiert: `(Schlüssel . Wert)`. In der Datei '`scm/lily.scm`' beispielsweise assoziiert die alist „type-p-name-alist“ bestimmte Prädikate (etwa `ly:music?`) mit Bezeichnungen (wie „music“) sodass Fehler der Typüberprüfung über eine Konsolennachricht mitgeteilt werden können, die auch die Bezeichnung des erwarteten Typprädikats mitteilt.

callback

Ein **callback** ist eine Routine, Funktion oder Methode, deren Referenz in einem Aufruf als Argument an eine andere Routine weitergereicht wird, sodass die aufgerufene Routine ermöglicht wird, das Argument zu aktivieren. Die Technik ermöglicht es einer niedrigeren Ebene des Programmes, eine Funktion aufzurufen, die auf höherer Ebene definiert wurde. Callbacks werden sehr ausgiebig in LilyPond eingesetzt, um es Scheme-Code auf der Benutzerebene zu erlauben, wie viele Funktionen der niedrigeren Ebene ausgeführt werden sollen.

closure

In Scheme entsteht ein **closure** (Abschluss), wenn eine Funktion, normalerweise ein Lambda-Ausdruck, als Variable weitergegeben wird. Das closure enthält den Code der Funktion plus Verweise zu den lexikalischen Verknüpfungen der freien Variablen der Funktion (also die Variablen, die in Ausdrücken benutzt werden, aber außerhalb von ihnen definiert werden). Wenn diese Funktion später einem anderen Argument zugewiesen wird, werden die freien Variabel-Verknüpfungend, die in das closure eingeschlossen sind, benutzt um die Werte der freien Variablen, die in der Rechnung benutzt werden sollen, zu errechnen. Eine nützliche Eigenschaft von closures ist, dass man interne variable Werte zwischen den Aufrufen wiederverwerten kann, sodass ein Status erhalten bleiben kann.

Ein **simple closure** (einfacher Abschluss) ist ein closure, dessen Ausdruck keine freien Variablen und auch keine freien Variablel-Verknüpfungen hat.

Ein simple closure wird in LilyPond von einem smob dargestellt, der den Ausdruck und eine Methode, wie der Ausdruck auf eine Liste von Argumenten angewendet werden soll, enthält.

glyph

Ein **glyph** (Glyphe) ist eine bestimmte graphische Repräsentation eines typographischen Charakters oder einer Kombination von zwei oder mehr Charakteren, die dann eine Ligatur bilden. Eine Gruppe an Glyphen des gleichen Stils bilden ein Font, und eine Gruppe an Fonts, die mehrere Stile darstellen, bilden eine Schriftfamilie (engl. typeface).

Siehe auch

Notationsreferenz: ⟨undefined⟩ [Fonts], Seite ⟨undefined⟩, ⟨undefined⟩ [Text encoding], Seite ⟨undefined⟩.

grob

LilyPond-Objekte, die Elemente der Notation in der graphischen Ausgabe des Programmen darstellen, wie etwa Notenköpfe, Hälse, Bögen, Bindebögen, Fingersatz, Schlüssel usw., werden „Layout-Objekte“ genannt, auch oft als „GRaphische OBjekte“ bezeichnet, was dann zu **grob** abgekürzt wird.

Siehe auch

Handbuch zum Lernen: Abschnitt “Objects and interfaces” in *Handbuch zum Lernen*, Abschnitt “Naming conventions of objects and properties” in *Handbuch zum Lernen*, Abschnitt “Properties of layout objects” in *Handbuch zum Lernen*.

Referenz der Interna: Abschnitt “All layout objects” in *Referenz der Interna*.

immutable

Ein **immutable** (unberührbares) Objekt ist ein, dessen Status nach der Erstellung nicht mehr verändert werden kann, entgegen einem mutable Objekt, das nach der Erstellung noch verändert werden kann.

In LilyPond sind unberührbare oder geteilte Eigenschaften das Standardverhalten von Grobs. Sie werden zwischen vielen Objekten geteilt. Entgegen ihrer Bezeichnung können sie jedoch mit `\override` und `\revert` verändert werden.

Siehe auch

Notationsreferenz: [mutable], Seite 644.

interface

Aktionen und Eigenschaften, die eine Gruppe von Grobs gemeinsam haben, werden in ein Objekt gesammelt, das als `grob-interface` oder auch „Schnittstelle“ (engl. interface) bezeichnet wird.

Siehe auch

Handbuch zum Lernen: Abschnitt “Objekte und Schnittstellen” in *Handbuch zum Lernen*, Abschnitt “Regeln zur Benennung von Objekten und Eigenschaften” in *Handbuch zum Lernen*, Abschnitt “Eigenschaften, die Schnittstellen besitzen können” in *Handbuch zum Lernen*.

Notationsreferenz: Abschnitt 5.2.2 [Layout-Schnittstellen], Seite 526.

Referenz der Interna: Abschnitt “Graphical Object Interfaces” in *Referenz der Interna*.

lexer

Ein **lexer** ist ein Programm, das eine Charaktersequenz in eines Sequenz von Tokens übersetzt. Dieser Prozess wird als lexikalische Analyse bezeichnet. Der LilyPond-Lexer konvertiert eine Eingabedatei (‘`.ly`’ in eine Datei mit Tokens, die sich besser für den nächsten Schritt der Verarbeitung, nämlich das Parsen, eignet. Siehe [parser], Seite 645.

mutable

Ein **mutable** (veränderbares) Objekt ist ein Objekt, dessen Status verändert werden kann, im Gegenteil zu einem immutable Objekt, dessen Status zur Entstehungszeit festgelegt ist.

In LilyPond enthalten mutable Eigenschaften Werte, die nur für einen Grob gelten. Normalerweise werden Listen von anderen Objekten oder Resultate einer Berechnung in mutablen Eigenschaften gespeichert.

Siehe auch

Notationsreferenz: [immutable], Seite 644.

output-def

Eine Instanz der `Output-def`-Klasse enthält die Methoden und Datenstruktur, die mit einem Ausgabeabschnitt assoziiert wird. Instanzen werden für `midi`, `layout` und `paper`-Umgebungen erstellt.

parser

Ein **parser** (Syntaxanalysierer) analysiert die Tokensequenzen, die von einem Lexer erstellt wurden, um deren grammatikalische Struktur zu entschlüsseln, wie sie von den Regeln des Eingabeformates vorgegeben werden. Dabei werden die Sequenzen in immer größere Gruppen entsprechend den grammatischen Regeln zusammengefasst. Wenn die Kette der Tokens gültig ist, ist das Endprodukt ein Token-Baum, dessen Wurzel das Startsymbol der Grammatik ist. Wenn dies nicht erreicht werden kann, ist die Datei nicht korrekt und ensprechende Fehlermeldungen werden ausgegeben. Die syntaktischen Gruppierungen und die Regeln, nach welchen die Gruppen aus ihren Einzelteilen nach der LilyPond-Syntax erstellt werden, finden sich in der Datei '`lily/parser.yy`' und werden in der Backus Normal Form (BNF) in Anhang C [LilyPond-Grammatik], Seite 705 gezeigt. Diese Datei wird benutzt, um den Parser während der Programmkompilation zu erstellen. Hierzu wird der Parser-Ersteller Bison verwendet. Er ist Teil des Quellcodes und nicht in die binäre Installation von LilyPond integriert.

parser variable

Diese Variablen werden direkt in Scheme definiert. Von ihrer direkten Benutzung durch den Benutzer wird streng abgeraten, weil ihre Semantikzuordnung sehr verwirrend sein kann.

Wenn der Wert einer derartigen Variable in einer '`.ly`'-Datei verändert wird, ist diese Änderung global, und wenn sie nicht explizit rückgängig gemacht wird, wird der neue Wert bis zum Ende der Datei gelten und dabei sowohl aufeinander folgende `\score`-Umgebungen als auch externe Dateien, die mit `\include` geladen werden, beeinflussen. Das kann zu nicht gewollten Konsequenzen führen, und in komplizierteren Projekten kann es sehr schwer sein, die immer wieder auftretenden Fehler zu beheben.

LilyPond benutzt folgende Parser-Variablen:

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- parseStringResult
- partCombineListener
- pitchnames
- toplevel-bookparts
- toplevel-scores
- showLastLength
- showFirstLength

prob

Property OBjects, also Eigenschaftsobjekte, oder kurz **Prob**, sind Mitglieder der `Prob`-Klasse, eine einfache Basisklasse für Objekte, die mutable oder immutable alists haben und die Methoden, mit denen sie verändert werden können. Die `Music`- und `Stream_event`-Klassen stammen von `Prob` ab. Verkörperungen der `Prob`-Klasse werden auch erstellt, um formatierte Inhalte von Systemgrobs und Titelblöcken während der Erstellung des Seitenlayouts zu speichern.

simple closure

Siehe [closure], Seite 643.

smob

Smobs sind ScheMe-OBjekte, Teile des Mechanismus von Guile, um C- und C++-Ojekte in Scheme-Code zu exportieren. In LilyPond werden Smobs von C++-Objekten mithilfe von Makros erstellt. Es gibt zwei Arten von Smob-Objekten: einfache Smobs, die da sind für einfach immutable Objekte wie Nummern, und komplexe Smobs, benutzt für Objekte mit einer Identität. Wenn Sie auf die LilyPond-Quellen zurückgreifen können, findet sich mehr Information hierzu in '`lily/includes/smob.hh`'.

stencil

Eine Einheit der **stencil**-Klasse enthält die Information, die benötigt wird um ein typographisches Objekt zu setzen. Es handelt sich um einen sehr einfachen Smob, der eine begrenzende Box enthält, welche die vertikale und horizontale Ausdehnung des Objekt beschreibt, und einen Scheme-Ausdruck, der das Objekt ausgibt, nachdem es ausgewertet wurde. Stencil-Objekte können kombiniert werden, um komplexere Stencil zu bilden, die aus einem Baum von Scheme-Ausdrücken des Typs Stencil bestehen.

Die `stencil`-Eigenschaft, die einen Grob mit seinem Stencil verbindet, ist in der `grob-interface`-Schnittstelle definiert.

Siehe auch

Referenz der Interna: Abschnitt "grob-interface" in *Referenz der Interna*.

A.14 Alle Kontexteigenschaften

`aDueText` (markup)
: Text to print at a unisono passage.

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBassFigureAccidentals` (boolean)
: If true, then the accidentals are aligned in bass figure context.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`associatedVoice` (string)
: Name of the `Voice` that has the melody for this `Lyrics` line.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Abschnitt "Score" in *Referenz der Interna* then all staves share accidentals, and if *context* is Abschnitt "Staff" in *Referenz der Interna* then all voices in the same staff share accidentals, but staves do not.

  procedure The procedure represents an accidental rule to be applied to the previously specified context.

  The procedure takes the following arguments:

  `context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos`
The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoBeamCheck` (procedure)
A procedure taking three arguments, *context*, *dir* [start/stop (-1 or 1)], and *test* [shortest note in the beam]. A non-`#f` return value starts or stops the auto beam.

`autoBeaming` (boolean)
If set to true then beams are generated automatically.

`autoCautionaries` (list)
List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`automaticBars` (boolean)
If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

`barAlways` (boolean)
If set to true a bar line is drawn after each note.

`barCheckSynchronize` (boolean)
If true then reset `measurePosition` when finding a bar check.

`barNumberVisibility` (procedure)
A Procedure that takes an integer and returns whether the corresponding bar number should be printed.

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`bassFigureFormatFunction` (procedure)
A procedure that is called to produce the formatting for a `BassFigure` grob. It takes a list of `BassFigureEvent`s, a context, and the grob to format.

`bassStaffProperties` (list)
An alist of property settings to apply for the down staff of `PianoStaff`. Used by `\autochange`.

`beamExceptions` (list)
An alist of exceptions to autobeam rules that normally end on beats.

`beatStructure` (list)
List of `baseMoment`s that are combined to make beats.

`chordChanges` (boolean)
Only show changes in chords scheme?

`chordNameExceptions` (list)
An alist of chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameExceptionsFull` (list)
: An alist of full chord exceptions. Contains (*`chord . markup`*) entries.

`chordNameExceptionsPartial` (list)
: An alist of partial chord exceptions. Contains (*`chord . (prefix-markup suffix-markup)`*) entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNameLowercaseMinor` (boolean)
: Downcase roots of minor chords?

`chordNameSeparator` (markup)
: The markup object used to separate parts of a chord name.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordPrefixSpacer` (number)
: The space added between the root symbol and the prefix of a chord name.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefOctavation` (integer)
: Add this much extra octavation. Values of 7 and -7 are common.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`completionBusy` (boolean)
: Whether a completion-note head is playing.

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefOctavation` (integer)
: Add this much extra octavation. Values of 7 and -7 are common.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

`defaultBarType` (string)
: Set the default type of bar line. See `whichBar` for information on available bar types.

  This variable is read by Abschnitt "Timing_translator" in *Referenz der Interna* at Abschnitt "Score" in *Referenz der Interna* level.

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`drumPitchTable` (hash table)
: A table mapping percussion instruments (symbols) to pitches.

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`', '`timbales-style`', '`congas-style`', '`bongos-style`', and '`percussion-style`'.

  The layout style is a hash table, containing the drum-pitches (e.g., the symbol '`hihat`') as keys, and a list `(`*`notehead-style script vertical-position`*`)` as values.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extendersOverRests` (boolean)
: Whether to continue extenders as they cross a rest.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals changing from a non-natural to another non-natural.

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`figuredBassPlusDirection` (direction)
: Where to put plus signs relative to the main figure.

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`firstClef` (boolean)
: If true, create a new clef when starting a staff.

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

`fontSize` (number)
: The relative size of all grobs in a context.

`forbidBreak` (boolean)
: If set to `##t`, prevent a line break at this point.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

`handleNegativeFrets` (symbol)
: How the automatic fret calculator should handle calculated negative frets. Values include `'ignore`, to leave them out of the diagram completely, `'include`, to include them as calculated, and `'recalculate`, to ignore the specified string and find a string where they will fit with a positive fret number.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`ignoreBarChecks` (boolean)
: Ignore bar checks.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`ignoreMelismata` (boolean)
: Ignore melismata for this Abschnitt "Lyrics" in *Referenz der Interna* line.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`implicitTimeSignatureVisibility` (vector)
: break visibility for the default time signature.

`includeGraceNotes` (boolean)
: Do not ignore grace notes for Abschnitt "Lyrics" in *Referenz der Interna*.

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

`instrumentEqualizer` (procedure)
: A function taking a string (instrument name), and returning a `(min . max)` pair of numbers for the loudness range of the instrument.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`instrumentTransposition` (pitch)
: Define the transposition of the instrument. Its value is the pitch that sounds like middle C. This is used to transpose the MIDI output, and `\quotes`.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(step . alter)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keySignature` (list)
: The current key signature. This is an alist containing `(step . alter)` or `((octave . step) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keySignature = #`((6 . ,FLAT))`.

`lyricMelismaAlignment` (direction)
: Alignment to use for a melisma syllable.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`markFormatter` (procedure)
: A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`maximumFretStretch` (number)
: Don't allocate frets further than this from specified frets.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`melismaBusyProperties` (list)
: A list of properties (symbols) to determine whether a melisma is playing. Setting this property will influence how lyrics are aligned to notes. For example, if set to `#'(melismaBusy beamMelismaBusy)`, only manual melismata and manual beams are considered. Possible values include `melismaBusy`, `slurMelismaBusy`, `tieMelismaBusy`, and `beamMelismaBusy`.

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`midiChannelMapping` (symbol)
: How to map MIDI channels: per `instrument` (default), `staff` or `voice`.

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`minimumPageTurnLength` (moment)
: Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)
: Minimum length of a repeated section for a page turn to be allowed within that section.

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

`output` (music output)
: The output produced by a score-level translator during music interpretation.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

`predefinedDiagramTable` (hash table)
: The hash table of predefined fret diagrams to use in FretBoards.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

`printOctaveNames` (boolean)
: Print octave marks for the `NoteNames` context.

`printPartCombineTexts` (boolean)
: Set ‚Solo‘ and ‚A due‘ texts in the part combiner?

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

`rehearsalMark` (integer)
: The last rehearsal mark printed.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta` *x*`)`, where *x* is a string or `#f`. `'end-repeat` is also accepted as a command.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

`restCompletionBusy` (boolean)
: Signal whether a completion-rest is active.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`searchForVoice` (boolean)
: Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

`shapeNoteStyles` (vector)
: Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`skipBars` (boolean)
: If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

```
{
  r1 r1*3 R1*3
  \set Score.skipBars= ##t
  r1*3 R1*3
}
```

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`soloIIText` (markup)
: The text for the start of a solo for voice ‚two‘ when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

`squashedPosition` (integer)
: Vertical position of squashing for Abschnitt “Pitch_squash_engraver” in *Referenz der Interna*.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

`stanza` (markup)
: Stanza ‚number‘ to print before the start of a verse. Use in `Lyrics` context.

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

`suggestAccidentals` (boolean)
: If set, accidentals are typeset as cautionary suggestions over the note.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

`tempoWholesPerMinute` (moment)
: The tempo in whole notes per minute.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

`timeSignatureFraction` (pair of numbers)
: A pair of numbers, signifying the time signature. For example, `#'(4 . 4)` is a 4/4 time signature.

`timeSignatureSettings` (list)
: A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for `baseMoment`, `beatStructure`, and `beamExceptions`.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

`tonic` (pitch)
: The tonic of the current scale.

`topLevelAlignment` (boolean)
: If true, the *Vertical_align_engraver* will create a *VerticalAlignment*; otherwise, it will create a *StaffGrouper*

`trebleStaffProperties` (list)
: An alist of property settings to apply for the up staff of `PianoStaff`. Used by `\autochange`.

`tremoloFlags` (integer)
: The number of tremolo flags to add if no number is specified.

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

`tupletSpannerDuration` (moment)
: Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. By setting this property, you can make brackets last shorter.

```
{
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

`vocalName` (markup)
: Name of a vocal line.

`voltaSpannerDuration` (moment)
: This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  `\set Staff.whichBar = "|:"`

  This will create a start-repeat bar in this staff only. Valid values are described in Abschnitt “bar-line-interface” in *Referenz der Interna*.

A.15 Eigenschaften des Layouts

`add-stem-support` (boolean)
: If set, the `Stem` object is included in this script's support.

`after-line-breaking` (boolean)
: Dummy property, used to trigger callback for `after-line-breaking`.

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`allow-loose-spacing` (boolean)
: If set, column can be detached from main spacing.

`allow-span-bar` (boolean)
: If false, no inter-staff bar line will be created below this bar line.

`alteration` (number)
: Alteration numbers for accidental.

`alteration-alist` (list)
: List of `(`*`pitch . accidental`*`)` pairs for key signature.

`annotation` (string)
: Annotate a grob for debug purposes.

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean)
: Print the line from an annotation to the grob that it annotates.

`arpeggio-direction` (direction)
: If set, put an arrow on the arpeggio squiggly line.

`arrow-length` (number)
: Arrow length.

`arrow-width` (number)
: Arrow width.

`auto-knee-gap` (dimension, in staff space)
: If a gap is found between note heads where a horizontal beam fits that is larger than this number, make a kneed beam.

`average-spacing-wishes` (boolean)
: If set, the spacing wishes are averaged over staves.

`avoid-note-head` (boolean)
: If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`axes` (list) List of axis numbers. In the case of alignment grobs, this should contain only one number.

`base-shortest-duration` (moment)
: Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`baseline-skip` (dimension, in staff space)
: Distance between base lines of multiple lines of text.

`beam-thickness` (dimension, in staff space)
: Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space)
: Width of the tremolo sign.

`beamed-stem-shorten` (list)
: How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair)
: Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`beamlet-default-length` (pair)
: A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair)
: The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`before-line-breaking` (boolean)
: Dummy property, used to trigger a callback function.

`between-cols` (pair)
: Where to attach a loose column to.

`bound-details` (list)
: An alist of properties for determining attachments of spanners to edges.

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`bracket-visibility` (boolean or symbol)
: This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-align-anchor` (number)
: Grobs aligned to this break-align grob will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)
: Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-orders` (vector)
: Defines the order in which prefatory matter (clefs, key signatures) appears. The format is a vector of length 3, where each element is one order for end-of-line, middle of line, and start-of-line, respectively. An order is a list of symbols.

  For example, clefs are put after key signatures by setting

```
\override Score.BreakAlignment #'break-align-orders =
  #(make-vector 3 '(span-bar
                    breathing-sign
                    staff-bar
                    key
                    clef
                    time-signature))
```

`break-align-symbol` (symbol)
: This key is used for aligning and spacing breakable items.

`break-align-symbols` (list)
: A list of symbols that determine which break-aligned grobs to align this to. If the grob selected by the first symbol in the list is invisible due to break-visibility, we will align to the next grob (and so on). Choices are `left-edge`, `ambitus`, `breathing-sign`, `clef`, `staff-bar`, `key-cancellation`, `key-signature`, `time-signature`, and `custos`.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`break-visibility` (vector)
: A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`breakable` (boolean)
: Allow breaks here.

`c0-position` (integer)
: An integer indicating the position of middle C.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`clip-edges` (boolean)
: Allow outward pointing beamlets at the edges of beams?

`collapse-height` (dimension, in staff space)
: Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`collision-interfaces` (list)
: A list of interfaces for which automatic beam-collision resolution is run.

`collision-voice-only` (boolean)
: Does automatic beam collsion apply only to the voice in which the beam was created?

`color` (color)
: The color of this grob.

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`concaveness` (number)
: A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`control-points` (list)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`damping` (number)
: Amount of beam slope damping.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`dash-fraction` (number)
: Size of the dashes, relative to `dash-period`. Should be between `0.0` (no line) and `1.0` (continuous line).

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`default-direction` (direction)
: Direction determined by note head positions.

`default-staff-staff-spacing` (list)
: The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`digit-names` (vector)
: Names for string finger digits.

`direction` (direction)
: If `side-axis` is `0` (or `#X`), then this property determines whether the object is placed `#LEFT`, `#CENTER` or `#RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `#UP`, `#CENTER` or `#DOWN`. Numerical values may also be used: `#UP=1`, `#DOWN=-1`, `#LEFT=-1`, `#RIGHT=1`, `#CENTER=0`.

`dot-count` (integer)
: The number of dots.

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`dot-placement-list` (list)
: List consisting of `(`*`description string-number fret-number finger-number`*`)` entries used to define fret diagrams.

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`eccentricity` (number)
: How asymmetrical to make a slur. Positive means move the center to the right.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`edge-text` (pair)
: A pair specifying the texts to be set at the edges: `(`*`left-text . right-text`*`)`.

`expand-limit` (integer)
: Maximum number of measures expanded in church rests.

`extra-dy` (number)
: Slope glissandi this much extra.

`extra-offset` (pair of numbers)
: A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff's `StaffSymbol`.

`extra-spacing-height` (pair of numbers)
: In the horizontal spacing problem, we increase the height of each item by this amount (by adding the ‚car‘ to the bottom of the item and adding the ‚cdr‘ to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers)
: In the horizontal spacing problem, we pad each item by this amount (by adding the ‚car‘ on the left side of the item and adding the ‚cdr‘ on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`extra-X-extent` (pair of numbers)
: A grob is enlarged in X dimension by this much.

`extra-Y-extent` (pair of numbers)
: A grob is enlarged in Y dimension by this much.

`flag` (stencil)
: A function returning the full flag stencil for the `Stem`, which is passed to the function as the only argument. The default ly:stem::calc-stencil function uses the `flag-style` property to determine the correct glyph for the flag. By providing your own function, you can create arbitrary flags.

`flag-count` (number)
: The number of tremolo beams.

`flag-style` (symbol)
: A symbol determining what style of flag glyph is typeset on a `Stem`. Valid options include `'()` for standard flags, `'mensural` and `'no-flag`, which switches off the flag.

`font-encoding` (symbol)
: The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-family` (symbol)
: The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-name` (string)
: Specifies a file name (without extension) of the font to load. This setting overrides selection using `font-family`, `font-series` and `font-shape`.

`font-series` (symbol)
: Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol)
: Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number)
: The font size, compared to the ‚normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. Fractional values are allowed.

`footnote-text` (markup)
: A footnote for the grob.

`force-hshift` (number)
: This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Abschnitt "note-collision-interface" in *Referenz der Interna*.

`fraction` (pair of numbers)
: Numerator and denominator of a time signature object.

`french-beaming` (boolean)
: Use French beaming style for this stem. The stem stops at the innermost beams.

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a `(`*`property`* `.` *`value`*`)` pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
  - `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
  - `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
  - `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
  - `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
  - `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
  - `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
  - `fret-count` – The number of frets. Default 4.
  - `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
  - `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.
  - `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
  - `label-dir` – Side to which the fret label is attached. `-1`, `#LEFT`, or `#DOWN` for left or down; `1`, `#RIGHT`, or `#UP` for right or up. Default `#RIGHT`.
  - `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
  - `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.

- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`full-length-padding` (number)
: How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)
: Run to the extent of the column for a full-length tuplet bracket.

`full-measure-extra-space` (number)
: Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`full-size-change` (boolean)
: Don't make a change clef smaller.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`gap-count` (integer)
: Number of gapped beams for tremolo.

`glyph` (string)
: A string determining what ‚style' of glyph is typeset. Valid choices depend on the function that is reading this property.

`glyph-name` (string)
: The glyph name within the font.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`graphical` (boolean)
: Display in graphical (vs. text) form.

`grow-direction` (direction)
: Crescendo or decrescendo?

`hair-thickness` (number)
: Thickness of the thin line in a bar line.

`harp-pedal-details` (list)
: An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a `(property . value)` pair. The properties which can be included in harp-pedal-details include the following:

  - `box-offset` – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
  - `box-width` – Width of each pedal box. Default value 0.4.
  - `box-height` – Height of each pedal box. Default value 1.0.
  - `space-before-divider` – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
  - `space-after-divider` – Space between boxes after the first divider. Default value 0.8.
  - `circle-thickness` – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
  - `circle-x-padding` – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
  - `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`height-limit` (dimension, in staff space)
: Maximum slur height: The longer the slur, the closer it is to this height.

`hide-tied-accidental-after-break` (boolean)
: If set, an accidental that appears on a tied note after a line break will not be displayed.

`horizontal-shift` (integer)
: An integer that identifies ranking of `NoteColumn`s for horizontal shifting. This is used by Abschnitt “note-collision-interface” in *Referenz der Interna*.

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`ignore-collision` (boolean)
: If set, don’t do note collision resolution on this `NoteColumn`.

`implicit` (boolean)
: Is this an implicit bass figure?

`inspect-index` (integer)
: If debugging is set, set beam and slur configuration to this index, and print the respective scores.

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur quants to this position, and print the respective scores.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`kern` (dimension, in staff space)
: Amount of extra white space to add. For bar lines, this is the amount of space after a thick line.

`knee` (boolean)
: Is this beam kneed?

`knee-spacing-correction` (number)
: Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`labels` (list)
: List of labels (symbols) placed on a column.

`layer` (integer)
: An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-line-thickness` (pair of numbers)
: The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`left-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`left-padding` (dimension, in staff space)
: The amount of space that is put left to an object (e.g., a lyric extender).

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems.

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`line-break-penalty` (number)
: Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

`line-break-permission` (symbol)
: Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`line-break-system-details` (list)
: An alist of properties to use if this column is the start of a system.

`line-count` (integer)
: The number of staff lines.

`line-positions` (list)
: Vertical positions of staff lines.

`line-thickness` (number)
: The thickness of the tie or slur contour.

`long-text` (markup)
: Text markup. See Abschnitt “Formatting text” in *Notationsreferenz*.

`max-beam-connect` (integer)
: Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

`max-stretch` (number)
: The maximum amount that this `VerticalAxisGroup` can be vertically stretched (for example, in order to better fill a page).

`measure-count` (integer)
: The number of measures for a multi-measure rest.

`measure-length` (moment)
: Length of a measure. Used in some spacing situations.

`merge-differently-dotted` (boolean)
: Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

  `merge-differently-dotted` only applies to opposing stem directions (i.e., voice 1 & 2).

`merge-differently-headed` (boolean)
: Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Abschnitt "note-collision-interface" in *Referenz der Interna*.

  `merge-differently-headed` only applies to opposing stem directions (i.e., voice 1 & 2).

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-length-fraction` (number)
: Minimum length of ledger line as fraction of note head size.

`minimum-space` (dimension, in staff space)
: Minimum distance that the victim should move (after padding).

`minimum-X-extent` (pair of numbers)
: Minimum size of an object in X dimension, measured in `staff-space` units.

`minimum-Y-extent` (pair of numbers)
: Minimum size of an object in Y dimension, measured in `staff-space` units.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`neutral-position` (number)
: Position (in half staff spaces) where to flip the direction of custos stem.

`next` (graphical (layout) object)
: Object that is next relation (e.g., the lyric syllable following an extender).

`no-alignment` (boolean)
: If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`no-ledgers` (boolean)
: If set, don't draw ledger lines on this object.

`no-stem-extend` (boolean)
: If set, notes with ledger lines do not get stems extending to the middle staff line.

`non-break-align-symbols` (list)
: A list of symbols that determine which NON-break-aligned interfaces to align this to.

`non-default` (boolean)
: Set for manually specified clefs.

`non-musical` (boolean)
: True if the grob belongs to a `NonMusicalPaperColumn`.

`nonstaff-nonstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-relatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-unrelatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`normalized-endpoints` (pair)
: Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`outside-staff-horizontal-padding` (number)
: By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)
: The padding to place between this grob and the staff when spacing according to `outside-staff-priority`.

`outside-staff-priority` (number)
: If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`packed-spacing` (boolean)
: If set, the notes are spaced as tightly as possible.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`padding-pairs` (list)
: An alist mapping `(name . name)` to distances.

`page-break-penalty` (number)
: Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

`page-break-permission` (symbol)
: Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`page-turn-penalty` (number)
: Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

`page-turn-permission` (symbol)
: Instructs the page breaker on whether to put a page turn at this column. Can be `force` or `allow`.

`parenthesized` (boolean)
: Parenthesize this grob.

`positions` (pair of numbers)
: Pair of staff coordinates `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`prefer-dotted-right` (boolean)
: For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

`ratio` (number)
: Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`remove-empty` (boolean)
: If set, remove group if it contains no interesting items.

`remove-first` (boolean)
: Remove the first staff of an orchestral score?

`restore-first` (boolean)
: Print a natural before the accidental.

`rhythmic-location` (rhythmic location)
: Where (bar number, measure position) in the score.

`right-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`rotation` (list)
: Number of degrees to rotate this object, and what point to rotate around. For example, `#'(45 0 0)` rotates by 45 degrees around the center of this object.

`same-direction-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`script-priority` (number)
: A sorting key that determines in what order a script is within a stack of scripts.

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`shorten-pair` (pair of numbers)
: The lengths to shorten a text-spanner on both sides, for example a pedal bracket. Positive values shorten the text-spanner, while negative values lengthen it.

`shortest-duration-space` (dimension, in staff space)
: Start with this much space for the shortest duration. This is expressed in `spacing-increment` as unit. See also Abschnitt “spacing-spanner-interface” in *Referenz der Interna*.

`shortest-playing-duration` (moment)
: The duration of the shortest note playing here.

`shortest-starter-duration` (moment)
: The duration of the shortest note that starts here.

`side-axis` (number)
: If the value is `#X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `#Y` or `1`, it is placed vertically.

`side-relative-direction` (direction)
: Multiply direction of `direction-source` with this to get the direction of this object.

`size` (number)
: Size of object, relative to standard size.

`skyline-horizontal-padding` (number)
: For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`skyline-vertical-padding` (number)
: The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`slur-padding` (number)
: Extra distance between slur and script.

`space-alist` (list)
: A table that specifies distances between prefatory items, like clef and time-signature. The format is an alist of spacing tuples: `(`*`break-align-symbol type . distance`*`)`, where *type* can be the symbols `minimum-space` or `extra-space`.

`space-to-barline` (boolean)
: If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`spacing-increment` (number)
: Add this much space for a doubled duration. Typically, the width of a note head. See also Abschnitt "spacing-spanner-interface" in *Referenz der Interna*.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

  ```
  \override MultiMeasureRest #'spacing-pair = #'(staff-bar . staff-bar)
  ```

`springs-and-rods` (boolean)
: Dummy variable for triggering spacing routines.

`stacking-dir` (direction)
: Stack objects in which direction?

`staff-affinity` (direction)
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

`staff-space` (dimension, in staff space)
: Amount of space between staff lines, expressed in global `staff-space`.

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
  - `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.

- `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list)
: The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

`stem-attachment` (pair of numbers)
: An `(x . y)` pair where the stem attaches to the notehead.

`stem-end-position` (number)
: Where does the stem end (the end is opposite to the support-head)?

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

`stemlet-length` (number)
: How long should be a stem over a rest?

`stencil` (stencil)
: The symbol to print.

`stencils` (list)
: Multiple stencils, used as intermediate value.

`strict-grace-spacing` (boolean)
: If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

`strict-note-spacing` (boolean)
: If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

`stroke-style` (string)
: Set to `"grace"` to turn stroke through flag on.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`text` (markup)
: Text markup. See Abschnitt "Formatting text" in *Notationsreferenz*.

`text-direction` (direction)
: This controls the ordering of the words. The default `RIGHT` is for roman text. Arabic or Hebrew should use `LEFT`.

`thick-thickness` (number)
: Bar line thickness, measured in `line-thickness`.

`thickness` (number)
: Line thickness, generally measured in `line-thickness`.

`thin-kern` (number)
: The space after a hair-line in a bar line.

`tie-configuration` (list)
: List of `(`*`position . dir`*`)` pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

`toward-stem-shift` (number)
: Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. `0.0` means keep the default position (centered on the note head), `1.0` means centered on the stem. Interpolated values are possible.

`transparent` (boolean)
: This makes the grob invisible.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their durations. This looks better in complex polyphonic patterns.

`used` (boolean)
: If set, this spacing column is kept in the spacing problem.

`vertical-skylines` (pair of skylines)
: Two skylines, one above and one below this grob.

`when` (moment)
: Global time step associated with this column happen?

`whiteout` (boolean)
: If true, the grob is printed over a white background to white-out underlying material, if the grob is visible. Usually #f by default.

`width` (dimension, in staff space)
: The width of a grob measured in staff space.

`word-space` (dimension, in staff space)
: Space to insert between words in texts.

`X-extent` (pair of numbers)
: Hard coded extent in X direction.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers)
: Hard coded extent in Y direction.

`Y-offset` (number)
: The vertical amount that this object is moved relative to its Y-parent.

`zigzag-length` (dimension, in staff space)
: The length of the lines of a zigzag, relative to `zigzag-width`. A value of `1` gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)
: The width of one zigzag squiggle. This number is adjusted slightly so that the glissando line can be constructed from a whole number of squiggles.

A.16 Erhältliche Musikfunktionen

`acciaccatura` - *music* (music)
: Create an acciaccatura from the following music expression

`addChordShape` - *key-symbol* (symbol) *tuning* (pair) *shape-definition* (string or pair)
: Add chord shape *shape-definition* to the *chord-shape-table* hash with the key *(cons key-symbol tuning)*.

`addInstrumentDefinition` - *name* (string) *lst* (list)
: Create instrument *name* with properties *list*.

`addQuote` - *name* (string) *music* (music)
: Define *music* as a quotable music expression named *name*

`afterGrace` - *main* (music) *grace* (music)
: Create *grace* note(s) after a *main* music expression.

`allowPageTurn`
: Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

`applyContext` - *proc* (procedure)
: Modify context properties with Scheme procedure *proc*.

`applyMusic` - *func* (procedure) *music* (music)
: Apply procedure *func* to *music*.

`applyOutput` - *ctx* (symbol) *proc* (procedure)
: Apply function `proc` to every layout object in context `ctx`

`appoggiatura` - *music* (music)
: Create an appoggiatura from *music*

`assertBeamQuant` - *l* (pair) *r* (pair)
: Testing function: check whether the beam quants *l* and *r* are correct

`assertBeamSlope` - *comp* (procedure)
: Testing function: check whether the slope of the beam is the same as `comp`

`autochange` - *music* (music)
: Make voices that switch between staves automatically

`balloonGrobText` - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
: Attach *text* to *grob-name* at offset *offset* (use like `\once`)

`balloonText` - *offset* (pair of numbers) *text* (markup)
: Attach *text* at *offset* (use like `\tweak`)

`bar` - *type* (string)
: Insert a bar line of type *type*

`barNumberCheck` - *n* (integer)
: Print a warning if the current bar number is not *n*.

`bendAfter` - *delta* (real number)
: Create a fall or doit of pitch interval *delta*.

`bookOutputName` - *newfilename* (string)
: Direct output for the current book block to *newfilename*.

`bookOutputSuffix` - *newsuffix* (string)
: Set the output filename suffix for the current book block to *newsuffix*.

`breathe` Insert a breath mark.

`chordGlissando` - *mus1* (music) *mus2* (music)
Make a glissando between the notes of triads `mus1` and `mus2`.

`clef` - *type* (string)
Set the current clef to *type*.

`compoundMeter` - *args* (pair)
Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the enumerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of (3+1)/8 + 2/4 would be created as `\compoundMeter #'((3 1 8) (2 4))`, and a time signature of (3+2)/8 as `\compoundMeter #'((3 2 8))` or shorter `\compoundMeter #'(3 2 8)`.

`contextStringTuning` - *tuning* (symbol) *chord* (music)
(undocumented; fixme)

`cueClef` - *type* (string)
Set the current cue clef to *type*.

`cueClefUnset`
Unset the current cue clef.

`cueDuring` - *what* (string) *dir* (direction) *main-music* (music)
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`cueDuringWithClef` - *what* (string) *dir* (direction) *clef* (string) *main-music* (music)
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`deadNote` - *note* (music)
Print *note* with a cross-shaped note head.

`defaultNoteHeads`
Revert to the default note head style.

`displayLilyMusic` - *music* (music)
Display the LilyPond input representation of *music* to the console.

`displayMusic` - *music* (music)
Display the internal representation of *music* to the console.

`endSpanners` - *music* (music)
Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.

`featherDurations` - *factor* (moment) *argument* (music)
Adjust durations of music in *argument* by rational *factor*.

`footnote` - *offset* (pair of numbers) *text* (markup) *footnote* (markup)
Attach *text* at *offset* with *text* referring to *footnote* (use like `\tweak`)

`footnoteGrob` - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup) *footnote* (markup)
Attach *text* to *grob-name* at offset *offset*, with *text* referring to *footnote* (use like `\once`)

`grace` - *music* (music)
Insert *music* as grace notes.

`harmonicByFret` - *fret* (number) *music* (music)
(undocumented; fixme)

`harmonicByRatio` - *ratio* (number) *music* (music)
(undocumented; fixme)

`harmonicNote` - *note* (music)
Print *note* with a diamond-shaped note head.

`harmonicsOn`
Set the default note head style to a diamond-shaped style.

`instrumentSwitch` - *name* (string)
Switch instrument to *name*, which must be predefined with `\addInstrumentDefinition`.

`inversion` - *around* (music) *to* (music) *music* (music)
Invert *music* about *around* and transpose from *around* to *to*.

`keepWithTag` - *tag* (symbol) *music* (music)
Include only elements of *music* that are tagged with *tag*.

`killCues` - *music* (music)
Remove cue notes from *music*.

`label` - *label* (symbol)
Create *label* as a bookmarking label.

`language` - *language* (string)
Set note names for language *language*.

`languageRestore`
Restore a previously-saved pitchnames alist.

`languageSaveAndChange` - *language* (string)
Store the previous pitchnames alist, and set a new one.

`makeClusters` - *arg* (music)
Display chords in *arg* as clusters.

`makeDefaultStringTunings` - *default-tuning-alist* (list)
(undocumented; fixme)

`makeStringTuning` - *tuning* (symbol) *chord* (music)
(undocumented; fixme)

`modalInversion` - *around* (music) *to* (music) *scale* (music) *music* (music)
Invert *music* about *around* using *scale* and transpose from *around* to *to*.

`modalTranspose` - *from* (music) *to* (music) *scale* (music) *music* (music)
Transpose *music* from pitch *from* to pitch *to* using *scale*.

`musicMap` - *proc* (procedure) *mus* (music)
Apply *proc* to *mus* and all of the music it contains.

`noPageBreak`
Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`noPageTurn`
Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.

`octaveCheck` - *pitch-note* (music)
Octave check.

`ottava` - *octave* (integer)
Set the octavation.

`overrideProperty` - *name* (string) *property* (symbol) *value* (any type)
Set *property* to *value* in all grobs named *name*. The *name* argument is a string of the form `"Context.GrobName"` or `"GrobName"`.

`overrideTimeSignatureSettings` - *time-signature* (pair) *base-moment* (pair) *beat-structure* (list) *beam-exceptions* (list)
Override `timeSignatureSettings` for time signatures of *time-signature* to have settings of *base-moment*, *beat-structure*, and *beam-exceptions*.

`pageBreak`
Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`pageTurn` Force a page turn between two scores or top-level markups.

`palmMute` - *note* (music)
Print *note* with a triangle-shaped note head.

`palmMuteOn`
Set the default note head style to a triangle-shaped style.

`parallelMusic` - *voice-ids* (list) *music* (music)
Define parallel music sequences, separated by '`|`' (bar check signs), and assign them to the identifiers provided in *voice-ids*.

voice-ids: a list of music identifiers (symbols containing only letters)

music: a music sequence, containing BarChecks as limiting expressions.

Example:

```
  \parallelMusic #'(A B C) {
    c c | d d | e e |
    d d | e e | f f |
  }
<==>
  A = { c c | d d | }
  B = { d d | e e | }
  C = { e e | f f | }
```

`parenthesize` - *arg* (music)
Tag *arg* to be parenthesized.

`partcombine` - *part1* (music) *part2* (music)
Take the music in *part1* and *part2* and typeset so that they share a staff.

`partcombineForce` - *type* (symbol-or-boolean) *once* (boolean)
Override the part-combiner.

`phrasingSlurDashPattern` - *dash-fraction* (number) *dash-period* (number)
Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval.

`pitchedTrill` - *main-note* (music) *secondary-note* (music)
Print a trill with *main-note* as the main note of the trill and print *secondary-note* as a stemless note head in parentheses.

`pointAndClickOff`
: Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

`pointAndClickOn`
: Enable generation of code in final-format (e.g. pdf) files to reference the originating lilypond source statement; this is helpful when developing a score but generates bigger final-format files.

`quoteDuring` - *what* (string) *main-music* (music)
: Indicate a section of music to be quoted. *what* indicates the name of the quoted voice, as specified in an `\addQuote` command. *main-music* is used to indicate the length of music to be quoted; usually contains spacers or multi-measure rests.

`removeWithTag` - *tag* (symbol) *music* (music)
: Remove elements of *music* that are tagged with *tag*.

`resetRelativeOctave` - *reference-note* (music)
: Set the octave inside a \relative section.

`retrograde` - *music* (music)
: Return *music* in reverse order.

`revertTimeSignatureSettings` - *time-signature* (pair)
: Revert `timeSignatureSettings` for time signatures of *time-signature*.

`rightHandFinger` - *finger* (number or string)
: Apply *finger* as a fingering indication.

`scaleDurations` - *fraction* (pair of numbers) *music* (music)
: Multiply the duration of events in *music* by *fraction*.

`shiftDurations` - *dur* (integer) *dots* (integer) *arg* (music)
: Scale *arg* up by a factor of *2^dur*(2-(1/2)^dots)*.

`slurDashPattern` - *dash-fraction* (number) *dash-period* (number)
: (undocumented; fixme)

`spacingTweaks` - *parameters* (list)
: Set the system stretch, by reading the 'system-stretch property of the 'parameters' assoc list.

`storePredefinedDiagram` - *fretboard-table* (hash table) *chord* (music) *tuning* (pair) *diagram-definition* (string or pair)
: Add predefined fret diagram defined by *diagram-definition* for the chord pitches *chord* and the stringTuning *tuning*.

`styledNoteHeads` - *style* (symbol) *heads* (list or symbol) *music* (music)
: Set *heads* in *music* to *style*.

`tabChordRepetition`
: Include the string information in a chord repetition.

`tag` - *tag* (symbol) *arg* (music)
: Add *tag* to the `tags` property of *arg*.

`tieDashPattern` - *dash-fraction* (number) *dash-period* (number)
: (undocumented; fixme)

`tocItem` - *text* (markup)
: Add a line to the table of content, using the `tocItemMarkup` paper variable markup

`transposedCueDuring` - *what* (string) *dir* (direction) *pitch-note* (music) *main-music* (music)
: Insert notes from the part *what* into a voice called `cue`, using the transposition defined by *pitch-note*. This happens simultaneously with *main-music*, which is usually a rest. The argument *dir* determines whether the cue notes should be notated as a first or second voice.

`transposition` - *pitch-note* (music)
: Set instrument transposition

`tweak` - *sym* (symbol) *val* (any type) *arg* (music)
: Add `sym . val` to the `tweaks` property of *arg*.

`unfoldRepeats` - *music* (music)
: Force any `\repeat volta`, `\repeat tremolo` or `\repeat percent` commands in *music* to be interpreted as `\repeat unfold`.

`withMusicProperty` - *sym* (symbol) *val* (any type) *music* (music)
: Set *sym* to *val* in *music*.

`xNote` - *note* (music)
: Print *note* with a cross-shaped note head.

`xNotesOn` Set the default note head style to a cross-shaped style.

A.17 Vordefinierte Typenprädikate

R5RS primary predicates

Type predicate	**Description**
`boolean?`	boolean
`char?`	character
`number?`	number
`pair?`	pair
`port?`	port
`procedure?`	procedure
`string?`	string
`symbol?`	symbol
`vector?`	vector

R5RS secondary predicates

Type predicate	**Description**
`char-alphabetic?`	alphabetic character
`char-lower-case?`	lower-case character
`char-numeric?`	numeric character
`char-upper-case?`	upper-case character
`char-whitespace?`	whitespace character
`complex?`	complex number
`eof-object?`	end-of-file object
`even?`	even number
`exact?`	exact number
`inexact?`	inexact number
`input-port?`	input port
`integer?`	integer
`list?`	list *(use* `cheap-list?` *for faster processing)*

`negative?`	negative number
`null?`	null
`odd?`	odd number
`output-port?`	output port
`positive?`	positive number
`rational?`	rational number
`real?`	real number
`zero?`	zero

Guile predicates

Type predicate	**Description**
`hash-table?`	hash table

LilyPond scheme predicates

Type predicate	**Description**
`boolean-or-symbol?`	boolean or symbol
`cheap-list?`	list *(use this instead of* `list?` *for faster processing)*
`color?`	color
`grob-list?`	list of grobs
`list-or-symbol?`	list or symbol
`markup?`	markup
`markup-command-list?`	markup command list
`markup-list?`	markup list
`moment-pair?`	pair of moment objects
`number-or-grob?`	number or grob
`number-or-pair?`	number or pair
`number-or-string?`	number or string
`number-pair?`	pair of numbers
`rhythmic-location?`	rhythmic location
`scheme?`	any type
`string-or-pair?`	string or pair
`string-or-symbol?`	string or symbol

LilyPond exported predicates

Type predicate	**Description**
`ly:box?`	box
`ly:context?`	context
`ly:dimension?`	dimension, in staff space
`ly:dir?`	direction
`ly:dispatcher?`	dispatcher
`ly:duration?`	duration
`ly:font-metric?`	font metric
`ly:grob?`	graphical (layout) object
`ly:grob-array?`	array of grobs
`ly:input-location?`	input location
`ly:item?`	item
`ly:iterator?`	iterator
`ly:lily-lexer?`	lily-lexer

`ly:lily-parser?`	lily-parser
`ly:listener?`	listener
`ly:moment?`	moment
`ly:music?`	music
`ly:music-function?`	music function
`ly:music-list?`	list of music objects
`ly:music-output?`	music output
`ly:otf-font?`	OpenType font
`ly:output-def?`	output definition
`ly:page-marker?`	page marker
`ly:pango-font?`	pango font
`ly:paper-book?`	paper book
`ly:paper-system?`	paper-system Prob
`ly:pitch?`	pitch
`ly:prob?`	property object
`ly:score?`	score
`ly:simple-closure?`	simple closure
`ly:skyline?`	skyline
`ly:skyline-pair?`	pair of skylines
`ly:source-file?`	source file
`ly:spanner?`	spanner
`ly:stencil?`	stencil
`ly:stream-event?`	stream event
`ly:translator?`	translator
`ly:translator-group?`	translator group

A.18 Scheme-Funktionen

`ly:add-context-mod` *contextmods modification* [Funktion]
Adds the given context *modification* to the list *contextmods* of context modifications.

`ly:add-file-name-alist` *alist* [Funktion]
Add mappings for error messages from *alist*.

`ly:add-interface` *iface desc props* [Funktion]
Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

`ly:add-listener` *list disp cl* [Funktion]
Add the listener *list* to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it is forwarded to *list*.

`ly:add-option` *sym val description* [Funktion]
Add a program option *sym*. *val* is the default value and *description* is a string description.

`ly:all-grob-interfaces` [Funktion]
Return the hash table with all grob interface descriptions.

`ly:all-options` [Funktion]
Get all option settings in an alist.

`ly:all-stencil-expressions` [Funktion]
Return all symbols recognized as stencil expressions.

`ly:assoc-get` *key alist default-value strict-checking* [Funktion]
Return value if *key* in *alist*, else *default-value* (or `#f` if not specified). If *strict-checking* is set to `#t` and *key* is not in *alist*, a programming_error is output.

`ly:axis-group-interface::add-element` *grob grob-element* [Funktion]
Set *grob* the parent of *grob-element* on all axes of *grob*.

`ly:beam-score-count` [Funktion]
count number of beam scores.

`ly:book-add-bookpart!` *book-smob book-part* [Funktion]
Add *book-part* to *book-smob* book part list.

`ly:book-add-score!` *book-smob score* [Funktion]
Add *score* to *book-smob* score list.

`ly:book-book-parts` *book* [Funktion]
Return book parts in *book*.

`ly:book-header` *book* [Funktion]
Return header in *book*.

`ly:book-paper` *book* [Funktion]
Return paper in *book*.

`ly:book-process` *book-smob default-paper default-layout output* [Funktion]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-process-to-systems` *book-smob default-paper default-layout output* [Funktion]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-scores` *book* [Funktion]
Return scores in *book*.

`ly:box?` *x* [Funktion]
Is *x* a `Box` object?

`ly:bp` *num* [Funktion]
num bigpoints (1/72th inch).

`ly:bracket` *a iv t p* [Funktion]
Make a bracket in direction *a*. The extent of the bracket is given by *iv*. The wings protrude by an amount of *p*, which may be negative. The thickness is given by *t*.

`ly:broadcast` *disp ev* [Funktion]
Send the stream event *ev* to the dispatcher *disp*.

`ly:camel-case->lisp-identifier` *name-sym* [Funktion]
Convert `FooBar_Bla` to `foo-bar-bla` style symbol.

`ly:chain-assoc-get` *key achain default-value strict-checking* [Funktion]
Return value for *key* from a list of alists *achain*. If no entry is found, return *default-value* or `#f` if *default-value* is not specified. With *strict-checking* set to `#t`, a programming_error is output in such cases.

`ly:cm` *num* [Funktion]
num cm.

`ly:command-line-code` [Funktion]
The Scheme code specified on command-line with '`-e`'.

`ly:command-line-options` [Funktion]
The Scheme options specified on command-line with '`-d`'.

`ly:command-line-verbose?` [Funktion]
Was `be_verbose_global` set?

`ly:connect-dispatchers` *to from* [Funktion]
Make the dispatcher *to* listen to events from *from*.

`ly:context?` *x* [Funktion]
Is *x* a `Context` object?

`ly:context-current-moment` *context* [Funktion]
Return the current moment of *context*.

`ly:context-event-source` *context* [Funktion]
Return `event-source` of context *context*.

`ly:context-events-below` *context* [Funktion]
Return a `stream-distributor` that distributes all events from *context* and all its subcontexts.

`ly:context-find` *context name* [Funktion]
Find a parent of *context* that has name or alias *name*. Return `#f` if not found.

`ly:context-grob-definition` *context name* [Funktion]
Return the definition of *name* (a symbol) within *context* as an alist.

`ly:context-id` *context* [Funktion]
Return the ID string of *context*, i.e., for `\context Voice = "one" ...` return the string `one`.

`ly:context-name` *context* [Funktion]
Return the name of *context*, i.e., for `\context Voice = "one" ...` return the symbol `Voice`.

`ly:context-now` *context* [Funktion]
Return `now-moment` of context *context*.

`ly:context-parent` *context* [Funktion]
Return the parent of *context*, `#f` if none.

`ly:context-property` *context sym def* [Funktion]
Return the value for property *sym* in *context*. If *def* is given, and property value is `'()`, return *def*.

`ly:context-property-where-defined` *context name* [Funktion]
Return the context above *context* where *name* is defined.

`ly:context-pushpop-property` *context grob eltprop val* [Funktion]
Do a single `\override` or `\revert` operation in *context*. The grob definition *grob* is extended with *eltprop* (if *val* is specified) or reverted (if unspecified).

`ly:context-set-property!` *context name val* [Funktion]
Set value of property *name* in context *context* to *val*.

`ly:context-unset-property` *context name* [Funktion]
Unset value of property *name* in context *context*.

`ly:default-scale` [Funktion]
Get the global default scale.

`ly:dimension?` *d* [Funktion]
Return *d* as a number. Used to distinguish length variables from normal numbers.

`ly:dir?` *s* [Funktion]
Is *s* a direction? Valid directions are `-1`, `0`, or `1`, where `-1` represents left or down, `1` represents right or up, and `0` represents a neutral direction.

`ly:dispatcher?` *x* [Funktion]
Is *x* a `Dispatcher` object?

`ly:duration?` *x* [Funktion]
Is *x* a `Duration` object?

`ly:duration<?` *p1 p2* [Funktion]
Is *p1* shorter than *p2*?

`ly:duration->string` *dur* [Funktion]
Convert *dur* to a string.

`ly:duration-dot-count` *dur* [Funktion]
Extract the dot count from *dur*.

`ly:duration-factor` *dur* [Funktion]
Extract the compression factor from *dur*. Return it as a pair.

`ly:duration-length` *dur* [Funktion]
The length of the duration as a `moment`.

`ly:duration-log` *dur* [Funktion]
Extract the duration log from *dur*.

`ly:effective-prefix` [Funktion]
Return effective prefix.

`ly:encode-string-for-pdf` *str* [Funktion]
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that's not possible to full UTF-16BE with Byte-Order-Mark (BOM).

`ly:engraver-announce-end-grob` *engraver grob cause* [Funktion]
Announce the end of a grob (i.e., the end of a spanner) originating from given *engraver* instance, with *grob* being a grob. *cause* should either be another grob or a music event.

`ly:engraver-make-grob` *engraver grob-name cause* [Funktion]
Create a grob originating from given *engraver* instance, with given *grob-name*, a symbol. *cause* should either be another grob or a music event.

`ly:error` *str rest* [Funktion]
A Scheme callable function to issue the error *str*. The error is formatted with `format` and *rest*.

`ly:eval-simple-closure` *delayed closure scm-start scm-end* [Funktion]
Evaluate a simple *closure* with the given *delayed* argument. If *scm-start* and *scm-end* are defined, evaluate it purely with those start and end points.

`ly:event-deep-copy` *m* [Funktion]
Copy *m* and all sub expressions of *m*.

`ly:event-property` *sev sym* [Funktion]
Get the property *sym* of stream event *mus*. If *sym* is undefined, return `'()`.

`ly:event-set-property!` *ev sym val* [Funktion]
Set property *sym* in event *ev* to *val*.

`ly:expand-environment` *str* [Funktion]
Expand `$VAR` and `${VAR}` in *str*.

`ly:export` *arg* [Funktion]
Export a Scheme object to the parser so it is treated as an identifier.

`ly:find-file` *name* [Funktion]
Return the absolute file name of *name*, or `#f` if not found.

`ly:font-config-add-directory` *dir* [Funktion]
Add directory *dir* to FontConfig.

`ly:font-config-add-font` *font* [Funktion]
Add font *font* to FontConfig.

`ly:font-config-display-fonts` [Funktion]
Dump a list of all fonts visible to FontConfig.

`ly:font-config-get-font-file` *name* [Funktion]
Get the file for font *name*.

`ly:font-design-size` *font* [Funktion]
Given the font metric *font*, return the design size, relative to the current output-scale.

`ly:font-file-name` *font* [Funktion]
Given the font metric *font*, return the corresponding file name.

`ly:font-get-glyph` *font name* [Funktion]
Return a stencil from *font* for the glyph named *name*. If the glyph is not available, return an empty stencil.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-charcode` *font name* [Funktion]
Return the character code for glyph *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-index` *font name* [Funktion]
Return the index for *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-index-to-charcode` *font index* [Funktion]
Return the character code for *index* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-magnification` *font* [Funktion]
Given the font metric *font*, return the magnification, relative to the current output-scale.

`ly:font-metric?` *x* [Funktion]
Is *x* a `Font_metric` object?

`ly:font-name` *font* [Funktion]
Given the font metric *font*, return the corresponding name.

`ly:font-sub-fonts` *font* [Funktion]
Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

`ly:format` *str rest* [Funktion]
LilyPond specific format, supporting `~a` and `~[0-9]f`. Basic support for `~s` is also provided.

`ly:format-output` *context* [Funktion]
Given a global context in its final state, process it and return the `Music_output` object in its final state.

`ly:get-all-function-documentation` [Funktion]
Get a hash table with all LilyPond Scheme extension functions.

`ly:get-all-translators` [Funktion]
Return a list of all translator objects that may be instantiated.

`ly:get-context-mods` *contextmod* [Funktion]
Returns the list of context modifications stored in *contextmod*.

`ly:get-listened-event-classes` [Funktion]
Return a list of all event classes that some translator listens to.

`ly:get-option` *var* [Funktion]
Get a global option setting.

`ly:gettext` *original* [Funktion]
A Scheme wrapper function for `gettext`.

`ly:grob?` *x* [Funktion]
Is *x* a `Grob` object?

`ly:grob-alist-chain` *grob global* [Funktion]
Get an alist chain for grob *grob*, with *global* as the global default. If unspecified, `font-defaults` from the layout block is taken.

`ly:grob-array?` *x* [Funktion]
Is *x* a `Grob_array` object?

`ly:grob-array->list` *grob-arr* [Funktion]
Return the elements of *grob-arr* as a Scheme list.

`ly:grob-array-length` *grob-arr* [Funktion]
Return the length of *grob-arr*.

`ly:grob-array-ref` *grob-arr index* [Funktion]
Retrieve the *index*th element of *grob-arr*.

`ly:grob-basic-properties` *grob* [Funktion]
Get the immutable properties of *grob*.

`ly:grob-chain-callback` *grob proc sym* [Funktion]
Find the callback that is stored as property *sym* of grob *grob* and chain *proc* to the head of this, meaning that it is called using *grob* and the previous callback's result.

`ly:grob-common-refpoint` *grob other axis* [Funktion]
Find the common refpoint of *grob* and *other* for *axis*.

`ly:grob-common-refpoint-of-array` *grob others axis* [Funktion]
Find the common refpoint of *grob* and *others* (a grob-array) for *axis*.

`ly:grob-default-font` *grob* [Funktion]
Return the default font for grob *grob*.

`ly:grob-extent` *grob refp axis* [Funktion]
Get the extent in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-interfaces` *grob* [Funktion]
Return the interfaces list of grob *grob*.

`ly:grob-layout` *grob* [Funktion]
Get `\layout` definition from grob *grob*.

`ly:grob-object` *grob sym* [Funktion]
Return the value of a pointer in grob *grob* of property *sym*. It returns `'()` (end-of-list) if *sym* is undefined in *grob*.

`ly:grob-original` *grob* [Funktion]
Return the unbroken original grob of *grob*.

`ly:grob-parent` *grob axis* [Funktion]
Get the parent of *grob*. *axis* is 0 for the X-axis, 1 for the Y-axis.

`ly:grob-pq<?` *a b* [Funktion]
Compare two grob priority queue entries. This is an internal function.

`ly:grob-properties` *grob* [Funktion]
Get the mutable properties of *grob*.

`ly:grob-property` *grob sym val* [Funktion]
Return the value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-property-data` *grob sym* [Funktion]
Return the value for property *sym* of *grob*, but do not process callbacks.

`ly:grob-relative-coordinate` *grob refp axis* [Funktion]
Get the coordinate in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-robust-relative-extent` *grob refp axis* [Funktion]
Get the extent in *axis* direction of *grob* relative to the grob *refp*, or `(0,0)` if empty.

`ly:grob-script-priority-less` *a b* [Funktion]
Compare two grobs by script priority. For internal use.

`ly:grob-set-nested-property!` *grob symlist val* [Funktion]
Set nested property *symlist* in grob *grob* to value *val*.

`ly:grob-set-object!` *grob sym val* [Funktion]
Set *sym* in grob *grob* to value *val*.

`ly:grob-set-parent!` *grob axis parent-grob* [Funktion]
Set *parent-grob* the parent of grob *grob* in axis *axis*.

`ly:grob-set-property!` *grob sym val* [Funktion]
Set *sym* in grob *grob* to value *val*.

`ly:grob-staff-position` *sg* [Funktion]
Return the Y-position of *sg* relative to the staff.

`ly:grob-suicide!` *grob* [Funktion]
Kill *grob*.

`ly:grob-system` *grob* [Funktion]
Return the system grob of *grob*.

`ly:grob-translate-axis!` *grob d a* [Funktion]
Translate *grob* on axis *a* over distance *d*.

`ly:gulp-file` *name size* [Funktion]
Read *size* characters from the file *name*, and return its contents in a string. If *size* is undefined, the entire file is read. The file is looked up using the search path.

`ly:hash-table-keys` *tab* [Funktion]
Return a list of keys in *tab*.

`ly:inch` *num* [Funktion]
num inches.

`ly:input-both-locations` *sip* [Funktion]
Return input location in *sip* as `(file-name first-line first-column last-line last-column)`.

`ly:input-file-line-char-column` *sip* [Funktion]
Return input location in *sip* as `(file-name line char column)`.

`ly:input-location?` *x* [Funktion]
Is *x* an `input-location`?

`ly:input-message` *sip msg rest* [Funktion]
Print *msg* as a GNU compliant error message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:interpret-music-expression` *mus ctx* [Funktion]
Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.

`ly:interpret-stencil-expression` *expr func arg1 offset* [Funktion]
Parse *expr*, feed bits to *func* with first arg *arg1* having offset *offset*.

`ly:intlog2` *d* [Funktion]
The 2-logarithm of $1/d$.

`ly:is-listened-event-class` *sym* [Funktion]
Is *sym* a listened event class?

`ly:item?` *g* [Funktion]
Is *g* an `Item` object?

`ly:item-break-dir` *it* [Funktion]
The break status direction of item *it*. `-1` means end of line, `0` unbroken, and `1` beginning of line.

`ly:iterator?` *x* [Funktion]
Is *x* a `Music_iterator` object?

`ly:lexer-keywords` *lexer* [Funktion]
Return a list of `(KEY . CODE)` pairs, signifying the LilyPond reserved words list.

`ly:lily-lexer?` *x* [Funktion]
Is *x* a `Lily_lexer` object?

`ly:lily-parser?` *x* [Funktion]
Is *x* a `Lily_parser` object?

`ly:listener?` *x* [Funktion]
Is *x* a `Listener` object?

`ly:make-book` *paper header scores* [Funktion]
Make a `\book` of *paper* and *header* (which may be `#f` as well) containing `\scores`.

`ly:make-book-part` *scores* [Funktion]
Make a `\bookpart` containing `\scores`.

`ly:make-dispatcher` [Funktion]
Return a newly created dispatcher.

`ly:make-duration` *length dotcount num den* [Funktion]
length is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.

The duration factor is optionally given by *num* and *den*.

A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.

`ly:make-global-context` *output-def* [Funktion]
Set up a global interpretation context, using the output block *output-def*. The context is returned.

`ly:make-global-translator` *global* [Funktion]
Create a translator group and connect it to the global context *global*. The translator group is returned.

`ly:make-listener` *callback* [Funktion]
Create a listener. Any time the listener hears an object, it will call *callback* with that object.

callback should take exactly one argument.

`ly:make-moment` *n d gn gd* [Funktion]
Create the rational number with main timing n/d, and optional grace timing gn/gd.

A *moment* is a point in musical time. It consists of a pair of rationals (m, g), where m is the timing for the main notes, and g the timing for grace notes. In absence of grace notes, g is zero.

`ly:make-music` *props* [Funktion]
Make a C++ `Music` object and initialize it with *props*.

This function is for internal use and is only called by `make-music`, which is the preferred interface for creating music objects.

`ly:make-music-function` *signature func* [Funktion]
Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature* is a list containing either `ly:music?` predicates or other type predicates.

`ly:make-output-def` [Funktion]
Make an output definition.

`ly:make-page-label-marker` *label* [Funktion]
Return page marker with label *label*.

`ly:make-page-permission-marker` *symbol permission* [Funktion]
Return page marker with page breaking and turning permissions.

`ly:make-pango-description-string` *chain size* [Funktion]
Make a `PangoFontDescription` string for the property alist *chain* at size *size*.

`ly:make-paper-outputter` *port format* [Funktion]
Create an outputter that evaluates within `output-`*format*, writing to *port*.

`ly:make-pitch` *octave note alter* [Funktion]
octave is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. *alter* is a rational number of 200-cent whole tones for alteration.

`ly:make-prob` *type init rest* [Funktion]
Create a `Prob` object.

`ly:make-scale` *steps* [Funktion]
Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.

`ly:make-score` *music* [Funktion]
Return score with *music* encapsulated in it.

`ly:make-simple-closure` *expr* [Funktion]
Make a simple closure. *expr* should be form of `(func a1 a2 ...)`, and will be invoked as `(func delayed-arg a1 a2 ...)`.

`ly:make-stencil` *expr xext yext* [Funktion]
Stencils are device independent output expressions. They carry two pieces of information:

1. A specification of how to print this object. This specification is processed by the output backends, for example '`scm/output-ps.scm`'.
2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use `(1000 . -1000)` as its value), it is taken to be empty.

`ly:make-stream-event` *cl proplist* [Funktion]
Create a stream event of class *cl* with the given mutable property list.

`ly:message` *str rest* [Funktion]
A Scheme callable function to issue the message *str*. The message is formatted with `format` and *rest*.

`ly:minimal-breaking` *pb* [Funktion]
Break (pages and lines) the `Paper_book` object *pb* without looking for optimal spacing: stack as many lines on a page before moving to the next one.

`ly:mm` *num* [Funktion]
num mm.

`ly:module->alist` *mod* [Funktion]
Dump the contents of module *mod* as an alist.

`ly:module-copy` *dest src* [Funktion]
Copy all bindings from module *src* into *dest*.

`ly:modules-lookup` *modules sym def* [Funktion]
Look up *sym* in the list *modules*, returning the first occurence. If not found, return *def* or `#f` if *def* isn't specified.

`ly:moment?` *x* [Funktion]
Is *x* a `Moment` object?

`ly:moment<?` *a b* [Funktion]
Compare two moments.

`ly:moment-add` *a b* [Funktion]
Add two moments.

`ly:moment-div` *a b* [Funktion]
Divide two moments.

`ly:moment-grace-denominator` *mom* [Funktion]
Extract denominator from grace timing.

`ly:moment-grace-numerator` *mom* [Funktion]
Extract numerator from grace timing.

`ly:moment-main-denominator` *mom* [Funktion]
Extract denominator from main timing.

`ly:moment-main-numerator` *mom* [Funktion]
Extract numerator from main timing.

`ly:moment-mod` *a b* [Funktion]
Modulo of two moments.

`ly:moment-mul` *a b* [Funktion]
Multiply two moments.

`ly:moment-sub` *a b* [Funktion]
Subtract two moments.

`ly:music?` *obj* [Funktion]
Is *obj* a music object?

`ly:music-compress` *m factor* [Funktion]
Compress music object *m* by moment *factor*.

`ly:music-deep-copy` *m* [Funktion]
Copy *m* and all sub expressions of *m*.

`ly:music-duration-compress` *mus fact* [Funktion]
Compress *mus* by factor *fact*, which is a `Moment`.

`ly:music-duration-length` *mus* [Funktion]
Extract the duration field from *mus* and return the length.

`ly:music-function?` *x* [Funktion]
Is *x* a `music-function`?

`ly:music-function-extract` *x* [Funktion]
Return the Scheme function inside *x*.

`ly:music-length` *mus* [Funktion]
Get the length of music expression *mus* and return it as a `Moment` object.

`ly:music-list?` *lst* [Funktion]
Is *lst* a list of music objects?

`ly:music-mutable-properties` *mus* [Funktion]
Return an alist containing the mutable properties of *mus*. The immutable properties are not available, since they are constant and initialized by the `make-music` function.

`ly:music-output?` *x* [Funktion]
Is *x* a `Music_output` object?

`ly:music-property` *mus sym val* [Funktion]
Return the value for property *sym* of music expression *mus*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:music-set-property!` *mus sym val* [Funktion]
Set property *sym* in music expression *mus* to *val*.

`ly:music-transpose` *m p* [Funktion]
Transpose *m* such that central C is mapped to *p*. Return *m*.

`ly:note-column-accidentals` *note-column* [Funktion]
Return the `AccidentalPlacement` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-column-dot-column` *note-column* [Funktion]
Return the `DotColumn` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-head::stem-attachment` *font-metric glyph-name* [Funktion]
Get attachment in *font-metric* for attaching a stem to notehead *glyph-name*.

`ly:number->string` *s* [Funktion]
Convert *s* to a string without generating many decimals.

`ly:optimal-breaking` *pb* [Funktion]
Optimally break (pages and lines) the `Paper_book` object *pb* to minimize badness in bother vertical and horizontal spacing.

`ly:option-usage` [Funktion]
Print `ly:set-option` usage.

`ly:otf->cff` *otf-file-name* [Funktion]
Convert the contents of an OTF file to a CFF file, returning it as a string.

`ly:otf-font?` *font* [Funktion]
Is *font* an OpenType font?

`ly:otf-font-glyph-info` *font glyph* [Funktion]
Given the font metric *font* of an OpenType font, return the information about named glyph *glyph* (a string).

`ly:otf-font-table-data` *font tag* [Funktion]
Extract a table *tag* from *font*. Return empty string for non-existent *tag*.

`ly:otf-glyph-count` *font* [Funktion]
Return the number of glyphs in *font*.

`ly:otf-glyph-list` *font* [Funktion]
Return a list of glyph names for *font*.

`ly:output-def?` *def* [Funktion]
Is *def* an output definition?

`ly:output-def-clone` *def* [Funktion]
Clone output definition *def*.

`ly:output-def-lookup` *def sym val* [Funktion]
Return the value of *sym* in output definition *def* (e.g., `\paper`). If no value is found, return *val* or `'()` if *val* is undefined.

`ly:output-def-parent` *def* [Funktion]
Return the parent output definition of *def*.

`ly:output-def-scope` *def* [Funktion]
Return the variable scope inside *def*.

`ly:output-def-set-variable!` *def sym val* [Funktion]
Set an output definition *def* variable *sym* to *val*.

`ly:output-description` *output-def* [Funktion]
Return the description of translators in *output-def*.

`ly:output-formats` [Funktion]
Formats passed to '`--format`' as a list of strings, used for the output.

`ly:outputter-close` *outputter* [Funktion]
Close port of *outputter*.

`ly:outputter-dump-stencil` *outputter stencil* [Funktion]
Dump stencil *expr* onto *outputter*.

`ly:outputter-dump-string` *outputter str* [Funktion]
Dump *str* onto *outputter*.

`ly:outputter-module` *outputter* [Funktion]
Return output module of *outputter*.

`ly:outputter-output-scheme` *outputter expr* [Funktion]
Eval *expr* in module of *outputter*.

`ly:outputter-port` *outputter* [Funktion]
Return output port for *outputter*.

`ly:page-marker?` *x* [Funktion]
Is *x* a `Page_marker` object?

`ly:page-turn-breaking` *pb* [Funktion]
Optimally break (pages and lines) the `Paper_book` object *pb* such that page turns only happen in specified places, returning its pages.

`ly:pango-font?` *f* [Funktion]
Is *f* a pango font?

`ly:pango-font-physical-fonts` *f* [Funktion]
Return alist of `(ps-name file-name font-index)` lists for Pango font *f*.

`ly:paper-book?` *x* [Funktion]
Is *x* a `Paper_book` object?

`ly:paper-book-header` *pb* [Funktion]
Return the header definition (`\header`) in `Paper_book` object *pb*.

`ly:paper-book-pages` *pb* [Funktion]
Return pages in `Paper_book` object *pb*.

`ly:paper-book-paper` *pb* [Funktion]
Return the paper output definition (`\paper`) in `Paper_book` object *pb*.

`ly:paper-book-performances` *pb* [Funktion]
Return performances in `Paper_book` object *pb*.

`ly:paper-book-scopes` *pb* [Funktion]
Return scopes in `Paper_book` object *pb*.

`ly:paper-book-systems` *pb* [Funktion]
Return systems in `Paper_book` object *pb*.

`ly:paper-fonts` *def* [Funktion]
Return a list containing the fonts from output definition *def* (e.g., `\paper`).

`ly:paper-get-font` *def chain* [Funktion]
Find a font metric in output definition *def* satisfying the font-qualifiers in alist chain *chain*, and return it. (An alist chain is a list of alists, containing grob properties.)

`ly:paper-get-number` *def sym* [Funktion]
Return the value of variable *sym* in output definition *def* as a double.

`ly:paper-outputscale` *def* [Funktion]
Return the output-scale for output definition *def*.

`ly:paper-score-paper-systems` *paper-score* [Funktion]
Return vector of `paper_system` objects from *paper-score*.

`ly:paper-system?` *obj* [Funktion]
Is *obj* a C++ `Prob` object of type `paper-system`?

`ly:paper-system-minimum-distance` *sys1 sys2* [Funktion]
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.

`ly:parse-file` *name* [Funktion]
Parse a single `.ly` file. Upon failure, throw `ly-file-failed` key.

`ly:parser-clear-error` *parser* [Funktion]
Clear the error flag for the parser.

`ly:parser-clone` *parser-smob* [Funktion]
Return a clone of *parser-smob*.

`ly:parser-define!` *parser-smob symbol val* [Funktion]
Bind *symbol* to *val* in *parser-smob*'s module.

`ly:parser-error` *parser msg input* [Funktion]
Display an error message and make the parser fail.

`ly:parser-has-error?` *parser* [Funktion]
Does *parser* have an error flag?

`ly:parser-include-string` *parser-smob ly-code* [Funktion]
Include the string *ly-code* into the input stream for *parser-smob*.

`ly:parser-lexer` *parser-smob* [Funktion]
Return the lexer for *parser-smob*.

`ly:parser-lookup` *parser-smob symbol* [Funktion]
Look up *symbol* in *parser-smob*'s module. Return `'()` if not defined.

`ly:parser-output-name` *parser* [Funktion]
Return the base name of the output file.

`ly:parser-parse-string` *parser-smob ly-code* [Funktion]
Parse the string *ly-code* with *parser-smob*. Upon failure, throw `ly-file-failed` key.

`ly:parser-set-note-names` *parser names* [Funktion]
Replace current note names in *parser*. *names* is an alist of symbols. This only has effect if the current mode is notes.

`ly:parser-set-repetition-function` *parser fun* [Funktion]
Replace the current repetition function in *parser*. *fun* is the new repetition function.

`ly:parser-set-repetition-symbol` *parser sym* [Funktion]
Replace the current repetition symbol in *parser*. *sym* is the new repetition symbol.

`ly:performance-write` *performance filename* [Funktion]
Write *performance* to *filename*.

`ly:pfb->pfa` *pfb-file-name* [Funktion]
Convert the contents of a Type 1 font in PFB format to PFA format.

`ly:pitch?` *x* [Funktion]
Is *x* a `Pitch` object?

`ly:pitch<?` *p1 p2* [Funktion]
Is *p1* lexicographically smaller than *p2*?

`ly:pitch-alteration` *pp* [Funktion]
Extract the alteration from pitch *pp*.

`ly:pitch-diff` *pitch root* [Funktion]
Return pitch *delta* such that *pitch* transposed by *delta* equals *root*.

`ly:pitch-negate` *p* [Funktion]
Negate *p*.

`ly:pitch-notename` *pp* [Funktion]
Extract the note name from pitch *pp*.

`ly:pitch-octave` *pp* [Funktion]
Extract the octave from pitch *pp*.

`ly:pitch-quartertones` *pp* [Funktion]
Calculate the number of quarter tones of *pp* from middle C.

`ly:pitch-semitones` *pp* [Funktion]
Calculate the number of semitones of *pp* from middle C.

`ly:pitch-steps` *p* [Funktion]
Number of steps counted from middle C of the pitch *p*.

`ly:pitch-transpose` *p delta* [Funktion]
Transpose *p* by the amount *delta*, where *delta* is relative to middle C.

`ly:pointer-group-interface::add-grob` *grob sym grob-element* [Funktion]
Add *grob-element* to *grob*'s *sym* grob array.

`ly:position-on-line?` *sg spos* [Funktion]
Return whether *spos* is on a line of the staff associated with the grob *sg* (even on an extender line).

`ly:prob?` *x* [Funktion]
Is *x* a `Prob` object?

`ly:prob-immutable-properties` *prob* [Funktion]
Retrieve an alist of immutable properties.

`ly:prob-mutable-properties` *prob* [Funktion]
Retrieve an alist of mutable properties.

`ly:prob-property` *prob sym val* [Funktion]
Return the value for property *sym* of Prob object *prob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:prob-property?` *obj sym* [Funktion]
Is boolean prop *sym* of *sym* set?

`ly:prob-set-property!` *obj sym value* [Funktion]
Set property *sym* of *obj* to *value*.

`ly:prob-type?` *obj type* [Funktion]
Is *obj* the specified prob-type?

`ly:programming-error` *str rest* [Funktion]
A Scheme callable function to issue the internal warning *str*. The message is formatted with `format` and *rest*.

`ly:progress` *str rest* [Funktion]
A Scheme callable function to print progress *str*. The message is formatted with `format` and *rest*.

`ly:property-lookup-stats` *sym* [Funktion]
Return hash table with a property access corresponding to *sym*. Choices are `prob`, `grob`, and `context`.

`ly:protects` [Funktion]
Return hash of protected objects.

`ly:pt` *num* [Funktion]
num printer points.

`ly:register-stencil-expression` *symbol* [Funktion]
Add *symbol* as head of a stencil expression.

`ly:relative-group-extent` *elements common axis* [Funktion]
Determine the extent of *elements* relative to *common* in the *axis* direction.

`ly:reset-all-fonts` [Funktion]
Forget all about previously loaded fonts.

`ly:round-filled-box` *xext yext blot* [Funktion]
Make a `Stencil` object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.

`ly:round-filled-polygon` *points blot* [Funktion]
Make a `Stencil` object that prints a black polygon with corners at the points defined by *points* (list of coordinate pairs) and roundness *blot*.

`ly:run-translator` *mus output-def* [Funktion]
Process *mus* according to *output-def*. An interpretation context is set up, and *mus* is interpreted with it. The context is returned in its final state.

Optionally, this routine takes an object-key to to uniquely identify the score block containing it.

`ly:score?` *x* [Funktion]
Is *x* a `Score` object?

`ly:score-add-output-def!` *score def* [Funktion]
Add an output definition *def* to *score*.

`ly:score-embedded-format` *score layout* [Funktion]
Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

`ly:score-error?` *score* [Funktion]
Was there an error in the score?

`ly:score-header` *score* [Funktion]
Return score header.

`ly:score-music` *score* [Funktion]
Return score music.

`ly:score-output-defs` *score* [Funktion]
All output definitions in a score.

`ly:score-set-header!` *score module* [Funktion]
Set the score header.

`ly:set-default-scale` *scale* [Funktion]
Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.

`ly:set-grob-modification-callback` *cb* [Funktion]
Specify a procedure that will be called every time LilyPond modifies a grob property. The callback will receive as arguments the grob that is being modified, the name of the C++ file in which the modification was requested, the line number in the C++ file in which the modification was requested, the name of the function in which the modification was requested, the property to be changed, and the new value for the property.

`ly:set-middle-C!` *context* [Funktion]
Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and middleCOffset.

`ly:set-option` *var val* [Funktion]
Set a program option.

`ly:set-property-cache-callback` *cb* [Funktion]
Specify a procedure that will be called whenever lilypond calculates a callback function and caches the result. The callback will receive as arguments the grob whose property it is, the name of the property, the name of the callback that calculated the property, and the new (cached) value of the property.

`ly:simple-closure?` *clos* [Funktion]
Is *clos* a simple closure?

`ly:skyline?` *x* [Funktion]
Is *x* a `Skyline` object?

`ly:skyline-pair?` *x* [Funktion]
Is *x* a `Skyline_pair` object?

`ly:slur-score-count` [Funktion]
count number of slur scores.

`ly:smob-protects` [Funktion]
Return LilyPond's internal smob protection list.

`ly:solve-spring-rod-problem` *springs rods length ragged* [Funktion]
Solve a spring and rod problem for *count* objects, that are connected by *count*-1 *springs*, and an arbitrary number of *rods*. *count* is implicitly given by *springs* and *rods*. The *springs*

argument has the format `(ideal, inverse_hook)` and *rods* is of the form `(idx1, idx2, distance)`.

length is a number, *ragged* a boolean.

The function returns a list containing the force (positive for stretching, negative for compressing and `#f` for non-satisfied constraints) followed by *spring-count*+1 positions of the objects.

`ly:source-file?` *x* [Funktion]
Is *x* a `Source_file` object?

`ly:spanner?` *g* [Funktion]
Is *g* a spanner object?

`ly:spanner-bound` *spanner dir* [Funktion]
Get one of the bounds of *spanner*. *dir* is `-1` for left, and `1` for right.

`ly:spanner-broken-into` *spanner* [Funktion]
Return broken-into list for *spanner*.

`ly:spanner-set-bound!` *spanner dir item* [Funktion]
Set grob *item* as bound in direction *dir* for *spanner*.

`ly:spawn` *command rest* [Funktion]
Simple interface to g_spawn_sync *str*. The error is formatted with `format` and *rest*.

`ly:staff-symbol-line-thickness` *grob* [Funktion]
Returns the `line-thickness` of the staff associated with *grob*.

`ly:staff-symbol-staff-space` *grob* [Funktion]
Returns the `staff-space` of the staff associated with *grob*.

`ly:start-environment` [Funktion]
Return the environment (a list of strings) that was in effect at program start.

`ly:stderr-redirect` *file-name mode* [Funktion]
Redirect stderr to *file-name*, opened with *mode*.

`ly:stencil?` *x* [Funktion]
Is *x* a `Stencil` object?

`ly:stencil-add` *args* [Funktion]
Combine stencils. Takes any number of arguments.

`ly:stencil-aligned-to` *stil axis dir* [Funktion]
Align *stil* using its own extents. *dir* is a number. `-1` and `1` are left and right, respectively. Other values are interpolated (so `0` means the center).

`ly:stencil-combine-at-edge` *first axis direction second padding minimum* [Funktion]
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. If this puts the reference points closer than *minimum*, they are moved by the latter amount. *first* and *second* may also be `'()` or `#f`.

`ly:stencil-empty?` *stil* [Funktion]
Return whether *stil* is empty.

`ly:stencil-expr` *stil* [Funktion]
Return the expression of *stil*.

`ly:stencil-extent` *stil axis* [Funktion]
Return a pair of numbers signifying the extent of *stil* in *axis* direction (`0` or `1` for x and y axis, respectively).

`ly:stencil-fonts` *s* [Funktion]
Analyze *s*, and return a list of fonts used in *s*.

`ly:stencil-in-color` *stc r g b* [Funktion]
Put *stc* in a different color.

`ly:stencil-rotate` *stil angle x y* [Funktion]
Return a stencil *stil* rotated *angle* degrees around the relative offset (*x*, *y*). E.g. an offset of (-1, 1) will rotate the stencil around the left upper corner.

`ly:stencil-rotate-absolute` *stil angle x y* [Funktion]
Return a stencil *stil* rotated *angle* degrees around point (*x*, *y*), given in absolute coordinates.

`ly:stencil-scale` *stil x y* [Funktion]
Scale *stil* using the horizontal and vertical scaling factors *x* and *y*.

`ly:stencil-translate` *stil offset* [Funktion]
Return a *stil*, but translated by *offset* (a pair of numbers).

`ly:stencil-translate-axis` *stil amount axis* [Funktion]
Return a copy of *stil* but translated by *amount* in *axis* direction.

`ly:stream-event?` *obj* [Funktion]
Is *obj* a `Stream_event` object?

`ly:string-percent-encode` *str* [Funktion]
Encode all characters in string *str* with hexadecimal percent escape sequences, with the following exceptions: characters `-`, `.`, `/`, and `_`; and characters in ranges `0-9`, `A-Z`, and `a-z`.

`ly:string-substitute` *a b s* [Funktion]
Replace string *a* by string *b* in string *s*.

`ly:success` *str rest* [Funktion]
A Scheme callable function to issue a success message *str*. The message is formatted with `format` and *rest*.

`ly:system-font-load` *name* [Funktion]
Load the OpenType system font '*name*`.otf`'. Fonts loaded with this command must contain three additional SFNT font tables called `LILC`, `LILF`, and `LILY`, needed for typesetting musical elements. Currently, only the Emmentaler and the Emmentaler-Brace fonts fulfill these requirements.

Note that only `ly:font-get-glyph` and derived code (like `\lookup`) can access glyphs from the system fonts; text strings are handled exclusively via the Pango interface.

`ly:text-interface::interpret-markup` [Funktion]
Convert a text markup into a stencil. Takes three arguments, *layout*, *props*, and *markup*.

layout is a `\layout` block; it may be obtained from a grob with `ly:grob-layout`. *props* is an alist chain, i.e. a list of alists. This is typically obtained with `(ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults))`. *markup* is the markup text to be processed.

`ly:translator?` *x* [Funktion]
Is *x* a `Translator` object?

`ly:translator-context` *trans* [Funktion]
Return the context of the translator object *trans*.

`ly:translator-description` *me* [Funktion]
Return an alist of properties of translator *me*.

`ly:translator-group?` *x* [Funktion]
Is *x* a `Translator_group` object?

`ly:translator-name` *trans* [Funktion]
Return the type name of the translator object *trans*. The name is a symbol.

`ly:transpose-key-alist` *l pit* [Funktion]
Make a new key alist of *l* transposed by pitch *pit*.

`ly:truncate-list!` *lst i* [Funktion]
Take at most the first *i* of list *lst*.

`ly:ttf->pfa` *ttf-file-name idx* [Funktion]
Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:ttf-ps-name` *ttf-file-name idx* [Funktion]
Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:unit` [Funktion]
Return the unit used for lengths as a string.
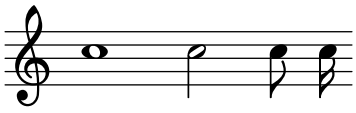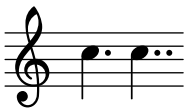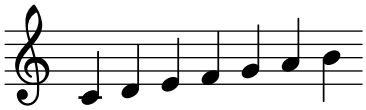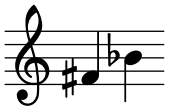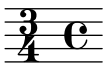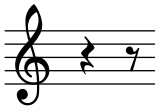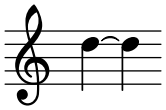
`ly:usage` [Funktion]
Print usage message.
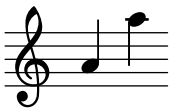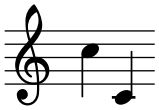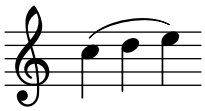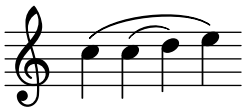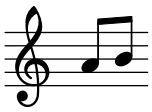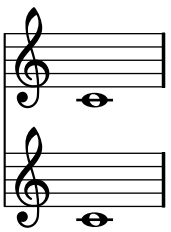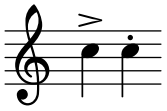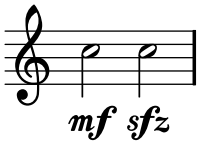
`ly:version` [Funktion]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.
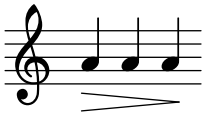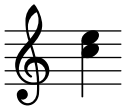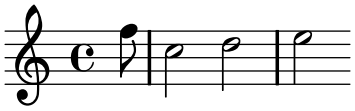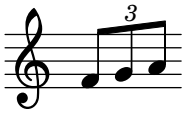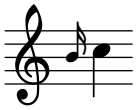
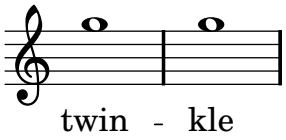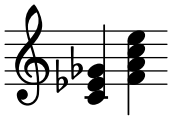
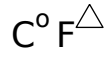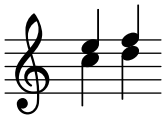
`ly:warning` *str rest* [Funktion]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.

`ly:wide-char->utf-8` *wc* [Funktion]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.

Anhang B Befehlsübersicht

Syntax	Erklärung	Beispiel
`1 2 8 16`	Tondauern	
`c4. c4..`	Punktierung	
`c d e f g a b`	Tonleiter	
`fis bes`	Vorzeichen	
`\clef treble \clef bass`	Notenschlüssel	
`\time 3/4 \time 4/4`	Taktangaben	
`r4 r8`	Pause	
`d ~ d`	Bindebogen	
`\key es \major`	Tonart	

note`'`	Oktavierung	
note`,`	Oktavierung nach unten	
`c( d e)`	Legatobogen	
`c\( c( d) e\)`	Phrasierungsbogen	
`a8[ b]`	Balken	
`<< \new Staff ... >>`	mehr Notensysteme	
`c-> c-.`	Artikulationszeichen	
`c2\mf c\sfz`	Dynamik	
`a\< a a\!`	Crescendo	

`a\> a a\!`	Decrescendo	
`< >`	Noten im Akkord	
`\partial 8`	Auftakt	
`\times 2/3 {f g a}`	Triolen	
`\grace`	Verzierungen	
`\lyricmode { twinkle }`	Texteingabe	twinkle
`\new Lyrics`	Gesangstext	twinkle
`twin -- kle`	Gesangstext-Trennstrich	
`\chordmode { c:dim f:maj7 }`	Akkorde	
`\context ChordNames`	Akkordsymbole drucken	
`<<{e f} \\ {c d}>>`	Mehrstimmigkeit	

`s4 s8 s16`	unsichtbare Pausen

Anhang C LilyPond-Grammatik

Dieser Anhang enthält eine Beschreibung der LilyPond-Grammatik, wie sie der Parser ausgibt.

`Grammar`

```
lilypond: /* empty */
        | lilypond toplevel_expression
        | lilypond assignment
        | lilypond error
        | lilypond "\invalid"

toplevel_expression: lilypond_header
                   | book_block
                   | bookpart_block
                   | score_block
                   | composite_music
                   | full_markup
                   | full_markup_list
                   | output_def

embedded_scm: SCM_TOKEN
            | SCM_IDENTIFIER

lilypond_header_body: /* empty */
                    | lilypond_header_body assignment

lilypond_header: "\header" '{' lilypond_header_body '}'

assignment_id: STRING
             | LYRICS_STRING

assignment: assignment_id '=' identifier_init
          | assignment_id property_path '=' identifier_init
          | embedded_scm

identifier_init: score_block
               | book_block
               | bookpart_block
               | output_def
               | context_def_spec_block
               | music
               | post_event
               | number_expression
               | string
               | embedded_scm
               | full_markup
               | full_markup_list
               | DIGIT
               | context_modification

context_def_spec_block: "\context" '{' context_def_spec_body '}'
```

```
39 context_def_spec_body: /* empty */
40                      | CONTEXT_DEF_IDENTIFIER
41                      | context_def_spec_body
                            "\grobdescriptions"
                            embedded_scm
42                      | context_def_spec_body context_mod
43                      | context_def_spec_body context_modification

44 book_block: "\book" '{' book_body '}'

45 book_body: /* empty */
46          | BOOK_IDENTIFIER
47          | book_body paper_block
48          | book_body bookpart_block
49          | book_body score_block
50          | book_body composite_music
51          | book_body full_markup
52          | book_body full_markup_list
53          | book_body lilypond_header
54          | book_body error

55 bookpart_block: "\bookpart" '{' bookpart_body '}'

56 bookpart_body: /* empty */
57              | BOOK_IDENTIFIER
58              | bookpart_body paper_block
59              | bookpart_body score_block
60              | bookpart_body composite_music
61              | bookpart_body full_markup
62              | bookpart_body full_markup_list
63              | bookpart_body lilypond_header
64              | bookpart_body error

65 score_block: "\score" '{' score_body '}'

66 score_body: music
67           | SCORE_IDENTIFIER
68           | score_body lilypond_header
69           | score_body output_def
70           | score_body error

71 paper_block: output_def

72 output_def: output_def_body '}'

73 output_def_head: "\paper"
74                | "\midi"
75                | "\layout"

76 output_def_head_with_mode_switch: output_def_head

77 output_def_body: output_def_head_with_mode_switch '{'
```

```
 78                | output_def_head_with_mode_switch
                       '{'
                       OUTPUT_DEF_IDENTIFIER
 79                | output_def_body assignment
 80                | output_def_body context_def_spec_block
 81                | output_def_body error

 82 tempo_event: "\tempo" steno_duration '=' tempo_range
 83            | "\tempo" scalar steno_duration '=' tempo_range
 84            | "\tempo" scalar

 85 music_list: /* empty */
 86           | music_list music
 87           | music_list embedded_scm
 88           | music_list error

 89 music: simple_music
 90      | composite_music

 91 alternative_music: /* empty */
 92                  | "\alternative" '{' music_list '}'

 93 repeated_music: "\repeat"
                     simple_string
                     unsigned_number
                     music
                     alternative_music

 94 sequential_music: "\sequential" '{' music_list '}'
 95                 | '{' music_list '}'

 96 simultaneous_music: "\simultaneous" '{' music_list '}'
 97                   | "<<" music_list ">>"

 98 simple_music: event_chord
 99             | MUSIC_IDENTIFIER
100             | music_property_def
101             | context_change

103 context_modification: "\with"  '{' context_mod_list '}'
104                     | "\with" CONTEXT_MOD_IDENTIFIER
105                     | CONTEXT_MOD_IDENTIFIER

106 optional_context_mod: /* empty */
107                     | context_modification

108 context_mod_list: /* empty */
109                 | context_mod_list context_mod
110                 | context_mod_list CONTEXT_MOD_IDENTIFIER

111 composite_music: prefix_composite_music
112                | grouped_music_list
```

```
113 grouped_music_list: simultaneous_music
114                   | sequential_music

115 function_scm_argument: embedded_scm
116                      | simple_string

117 function_arglist_music_last: EXPECT_MUSIC function_arglist music

118 function_arglist_nonmusic_last: EXPECT_MARKUP
                                      function_arglist
                                      full_markup
119                                 | EXPECT_MARKUP
                                      function_arglist
                                      simple_string
120                                 | EXPECT_SCM
                                      function_arglist
                                      function_scm_argument

121 function_arglist_nonmusic: EXPECT_NO_MORE_ARGS
122                            | EXPECT_MARKUP
                                 function_arglist_nonmusic
                                 full_markup
123                            | EXPECT_MARKUP
                                 function_arglist_nonmusic
                                 simple_string
124                            | EXPECT_SCM
                                 function_arglist_nonmusic
                                 function_scm_argument

125 function_arglist: EXPECT_NO_MORE_ARGS
126                 | function_arglist_music_last
127                 | function_arglist_nonmusic_last

128 generic_prefix_music_scm: MUSIC_FUNCTION function_arglist

129 optional_id: /* empty */
130            | '=' simple_string

131 prefix_composite_music: generic_prefix_music_scm
132                       | "\context"
                            simple_string
                            optional_id
                            optional_context_mod
                            music
133                       | "\new"
                            simple_string
                            optional_id
                            optional_context_mod
                            music
134                       | "\times" fraction music
135                       | repeated_music
```

```
136                       | "\transpose"
                              pitch_also_in_chords
                              pitch_also_in_chords
                              music
137                       | mode_changing_head grouped_music_list
138                       | mode_changing_head_with_context
                              optional_context_mod
                              grouped_music_list
139                       | relative_music
140                       | re_rhythmed_music

141 mode_changing_head: "\notemode"
142                   | "\drummode"
143                   | "\figuremode"
144                   | "\chordmode"
145                   | "\lyricmode"

146 mode_changing_head_with_context: "\drums"
147                                 | "\figures"
148                                 | "\chords"
149                                 | "\lyrics"

150 relative_music: "\relative" absolute_pitch music
151               | "\relative" composite_music

153 new_lyrics: "\addlyrics"  grouped_music_list

155 new_lyrics: "\addlyrics"  MUSIC_IDENTIFIER

157 new_lyrics: new_lyrics "\addlyrics"  grouped_music_list

159 new_lyrics: new_lyrics "\addlyrics"  MUSIC_IDENTIFIER

160 re_rhythmed_music: grouped_music_list new_lyrics
161                  | MUSIC_IDENTIFIER new_lyrics

163 re_rhythmed_music: "\lyricsto" simple_string  music

164 context_change: "\change" STRING '=' STRING

165 property_path_revved: embedded_scm
166                     | property_path_revved embedded_scm

167 property_path: property_path_revved

168 property_operation: STRING '=' scalar
169                   | "\unset" simple_string
170                   | "\override" simple_string property_path '=' scalar
171                   | "\revert" simple_string embedded_scm

172 context_def_mod: "\consists"
173               | "\remove"
```

```
174                | "\accepts"
175                | "\defaultchild"
176                | "\denies"
177                | "\alias"
178                | "\type"
179                | "\description"
180                | "\name"

181 context_mod: property_operation
182             | context_def_mod STRING
183             | context_def_mod embedded_scm

184 context_prop_spec: simple_string
185                   | simple_string '.' simple_string

186 simple_music_property_def: "\override"
                                context_prop_spec
                                property_path
                                '='
                                scalar
187                          | "\revert" context_prop_spec embedded_scm
188                          | "\set" context_prop_spec '=' scalar
189                          | "\unset" context_prop_spec

190 music_property_def: simple_music_property_def
191                    | "\once" simple_music_property_def

192 string: STRING
193       | STRING_IDENTIFIER
194       | string '+' string

195 simple_string: STRING
196               | LYRICS_STRING
197               | STRING_IDENTIFIER

198 scalar: string
199       | LYRICS_STRING
200       | bare_number
201       | embedded_scm
202       | full_markup
203       | DIGIT

204 event_chord: simple_chord_elements post_events
205             | CHORD_REPETITION optional_notemode_duration post_events
206             | MULTI_MEASURE_REST optional_notemode_duration post_events
207             | command_element
208             | note_chord_element

209 note_chord_element: chord_body optional_notemode_duration post_events

210 chord_body: "<" chord_body_elements ">"
```

```
211 chord_body_elements: /* empty */
212                    | chord_body_elements chord_body_element

213 chord_body_element: pitch
                          exclamations
                          questions
                          octave_check
                          post_events
214                   | DRUM_PITCH post_events
215                   | music_function_chord_body

216 music_function_identifier_musicless_prefix: MUSIC_FUNCTION

217 music_function_chord_body: music_function_identifier_musicless_prefix
                                 EXPECT_MUSIC
                                 function_arglist_nonmusic
                                 chord_body_element
218                           | music_function_identifier_musicless_prefix
                                 function_arglist_nonmusic

219 music_function_event: music_function_identifier_musicless_prefix
                            EXPECT_MUSIC
                            function_arglist_nonmusic
                            post_event
220                     | music_function_identifier_musicless_prefix
                            function_arglist_nonmusic

221 command_element: command_event
222                | "\skip" duration_length
223                | "\["
224                | "\]"
225                | "\"
226                | '|'
227                | "\partial" duration_length
228                | "\time" fraction
229                | "\mark" scalar

230 command_event: "\~"
231              | "\mark" "\default"
232              | tempo_event
233              | "\key" "\default"
234              | "\key" NOTENAME_PITCH SCM_IDENTIFIER

235 post_events: /* empty */
236            | post_events post_event

237 post_event: direction_less_event
238           | script_dir music_function_event
239           | "--"
240           | "__"
241           | script_dir direction_reqd_event
242           | script_dir direction_less_event
```

```
243           | string_number_event

244 string_number_event: E_UNSIGNED

245 direction_less_char: '['
246                    | ']'
247                    | '~'
248                    | '('
249                    | ')'
250                    | "\!"
251                    | "\("
252                    | "\)"
253                    | "\>"
254                    | "\<"

255 direction_less_event: direction_less_char
256                     | EVENT_IDENTIFIER
257                     | tremolo_type

258 direction_reqd_event: gen_text_def
259                     | script_abbreviation

260 octave_check: /* empty */
261             | '='
262             | '=' sub_quotes
263             | '=' sup_quotes

264 sup_quotes: '''
265           | sup_quotes '''

266 sub_quotes: ','
267           | sub_quotes ','

268 steno_pitch: NOTENAME_PITCH
269            | NOTENAME_PITCH sup_quotes
270            | NOTENAME_PITCH sub_quotes

271 steno_tonic_pitch: TONICNAME_PITCH
272                  | TONICNAME_PITCH sup_quotes
273                  | TONICNAME_PITCH sub_quotes

274 pitch: steno_pitch

275 pitch_also_in_chords: pitch
276                     | steno_tonic_pitch

277 gen_text_def: full_markup
278             | string
279             | DIGIT

280 script_abbreviation: '^'
281                    | '+'
```

```
282                     | '-'
283                     | '|'
284                     | ">"
285                     | '.'
286                     | '_'

287 script_dir: '_'
288           | '^'
289           | '-'

290 absolute_pitch: steno_pitch

291 duration_length: multiplied_duration

292 optional_notemode_duration: /* empty */
293                           | multiplied_duration

294 steno_duration: bare_unsigned dots
295               | DURATION_IDENTIFIER dots

296 multiplied_duration: steno_duration
297                     | multiplied_duration '*' bare_unsigned
298                     | multiplied_duration '*' FRACTION

299 fraction: FRACTION
300         | UNSIGNED '/' UNSIGNED

301 dots: /* empty */
302     | dots '.'

303 tremolo_type: ':'
304             | ':' bare_unsigned

305 bass_number: DIGIT
306            | UNSIGNED
307            | STRING
308            | full_markup

309 figured_bass_alteration: '-'
310                        | '+'
311                        | '!'

312 bass_figure: "_"
313            | bass_number
314            | bass_figure ']'
315            | bass_figure figured_bass_alteration
316            | bass_figure figured_bass_modification

317 figured_bass_modification: "\+"
318                           | "\!"
319                           | '/'
320                           | "\"
```

```
321 br_bass_figure: bass_figure
322               | '[' bass_figure

323 figure_list: /* empty */
324            | figure_list br_bass_figure

325 figure_spec: FIGURE_OPEN figure_list FIGURE_CLOSE

326 optional_rest: /* empty */
327              | "\rest"

328 simple_element: pitch
                     exclamations
                     questions
                     octave_check
                     optional_notemode_duration
                     optional_rest
329               | DRUM_PITCH optional_notemode_duration
330               | RESTNAME optional_notemode_duration
331               | lyric_element optional_notemode_duration

332 simple_chord_elements: simple_element
333                       | new_chord
334                       | figure_spec optional_notemode_duration

335 lyric_element: lyric_markup
336              | LYRICS_STRING

337 new_chord: steno_tonic_pitch optional_notemode_duration
338          | steno_tonic_pitch
               optional_notemode_duration
               chord_separator
               chord_items

339 chord_items: /* empty */
340            | chord_items chord_item

341 chord_separator: ":"
342                | "^"
343                | "/" steno_tonic_pitch
344                | "/+" steno_tonic_pitch

345 chord_item: chord_separator
346           | step_numbers
347           | CHORD_MODIFIER

348 step_numbers: step_number
349             | step_numbers '.' step_number

350 step_number: bare_unsigned
351            | bare_unsigned '+'
```

```
352             | bare_unsigned "-"

353 tempo_range: bare_unsigned
354             | bare_unsigned '~' bare_unsigned

355 number_expression: number_expression '+' number_term
356                  | number_expression '-' number_term
357                  | number_term

358 number_term: number_factor
359             | number_factor '*' number_factor
360             | number_factor '/' number_factor

361 number_factor: '-' number_factor
362               | bare_number

363 bare_number: UNSIGNED
364             | REAL
365             | NUMBER_IDENTIFIER
366             | REAL NUMBER_IDENTIFIER
367             | UNSIGNED NUMBER_IDENTIFIER

368 bare_unsigned: UNSIGNED
369               | DIGIT

370 unsigned_number: bare_unsigned
371                 | NUMBER_IDENTIFIER

372 exclamations: /* empty */
373              | exclamations '!'

374 questions: /* empty */
375           | questions '?'

376 lyric_markup: LYRIC_MARKUP_IDENTIFIER

378 lyric_markup: LYRIC_MARKUP  markup_top

379 full_markup_list: MARKUPLINES_IDENTIFIER

381 full_markup_list: "\markuplines"  markup_list

382 full_markup: MARKUP_IDENTIFIER

384 full_markup: "\markup"  markup_top

385 markup_top: markup_list
386           | markup_head_1_list simple_markup
387           | simple_markup

388 markup_list: MARKUPLINES_IDENTIFIER
389             | markup_composed_list
```

```
  390             | markup_braced_list
  391             | markup_command_list

  392 markup_composed_list: markup_head_1_list markup_braced_list

  393 markup_braced_list: '{' markup_braced_list_body '}'

  394 markup_braced_list_body: /* empty */
  395                        | markup_braced_list_body markup
  396                        | markup_braced_list_body markup_list

  397 markup_command_list: MARKUP_LIST_FUNCTION markup_command_list_arguments

  398 markup_command_basic_arguments: EXPECT_MARKUP_LIST
                                        markup_command_list_arguments
                                        markup_list
  399                               | EXPECT_SCM
                                        markup_command_list_arguments
                                        embedded_scm
  400                               | EXPECT_NO_MORE_ARGS

  401 markup_command_list_arguments: markup_command_basic_arguments
  402                              | EXPECT_MARKUP
                                       markup_command_list_arguments
                                       markup

  403 markup_head_1_item: MARKUP_FUNCTION
                            EXPECT_MARKUP
                            markup_command_list_arguments

  404 markup_head_1_list: markup_head_1_item
  405                   | markup_head_1_list markup_head_1_item

  406 simple_markup: STRING
  407              | MARKUP_IDENTIFIER
  408              | LYRIC_MARKUP_IDENTIFIER
  409              | STRING_IDENTIFIER

  411 simple_markup: "\score" 0 '{' score_body '}'
  412              | MARKUP_FUNCTION markup_command_basic_arguments

  413 markup: markup_head_1_list simple_markup
  414       | simple_markup


Terminals, with rules where they appear



"-" (319) 352
"--" (340) 239
"/" (320) 343
```

"/+" (316) 344
":" (318) 341
"<" (321) 210
"<<" (323) 97
">" (322) 210 284
">>" (324) 97
"\!" (329) 250 318
"\" (325) 225 320
"\(" (331) 251
"\)" (328) 252
"\+" (334) 317
"\<" (333) 254
"\>" (326) 253
"\[" (330) 223
"\]" (332) 224
"\accepts" (261) 174
"\addlyrics" (259) 153 155 157 159
"\alias" (262) 177
"\alternative" (263) 92
"\book" (264) 44
"\bookpart" (265) 55
"\C[haracter]" (327)
"\change" (266) 164
"\chordmode" (267) 144
"\chords" (268) 148
"\consists" (269) 172
"\context" (270) 38 132
"\default" (271) 231 233
"\defaultchild" (272) 175
"\denies" (273) 176
"\description" (274) 179
"\drummode" (275) 142
"\drums" (276) 146
"\figuremode" (277) 143
"\figures" (278) 147
"\grobdescriptions" (279) 41
"\header" (280) 18
"\invalid" (281) 5
"\key" (282) 233 234
"\layout" (283) 75
"\lyricmode" (284) 145
"\lyrics" (285) 149
"\lyricsto" (286) 163
"\mark" (287) 229 231
"\markup" (288) 384
"\markuplines" (289) 381
"\midi" (290) 74
"\name" (291) 180
"\new" (315) 133
"\notemode" (292) 141
"\octave" (293)
"\once" (294) 191

```
"\override" (295) 170 186
"\paper" (296) 73
"\partial" (297) 227
"\relative" (298) 150 151
"\remove" (299) 173
"\repeat" (300) 93
"\rest" (301) 327
"\revert" (302) 171 187
"\score" (303) 65 411
"\sequential" (304) 94
"\set" (305) 188
"\simultaneous" (306) 96
"\skip" (307) 222
"\tempo" (308) 82 83 84
"\time" (314) 228
"\times" (309) 134
"\transpose" (310) 136
"\type" (311) 178
"\unset" (312) 169 189
"\with" (313) 103 104
"\~" (335) 230
"^" (317) 342
"_" (339) 312
"__" (336) 240
$end (0) 0
'!' (33) 311 373
''' (39) 264 265
'(' (40) 248
')' (41) 249
'*' (42) 297 298 359
'+' (43) 194 281 310 351 355
',' (44) 266 267
'-' (45) 282 289 309 356 361
'.' (46) 185 285 302 349
'/' (47) 300 319 360
':' (58) 303 304
'=' (61) 21 22 82 83 130 164 168 170 186 188 261 262 263
'?' (63) 375
'[' (91) 245 322
']' (93) 246 314
'^' (94) 280 288
'_' (95) 286 287
'{' (123) 18 38 44 55 65 77 78 92 94 95 96 103 393 411
'|' (124) 226 283
'}' (125) 18 38 44 55 65 72 92 94 95 96 103 393 411
'~' (126) 247 354
BOOK_IDENTIFIER (352) 46 57
CHORD_MODIFIER (354) 347
CHORD_REPETITION (355) 205
CHORDMODIFIER_PITCH (353)
CHORDMODIFIERS (341)
CONTEXT_DEF_IDENTIFIER (356) 40
```

```
CONTEXT_MOD_IDENTIFIER (357) 104 105 110
DIGIT (344) 36 203 279 305 369
DRUM_PITCH (358) 214 329
DURATION_IDENTIFIER (359) 295
E_UNSIGNED (345) 244
error (256) 4 54 64 70 81 88
EVENT_IDENTIFIER (360) 256
EXPECT_MARKUP (347) 118 119 122 123 402 403
EXPECT_MARKUP_LIST (350) 398
EXPECT_MUSIC (348) 117 217 219
EXPECT_NO_MORE_ARGS (351) 121 125 400
EXPECT_SCM (349) 120 124 399
FIGURE_CLOSE (337) 325
FIGURE_OPEN (338) 325
FRACTION (361) 298 299
LYRIC_MARKUP (342) 378
LYRIC_MARKUP_IDENTIFIER (363) 376 408
LYRICS_STRING (362) 20 196 199 336
MARKUP_FUNCTION (364) 403 412
MARKUP_IDENTIFIER (366) 382 407
MARKUP_LIST_FUNCTION (365) 397
MARKUPLINES_IDENTIFIER (367) 379 388
MULTI_MEASURE_REST (343) 206
MUSIC_FUNCTION (368) 128 216
MUSIC_IDENTIFIER (369) 99 155 159 161
NOTENAME_PITCH (370) 234 268 269 270
NUMBER_IDENTIFIER (371) 365 366 367 371
OUTPUT_DEF_IDENTIFIER (372) 78
PREC_BOT (260)
PREC_TOP (258)
REAL (373) 364 366
RESTNAME (374) 330
SCM_IDENTIFIER (375) 15 234
SCM_TOKEN (376) 14
SCORE_IDENTIFIER (377) 67
STRING (378) 19 164 168 182 192 195 307 406
STRING_IDENTIFIER (379) 193 197 409
TONICNAME_PITCH (380) 271 272 273
UNARY_MINUS (381)
UNSIGNED (346) 300 306 363 367 368

Nonterminals, with rules where they appear


absolute_pitch (240)
    on left: 290, on right: 150
alternative_music (173)
    on left: 91 92, on right: 93
assignment (155)
    on left: 21 22 23, on right: 3 17 79
assignment_id (154)
    on left: 19 20, on right: 21 22
```

```
bare_number (269)
    on left: 363 364 365 366 367, on right: 200 362
bass_number (248)
    on left: 305 306 307 308, on right: 313
book_block (159)
    on left: 44, on right: 7 25
bookpart_block (161)
    on left: 55, on right: 8 26 48
br_bass_figure (252)
    on left: 321 322, on right: 324
chord_body (216)
    on left: 210, on right: 209
chord_body_element (218)
    on left: 213 214 215, on right: 212 217
chord_body_elements (217)
    on left: 211 212, on right: 210 212
chord_item (262)
    on left: 345 346 347, on right: 340
chord_items (260)
    on left: 339 340, on right: 338 340
chord_separator (261)
    on left: 341 342 343 344, on right: 338 345
command_event (223)
    on left: 230 231 232 233 234, on right: 221
composite_music (182)
    on left: 111 112, on right: 10 50 60 90 151
context_change (202)
    on left: 164, on right: 101
context_def_spec_block (157)
    on left: 38, on right: 28 80
context_mod (207)
    on left: 181 182 183, on right: 42 109
context_mod_list (181)
    on left: 108 109 110, on right: 103 109 110
context_modification (178)
    on left: 103 104 105, on right: 37 43 107
context_prop_spec (208)
    on left: 184 185, on right: 186 187 188 189
direction_less_event (228)
    on left: 255 256 257, on right: 237 242
direction_reqd_event (229)
    on left: 258 259, on right: 241
dots (246)
    on left: 301 302, on right: 294 295 302
duration_length (241)
    on left: 291, on right: 222 227
event_chord (214)
    on left: 204 205 206 207 208, on right: 98
exclamations (272)
    on left: 372 373, on right: 213 328 373
figure_list (253)
    on left: 323 324, on right: 324 325
```

```
figure_spec (254)
    on left: 325, on right: 334
figured_bass_alteration (249)
    on left: 309 310 311, on right: 315
figured_bass_modification (251)
    on left: 317 318 319 320, on right: 316
fraction (245)
    on left: 299 300, on right: 134 228
full_markup_list (276)
    on left: 379 381, on right: 12 35 52 62
function_arglist_music_last (185)
    on left: 117, on right: 126
function_arglist_nonmusic_last (186)
    on left: 118 119 120, on right: 127
function_scm_argument (184)
    on left: 115 116, on right: 120 124
gen_text_def (237)
    on left: 277 278 279, on right: 258
generic_prefix_music_scm (189)
    on left: 128, on right: 131
lilypond (149)
    on left: 1 2 3 4 5, on right: 0 2 3 4 5
lilypond_header (153)
    on left: 18, on right: 6 53 63 68
lilypond_header_body (152)
    on left: 16 17, on right: 17 18
lyric_element (258)
    on left: 335 336, on right: 331
lyric_markup (274)
    on left: 376 378, on right: 335
markup (292)
    on left: 413 414, on right: 395 402
markup_braced_list (283)
    on left: 393, on right: 390 392
markup_braced_list_body (284)
    on left: 394 395 396, on right: 393 395 396
markup_command_list (285)
    on left: 397, on right: 391
markup_composed_list (282)
    on left: 392, on right: 389
markup_head_1_item (288)
    on left: 403, on right: 404 405
markup_head_1_list (289)
    on left: 404 405, on right: 386 392 405 413
markup_list (281)
    on left: 388 389 390 391, on right: 381 385 396 398
markup_top (280)
    on left: 385 386 387, on right: 378 384
mode_changing_head (192)
    on left: 141 142 143 144 145, on right: 137
multiplied_duration (244)
    on left: 296 297 298, on right: 291 293 297 298
```

```
music_function_chord_body (220)
    on left: 217 218, on right: 215
music_function_event (221)
    on left: 219 220, on right: 238
music_list (171)
    on left: 85 86 87 88, on right: 86 87 88 92 94 95 96 97
music_property_def (210)
    on left: 190 191, on right: 100
new_chord (259)
    on left: 337 338, on right: 333
new_lyrics (195)
    on left: 153 155 157 159, on right: 157 159 160 161
note_chord_element (215)
    on left: 209, on right: 208
number_expression (266)
    on left: 355 356 357, on right: 31 355 356
number_factor (268)
    on left: 361 362, on right: 358 359 360 361
number_term (267)
    on left: 358 359 360, on right: 355 356 357
octave_check (230)
    on left: 260 261 262 263, on right: 213 328
optional_context_mod (180)
    on left: 106 107, on right: 132 133 138
optional_id (190)
    on left: 129 130, on right: 132 133
optional_rest (255)
    on left: 326 327, on right: 328
output_def (166)
    on left: 72, on right: 13 27 69 71
output_def_body (169)
    on left: 77 78 79 80 81, on right: 72 79 80 81
output_def_head (167)
    on left: 73 74 75, on right: 76
output_def_head_with_mode_switch (168)
    on left: 76, on right: 77 78
paper_block (165)
    on left: 71, on right: 47 58
pitch (235)
    on left: 274, on right: 213 275 328
pitch_also_in_chords (236)
    on left: 275 276, on right: 136
post_events (224)
    on left: 235 236, on right: 204 205 206 209 213 214 236
property_operation (205)
    on left: 168 169 170 171, on right: 181
property_path (204)
    on left: 167, on right: 22 170 186
property_path_revved (203)
    on left: 165 166, on right: 166 167
questions (273)
    on left: 374 375, on right: 213 328 375
```

```
re_rhythmed_music (200)
    on left: 160 161 163, on right: 140
relative_music (194)
    on left: 150 151, on right: 139
repeated_music (174)
    on left: 93, on right: 135
score_block (163)
    on left: 65, on right: 9 24 49 59
score_body (164)
    on left: 66 67 68 69 70, on right: 65 68 69 70 411
script_dir (239)
    on left: 287 288 289, on right: 238 241 242
sequential_music (175)
    on left: 94 95, on right: 114
simple_chord_elements (257)
    on left: 332 333 334, on right: 204
simple_element (256)
    on left: 328 329 330 331, on right: 332
simple_music (177)
    on left: 98 99 100 101, on right: 89
simultaneous_music (176)
    on left: 96 97, on right: 113
steno_duration (243)
    on left: 294 295, on right: 82 83 296
steno_pitch (233)
    on left: 268 269 270, on right: 274 290
step_number (264)
    on left: 350 351 352, on right: 348 349
step_numbers (263)
    on left: 348 349, on right: 346 349
string (211)
    on left: 192 193 194, on right: 32 194 198 278
string_number_event (226)
    on left: 244, on right: 243
sub_quotes (232)
    on left: 266 267, on right: 262 267 270 273
sup_quotes (231)
    on left: 264 265, on right: 263 265 269 272
tempo_event (170)
    on left: 82 83 84, on right: 232
tempo_range (265)
    on left: 353 354, on right: 82 83
toplevel_expression (150)
    on left: 6 7 8 9 10 11 12 13, on right: 2
tremolo_type (247)
    on left: 303 304, on right: 257
unsigned_number (271)
    on left: 370 371, on right: 93
```

Anhang D GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

   The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Anhang E Index der LilyPond-Befehle

Dieser Index listet alle LilyPond Befehle und Schlüsselwörter auf, versehen mit Verweisen zu den Abschnitten im Handbuch, die den Befehl beschreiben oder seine Verwendung diskutieren. Der erste Teil zeigt auf die genaue Stelle im Handbuch, an der der Befehl oder das Schlüsselwort erscheint, der zweite Teil zeigt auf den entsprechenden Abschnitt.

!

! 6

'

' 1

,

, ... 1

-

- 106

.

. 41

/

/ 375
/+ 376

:

: 141

<

< 142
<...> 142

=

= 9

>

> 142

?

? 6

[

[..... 84

]

] 84

^

^ 375

_

_ 234

\

\! 109
\(..... 119
\) 119
\< 109
\> 109
\abs-fontsize 597
\accent 106
\accepts 522, 523, 524
\addChordShape 333
\addInstrumentDefinition 181
\addlyrics 228, 230, 231
\addQuote 181
\aeolian 20
\afterGrace 99
\aikenHeads 37
\aikenHeadsMinor 37
\alias 522
\allowPageTurn 477
\alternative 129
\arpeggio 123
\arpeggioArrowDown 123
\arpeggioArrowUp 123
\arpeggioBracket 124
\arpeggioNormal 123
\arpeggioParenthesis 124
\arpeggioParenthesisDashed 124
\arrow-head 218, 620
\ascendens 407, 414
\auctum 407, 414
\augmentum 414
\autoBeamOff 74
\autoBeamOn 74
\autochange 290
\backslashed-digit 633
\balloonGrobText 197
\balloonLengthOff 197
\balloonLengthOn 197
\balloonText 197
\bar 87, 92
\barNumberCheck 96
\beam 621
\bendAfter 121
\bold 211, 598
\book 428, 431
\bookOutputName 430
\bookOutputSuffix 430

\bookpart ... 429, 431, 475
\box ... 217, 598
\bracket ... 114, 217, 621
\break ... 475
\breathe ... 120
\breve ... 41
\breve ... 51
\cadenzaOff ... 67
\cadenzaOn ... 67
\caesura ... 405
\caps ... 598
\cavum ... 407, 414
\center-align ... 213, 606
\center-column ... 215, 606
\change ... 289
\char ... 634
\chordGlissando ... 309
\chordmode ... 5, 13, 331
\chords ... 377
\circle ... 217, 621
\clef ... 16
\cm ... 539
\coda ... 106
\column ... 215, 607
\column-lines ... 639
\combine ... 218, 607
\compressFullBarRests ... 55
\compressFullBarRests ... 56
\concat ... 607
\consists ... 522
\context ... 515
\contextStringTunings ... 318
\cr ... 109
\cresc ... 110
\crescHairpin ... 110
\crescTextCresc ... 110
\cueDuring ... 185
\customTabClef ... 626
\decr ... 109
\decresc ... 110
\defaultTimeSignature ... 59
\deminutum ... 407, 414
\denies ... 522, 523, 524
\descendens ... 407, 414
\dim ... 110
\dimHairpin ... 110
\dimTextDecr ... 110
\dimTextDecresc ... 110
\dimTextDim ... 110
\dir-column ... 608
\displayLilyMusic ... 448
\divisioMaior ... 405
\divisioMaxima ... 405
\divisioMinima ... 405
\dorian ... 20
\dotsDown ... 42
\dotsNeutral ... 42
\dotsUp ... 42
\doubleflat ... 626
\doublesharp ... 626
\downbow ... 106, 301
\downmordent ... 106
\downprall ... 106
\draw-circle ... 218, 621
\draw-hline ... 622
\draw-line ... 218, 622
\drummode ... 161
\dynamic ... 114, 598
\dynamicDown ... 111
\dynamicNeutral ... 111
\dynamicUp ... 111
\easyHeadsOff ... 35
\easyHeadsOn ... 35
\epsfile ... 219, 622
\espressivo ... 106, 110
\expandFullBarRests ... 55
\expandFullBarRests ... 56
\eyeglasses ... 634
\f ... 109
\featherDurations ... 86
\fermata ... 106
\fermataMarkup ... 55
\fermataMarkup ... 56
\fermataMarkup ... 106
\ff ... 109
\fff ... 109
\ffff ... 109
\fffff ... 109
\fill-line ... 215, 608
\fill-with-pattern ... 609
\filled-box ... 218, 623
\finalis ... 405
\finger ... 191, 599
\flageolet ... 106
\flat ... 627
\flexa ... 414
\fontCaps ... 599
\fontsize ... 211, 599
\footnote ... 634
\fp ... 109
\fraction ... 634
\frenchChords ... 380
\fret-diagram ... 322, 630
\fret-diagram-terse ... 324, 630
\fret-diagram-verbose ... 325, 631
\fromproperty ... 634
\funkHeads ... 37
\funkHeadsMinor ... 37
\general-align ... 215, 609
\germanChords ... 380
\glissando ... 122
\grace ... 99
\halfopen ... 106
\halign ... 214, 610
\harmonic ... 301, 309
\harmonicByFret ... 309
\harmonicByRatio ... 309
\harp-pedal ... 632
\hbracket ... 217, 623
\hcenter-in ... 611
\header ... 431
\hideKeySignature ... 361
\hideNotes ... 193
\hideSplitTiedTabNotes ... 308
\hideStaffSwitch ... 292
\hspace ... 611
\huge ... 189, 213, 599
\improvisationOff ... 40, 72
\improvisationOn ... 40, 72
\in ... 539

\inclinatum ... 407, 414
\include ... 441
\instrumentSwitch ... 181
\inversion ... 13
\ionian ... 20
\italianChords ... 380
\italic ... 211, 599
\justified-lines ... 222, 639
\justify ... 216, 612
\justify-field ... 612
\justify-string ... 613
\keepWithTag ... 444
\key ... 20
\key ... 37
\killCues ... 187
\label, ... 439
\laissezVibrer ... 49
\large ... 189, 213, 600
\larger ... 211, 213, 600
\layout ... 431, 471
\left-align ... 213, 614
\left-brace ... 634
\left-column ... 614
\lheel ... 106
\line ... 614
\linea ... 414
\lineprall ... 106
\locrian ... 20
\longa ... 41
\longa ... 51
\longfermata ... 106
\lookup ... 635
\lower ... 214, 614
\ltoe ... 106
\lydian ... 20
\lyricmode ... 227, 228
\lyricsto ... 228, 230, 231
\magnify ... 211, 600
\major ... 20
\makeClusters ... 145
\makeStringTuning ... 318
\marcato ... 106
\mark ... 97, 204
\markalphabet ... 635
\markletter ... 635
\markup ... 204, 208, 209, 210
\markuplines ... 209, 222
\maxima ... 41
\maxima ... 51
\medium ... 600
\melisma ... 235
\melismaEnd ... 235
\mergeDifferentlyDottedOff ... 149
\mergeDifferentlyDottedOn ... 149
\mergeDifferentlyHeadedOff ... 149
\mergeDifferentlyHeadedOn ... 149
\mf ... 109
\midi ... 431
\minor ... 20
\mixolydian ... 20
\mm ... 539
\modalInversion ... 15
\modalTranspose ... 14
\mordent ... 106
\mp ... 109
\musicglyph ... 98, 627
\name ... 522
\natural ... 627
\new ... 515
\noBeam ... 84
\noBreak ... 475
\noPageBreak ... 476
\noPageTurn ... 477
\normal-size-sub ... 601
\normal-size-super ... 601
\normal-text ... 601
\normalsize ... 189, 213, 602
\note ... 628
\note-by-number ... 627
\null ... 214, 635
\number ... 602
\numericTimeSignature ... 59
\octaveCheck ... 9
\on-the-fly ... 636
\once ... 530, 532
\oneVoice ... 146
\open ... 106, 301
\oriscus ... 407, 414
\ottava ... 21
\override ... 531, 636
\override in \lyricmode ... 227
\override-lines ... 639
\overrideTimeSignatureSettings ... 59
\p ... 109
\pad-around ... 217, 615
\pad-markup ... 217, 615
\pad-to-box ... 217, 615
\pad-x ... 217, 616
\page-link ... 636
\page-ref ... 636
\page-ref ... 439
\pageBreak ... 476
\pageTurn ... 477
\paper ... 431, 437
\paper ... 462
\parallelMusic ... 158
\parenthesize ... 195, 623
\partcombine ... 154
\partial ... 66, 129, 130
\path ... 624
\pattern ... 636
\pes ... 414
\phrasingSlurDashed ... 119
\phrasingSlurDashPattern ... 119
\phrasingSlurDotted ... 119
\phrasingSlurDown ... 119
\phrasingSlurHalfDashed ... 119
\phrasingSlurHalfSolid ... 119
\phrasingSlurNeutral ... 119
\phrasingSlurSolid ... 119
\phrasingSlurUp ... 119
\phrygian ... 20
\pitchedTrill ... 127
\portato ... 106
\postscript ... 219, 625
\powerChords ... 346
\pp ... 109
\ppp ... 109
\pppp ... 109
\ppppp ... 109

\prall ... 106
\pralldown ... 106
\prallmordent ... 106
\prallprall ... 106
\prallup ... 106
\predefinedFretboardsOff ... 341
\predefinedFretboardsOn ... 341
\pt ... 539
\put-adjacent ... 616
\quilisma ... 407, 414
\quoteDuring ... 181, 185
\raise ... 214, 616
\relative ... 2, 5, 13, 291
\RemoveEmptyStaves ... 175
\RemoveEmptyStaves ... 176
\removeWithTag ... 444
\repeat ... 129
\repeat percent ... 138
\repeat tremolo ... 140
\repeatTie ... 48, 131, 251
\rest ... 51
\retrograde ... 13
\reverseturn ... 106
\revert ... 532
\revertTimeSignatureSettings ... 60
\rfz ... 109
\rheel ... 106
\right-align ... 213, 616
\right-brace ... 637
\right-column ... 617
\rightHandFinger ... 343
\roman ... 602
\rotate ... 617
\rounded-box ... 217, 625
\rtoe ... 106
\sacredHarpHeads ... 37
\sacredHarpHeadsMinor ... 37
\sans ... 602
\scale ... 625
\scaleDurations ... 47, 68
\score ... 427, 431, 628
\segno ... 106
\semiflat ... 629
\semiGermanChords ... 380
\semisharp ... 629
\sesquiflat ... 629
\sesquisharp ... 629
\set ... 77, 529
\sf ... 109
\sff ... 109
\sfz ... 109
\sharp ... 629
\shiftOff ... 149
\shiftOn ... 149
\shiftOnn ... 149
\shiftOnnn ... 149
\shortfermata ... 106
\showKeySignature ... 361
\showStaffSwitch ... 292
\signumcongruentiae ... 106
\simple ... 603
\skip ... 53, 250
\slashed-digit ... 637
\slurDashed ... 116
\slurDashPattern ... 117
\slurDotted ... 116
\slurDown ... 116
\slurHalfDashed ... 116
\slurHalfSolid ... 116
\slurNeutral ... 116
\slurSolid ... 116
\slurUp ... 117
\small ... 189, 213, 603
\smallCaps ... 603
\smaller ... 211, 213, 603
\snappizzicato ... 106
\sostenutoOff ... 294
\sostenutoOn ... 294
\southernHarmonyHeads ... 37
\southernHarmonyHeadsMinor ... 37
\sp ... 109
\spp ... 109
\staccatissimo ... 106
\staccato ... 106
\startGroup ... 200
\startStaff ... 170, 171
\startTrillSpan ... 126
\stemDown ... 196
\stemNeutral ... 196
\stemUp ... 196
\stencil ... 637
\stopGroup ... 200
\stopped ... 106
\stopStaff ... 170, 171, 175
\stopTrillSpan ... 126
\storePredefinedDiagram ... 333
\stropha ... 407, 414
\strut ... 638
\sub ... 212, 604
\super ... 212, 604
\sustainOff ... 294
\sustainOn ... 294
\tabChordRepetition ... 307
\tabFullNotation ... 306
\table-of-contents ... 440, 639
\tag ... 444
\taor ... 361
\teeny ... 189, 213, 604
\tempo ... 63
\tenuto ... 106
\text ... 605
\textLengthOff ... 56
\textLengthOff ... 202
\textLengthOn ... 56
\textLengthOn ... 202
\textSpannerDown ... 203
\textSpannerNeutral ... 203
\textSpannerUp ... 203
\thumb ... 106, 191
\tied-lyric ... 630
\tieDashed ... 49
\tieDotted ... 49
\tieDown ... 49
\tieNeutral ... 49
\tieSolid ... 49
\tieUp ... 49
\time ... 58, 77
\times ... 43
\times ... 68
\tiny ... 189, 213, 605

\tocItem 440
\translate 215, 617
\translate-scaled 215, 618
\transparent 638
\transpose 5, 10, 13
\transposedCueDuring 188
\transposition 22, 181
\treCorde 294
\triangle 218, 626
\trill 106, 126
\tupletDown 43
\tupletNeutral 43
\tupletUp 43
\turn 106
\tweak 533
\type 522
\typewriter 605
\unaCorda 294
\underline 211, 605
\unfoldRepeats 454
\unHideNotes 193
\unset 530
\upbow 106, 301
\upmordent 106
\upprall 106
\upright 606
\varcoda 106
\vcenter 618
\verbatim-file 638
\verylongfermata 106
\virga 407, 414
\virgula 405
\voiceFourStyle 149
\voiceNeutralStyle 149
\voiceOne 146
\voiceOne ... \voiceFour 146
\voiceOneStyle 149
\voiceThreeStyle 149
\voiceTwoStyle 149
\vspace 618
\walkerHeads 37
\walkerHeadsMinor 37
\whiteout 638
\with 519
\with-color 194, 638
\with-dimensions 639
\with-link 639
\with-url 626
\woodwind-diagram 632
\wordwrap 216, 619
\wordwrap-field 619
\wordwrap-internal 639
\wordwrap-lines 222, 639
\wordwrap-string 620
\wordwrap-string-internal 640

|

| 96

~

~ 48

A

accepts 522, 523, 524
addChordShape 333
addInstrumentDefinition 181
addlyrics 231
addQuote 181
aeolian 20
afterGrace 99
aikenHeads 37
aikenHeadsMinor 37
alias 522
alignAboveContext 524
alignBelowContext 524
annotate-spacing 509
arpeggio 123
arpeggioArrowDown 123
arpeggioArrowUp 123
arpeggioBracket 124
arpeggioNormal 123
arpeggioParenthesis 124
arpeggioParenthesisDashed 124
arranger 433
arrow-head 218
ascendens 407
auctum 407
aug 373
auto-first-page-number 470
autoBeaming 77
autoBeamOff 74
autoBeamOn 74
autoBeamSettings 81
autochange 290
automaticBars 550

B

Balloon_engraver 197
balloonGrobText 197
balloonLengthOff 197
balloonLengthOn 197
balloonText 197
banjo-c-tuning 348
banjo-modal-tuning 348
banjo-open-d-tuning 348
banjo-open-dm-tuning 348
bar 87, 92
barCheckSynchronize 96
BarNumber 93
barNumberCheck 96
barNumberVisibility 93
bartype 92
base-shortest-duration 498
baseMoment 77
beamExceptions 77
beatStructure 77
bendAfter 121
binding-offset 468
blank-after-score-page-force 469
blank-last-page-force 469
blank-page-force 469
bold 211
bookTitleMarkup 436
bottom-margin 463
box 217

bracket ... 114, 217, 294
break-align-symbols ... 556
break-visibility ... 547
breakable ... 75
breakbefore ... 433
breathe ... 120
breve ... 41
breve ... 51

C

cadenzaOff ... 67
cadenzaOn ... 67
caesura ... 405
cavum ... 407
center-align ... 213
center-column ... 215
change ... 289
check-consistency ... 467
chordChanges ... 378
chordmode ... 5, 13, 331
chordNameExceptions ... 380
chordNameLowercaseMinor ... 380
ChordNames ... 331, 376
chordNameSeparator ... 380
chordNoteNamer ... 380
chordPrefixSpacer ... 380
chordRootNamer ... 380
circle ... 217
clef ... 16
color ... 194
column ... 215
combine ... 218
common-shortest-duration ... 498
Completion_heads_engraver ... 71
Completion_rest_engraver ... 71
composer ... 433
compressFullBarRests ... 55
compressFullBarRests ... 56
consists ... 522
context ... 515
contextStringTunings ... 318
controlpitch ... 9
copyright ... 434
cr ... 109
cresc ... 110
crescHairpin ... 110
crescTextCresc ... 110
cross ... 34
cross-staff ... 292
cueDuring ... 185
currentBarNumber ... 92
currentBarNumber ... 105

D

decr ... 109
decresc ... 110
dedication ... 433
default ... 24
default-staff-staff-spacing ... 482
defaultBarType ... 92
defaultTimeSignature ... 59
deminutum ... 407
denies ... 522, 523, 524
descendens ... 407
dim ... 110, 373
dimHairpin ... 110
dimTextDecr ... 110
dimTextDecresc ... 110
dimTextDim ... 110
divisioMaior ... 405
divisioMaxima ... 405
divisioMinima ... 405
dodecaphonic ... 29
dorian ... 20
dotsDown ... 42
dotsNeutral ... 42
dotsUp ... 42
draw-circle ... 218
draw-line ... 218
drummode ... 161
DrumStaff ... 161
dynamic ... 114
dynamicDown ... 111
dynamicNeutral ... 111
dynamicUp ... 111

E

easyHeadsOff ... 35
easyHeadsOn ... 35
epsfile ... 219
espressivo ... 110
evenFooterMarkup ... 437
evenHeaderMarkup ... 437
expandFullBarRests ... 55
expandFullBarRests ... 56
explicitClefVisibility ... 549
explicitKeySignatureVisibility ... 549
extra-offset ... 482

F

f ... 109
featherDurations ... 86
fermataMarkup ... 55
fermataMarkup ... 56
ff ... 109
fff ... 109
ffff ... 109
fffff ... 109
fill-line ... 215
filled-box ... 218
finalis ... 405
finger ... 191
first-page-number ... 470
flag-style ... 292
followVoice ... 292
font-interface ... 190, 223
font-size ... 189, 190
fontsize ... 211
fontSize ... 189
forget ... 30
four-string-banjo ... 348
fp ... 109
fret-diagram ... 322
fret-diagram-interface ... 327
fret-diagram-terse ... 324

fret-diagram-verbose 325
FretBoards 330
funkHeads 37
funkHeadsMinor 37

G

general-align 215
glissando 122
grace 99
GregorianTranscriptionStaff 161
Grid_line_span_engraver 198
Grid_point_engraver 198
gridInterval 198
grow-direction 86

H

halign 214
harmonic 301
hbracket 217
hideKeySignature 361
hideNotes 193
hideStaffSwitch 292
horizontal-shift 468
Horizontal_bracket_engraver 200
huge 189, 213

I

improvisationOff 40, 72
improvisationOn 40, 72
inclinatum 407
include-settings 447
indent 179, 468, 501
inner-margin 468
instrument 433
instrumentSwitch 181
inversion 13
ionian 20
italic 211

J

justified-lines 222
justify 216

K

keepWithTag 444
key 20
key 37
killCues 187

L

laissezVibrer 49
landscape 462
large 189, 213
larger 211, 213
last-bottom-spacing 466
layout file 472
left-align 213
left-margin 467
length 292
line-width 466, 501
linea 407
locrian 20
longa 41
longa 51
lower 214
ly:minimal-breaking 477
ly:optimal-breaking 476
ly:page-turn-breaking 476
lydian 20
lyricsto 231

M

m 373
magnify 211
magstep 189, 540
maj 373
major 20
major seven symbols 380
majorSevenSymbol 380
make-dynamic-script 114
make-pango-font-tree 225
makeClusters 145
makeStringTuning 318
mark 97, 204
markup 204, 208, 209, 210
markup-markup-spacing 466
markup-system-spacing 465
markuplines 209, 222
max-systems-per-page 468
maxima 41
maxima 51
measureLength 77
measureLength 105
measurePosition 66
measurePosition 105
MensuralStaff 161, 395
MensuralVoice 395
mergeDifferentlyDottedOff 149
mergeDifferentlyDottedOn 149
mergeDifferentlyHeadedOff 149
mergeDifferentlyHeadedOn 149
meter 433
mf 109
min-systems-per-page 468
minimum-Y-extent 482
minimumFret 306, 342
minimumPageTurnLength 477
minimumRepeatLengthForPageTurn 477
minor 20
mixed 294
mixolydian 20
modalInversion 15
modalTranspose 14
modern 26
modern-cautionary 26
modern-voice 27
modern-voice-cautionary 27
moderntab 320
mp 109
MultiMeasureRestText 55
musicglyph 98

N

name ... 522
neo-modern ... 28
neo-modern-cautionary ... 28
neo-modern-voice ... 28
neo-modern-voice-cautionary ... 29
new ... 515
no-reset ... 29
noBeam ... 84
nonstaff-nonstaff-spacing ... 482
nonstaff-relatedstaff-spacing ... 482
nonstaff-unrelatedstaff-spacing ... 482
normalsize ... 189, 213
Note_heads_engraver ... 71
null ... 214
numericTimeSignature ... 59

O

octaveCheck ... 9
oddFooterMarkup ... 437
oddHeaderMarkup ... 437
once ... 530, 532
oneVoice ... 146
opus ... 433
oriscus ... 407
ottava ... 21
outer-margin ... 468
outside-staff-horizontal-padding ... 496
outside-staff-padding ... 496
outside-staff-priority ... 496
override ... 531
overrideTimeSignatureSettings ... 59

P

p ... 109
pad-around ... 217
pad-markup ... 217
pad-to-box ... 217
pad-x ... 217
page-breaking ... 469
page-breaking-system-system-spacing ... 469
page-count ... 469
page-spacing-weight ... 470
paper-height ... 463
paper-width ... 466
parallelMusic ... 158
parenthesize ... 195
partcombine ... 154
partial ... 66
pedalSustainStyle ... 294
percent ... 138
phrasingSlurDashed ... 119
phrasingSlurDashPattern ... 119
phrasingSlurDotted ... 119
phrasingSlurDown ... 119
phrasingSlurHalfSolid ... 119
phrasingSlurNeutral ... 119
phrasingSlurSolid ... 119
phrasingSlurUp ... 119
phrygian ... 20
piano ... 27
piano-cautionary ... 28
PianoStaff ... 288, 290
piece ... 433
pipeSymbol ... 96
Pitch_squash_engraver ... 72
pitchedTrill ... 127
poet ... 433
postscript ... 219
powerChords ... 346
pp ... 109
ppp ... 109
pppp ... 109
ppppp ... 109
predefinedFretboardsOff ... 341
predefinedFretboardsOn ... 341
print-all-headers ... 436, 470
print-first-page-number ... 470
print-page-number ... 470

Q

quilisma ... 407
quotedCueEventTypes ... 184
quotedEventTypes ... 184
quoteDuring ... 181, 185

R

r ... 51
R ... 54
ragged-bottom ... 464
ragged-last ... 467, 501
ragged-last-bottom ... 464
ragged-right ... 467, 501
raise ... 214
relative ... 2, 5, 13, 291
RemoveEmptyStaves ... 175
RemoveEmptyStaves ... 176
removeWithTag ... 444
repeatCommands ... 134
repeatTie ... 48
rest ... 51
retrograde ... 13
revert ... 532
revertTimeSignatureSettings ... 60
rfz ... 109
rgb-color ... 194
RhythmicStaff ... 161
right-align ... 213
right-margin ... 467
rightHandFinger ... 343
rounded-box ... 217

S

s ... 53
sacredHarpHeads ... 37
sacredHarpHeadsMinor ... 37
scaleDurations ... 47, 68
score-markup-spacing ... 465
score-system-spacing ... 466
scoreTitleMarkup ... 436
self-alignment-X ... 482
set ... 77, 529
set-accidental-style ... 24

set-octavation ... 21
sf ... 109
sff ... 109
sfz ... 109
shiftOff ... 149
shiftOn ... 149
shiftOnn ... 149
shiftOnnn ... 149
short-indent ... 179, 468
show-available-fonts ... 225
showFirstLength ... 449
showKeySignature ... 361
showLastLength ... 449
showStaffSwitch ... 292
skip ... 53
skipTypesetting ... 449
slurDashed ... 116
slurDashPattern ... 117
slurDotted ... 116
slurDown ... 116
slurHalfDashed ... 116
slurHalfSolid ... 116
slurNeutral ... 116
slurSolid ... 116
slurUp ... 117
small ... 189, 213
smaller ... 211, 213
sostenutoOff ... 294
sostenutoOn ... 294
southernHarmonyHeads ... 37
southernHarmonyHeadsMinor ... 37
sp ... 109
spacing ... 498
spp ... 109
staff-affinity ... 482
staff-staff-spacing ... 482
Staff.midiInstrument ... 452
Staff_symbol_engraver ... 175
staffgroup-staff-spacing ... 482
start-repeat ... 134
startGroup ... 200
startStaff ... 170, 171
startTrillSpan ... 126
Stem ... 292
stem-spacing-correction ... 498
stemDown ... 196
stemLeftBeamCount ... 85
stemNeutral ... 196
stemRightBeamCount ... 85
stemUp ... 196
stopGroup ... 200
stopStaff ... 170, 171, 175
stopTrillSpan ... 126
storePredefinedDiagram ... 333
stringTunings ... 330
StringTunings ... 317
stropha ... 407
sub ... 212
subdivideBeams ... 80
subsubtitle ... 433
subtitle ... 433
suggestAccidentals ... 401
super ... 212
sus ... 375
sustainOff ... 294
sustainOn ... 294
system-count ... 469
system-separator-markup ... 470
system-system-spacing ... 466
systems-per-page ... 469

T

tabFullNotation ... 306
TabStaff ... 161, 305
TabVoice ... 305
tag ... 444
tagline ... 434
taor ... 361
teaching ... 29
teeny ... 189, 213
tempo ... 63
text ... 294
textLengthOff ... 56
textLengthOff ... 202
textLengthOn ... 56
textLengthOn ... 202
textSpannerDown ... 203
textSpannerNeutral ... 203
textSpannerUp ... 203
thumb ... 191
tieDashed ... 49
tieDotted ... 49
tieDown ... 49
tieNeutral ... 49
tieSolid ... 49
tieUp ... 49
time ... 58, 77
times ... 43
times ... 68
timeSignatureFraction ... 68
tiny ... 189, 213
title ... 433
top-margin ... 463
top-markup-spacing ... 466
top-system-spacing ... 466
translate ... 215
translate-scaled ... 215
transpose ... 5, 10, 13
transposedCueDuring ... 188
transposition ... 22, 181
treCorde ... 294
tremolo ... 140
tremoloFlags ... 141
triangle ... 218
trill ... 126
tupletDown ... 43
tupletNeutral ... 43
TupletNumber ... 44
tupletNumberFormatFunction ... 43
tupletSpannerDuration ... 43
tupletUp ... 43
tweak ... 533
two-sided ... 467
type ... 522

U

unaCorda ... 294

underline 211
unfold 136
unHideNotes 193
unset 530

V

VaticanaStaff 161, 403
VaticanaVoice 403
VerticalAxisGroup 482
virga 407
virgula 405
voice 24, 26
Voice 146
voiceOne 146

W

walkerHeads 37
walkerHeadsMinor 37
whichBar 92
with 519
with-color 194
wordwrap 216
wordwrap-lines 222

X

X-offset 482
x11-color 194, 195

Anhang F LilyPond-Index

Zusätzlich zu allen LilyPond Befehlen und Schlüsselwörtern listet dieser Index alle relevanten Begriffe auf und verlinkt sie mit den entsprechenden Abschnitten, wo sie erklärt werden. Der erste Teil zeigt auf die genaue Stelle im Handbuch, an der der der Begriff vorkommt, der zweite Teil zeigt auf den gesamten Abschnitt, in dem das Thema behandelt wird.

!

! 6

'

' 1

,

, ... 1

-

- 106

.

. 41

/

/ 375
/+ 376

:

: 141

<

< 142
<...> 142

=

= 9

>

> 142

?

? 6

[

[..... 84

]

] 84

^

^ 375

_

_ 234

\

\! 109
\(..... 119
\) 119
\< 109
\> 109
\abs-fontsize 597
\accent 106
\accepts 522, 523, 524
\addChordShape 333
\addInstrumentDefinition 181
\addlyrics 228, 230, 231
\addQuote 181
\aeolian 20
\afterGrace 99
\aikenHeads 37
\aikenHeadsMinor 37
\alias 522
\allowPageTurn 477
\alternative 129
\arpeggio 123
\arpeggioArrowDown 123
\arpeggioArrowUp 123
\arpeggioBracket 124
\arpeggioNormal 123
\arpeggioParenthesis 124
\arpeggioParenthesisDashed 124
\arrow-head 218, 620
\ascendens 407, 414
\auctum 407, 414
\augmentum 414
\autoBeamOff 74
\autoBeamOn 74
\autochange 290
\backslashed-digit 633
\balloonGrobText 197
\balloonLengthOff 197
\balloonLengthOn 197
\balloonText 197
\bar 87, 92
\barNumberCheck 96
\beam 621
\bendAfter 121
\bold 211, 598
\book 428, 431
\bookOutputName 430
\bookOutputSuffix 430

\bookpart ... 429, 431, 475
\box ... 217, 598
\bracket ... 114, 217, 621
\break ... 475
\breathe ... 120
\breve ... 41
\breve ... 51
\cadenzaOff ... 67
\cadenzaOn ... 67
\caesura ... 405
\caps ... 598
\cavum ... 407, 414
\center-align ... 213, 606
\center-column ... 215, 606
\change ... 289
\char ... 634
\chordGlissando ... 309
\chordmode ... 5, 13, 331
\chords ... 377
\circle ... 217, 621
\clef ... 16
\cm ... 539
\coda ... 106
\column ... 215, 607
\column-lines ... 639
\combine ... 218, 607
\compressFullBarRests ... 55
\compressFullBarRests ... 56
\concat ... 607
\consists ... 522
\context ... 515
\contextStringTunings ... 318
\cr ... 109
\cresc ... 110
\crescHairpin ... 110
\crescTextCresc ... 110
\cueDuring ... 185
\customTabClef ... 626
\decr ... 109
\decresc ... 110
\defaultTimeSignature ... 59
\deminutum ... 407, 414
\denies ... 522, 523, 524
\descendens ... 407, 414
\dim ... 110
\dimHairpin ... 110
\dimTextDecr ... 110
\dimTextDecresc ... 110
\dimTextDim ... 110
\dir-column ... 608
\displayLilyMusic ... 448
\divisioMaior ... 405
\divisioMaxima ... 405
\divisioMinima ... 405
\dorian ... 20
\dotsDown ... 42
\dotsNeutral ... 42
\dotsUp ... 42
\doubleflat ... 626
\doublesharp ... 626
\downbow ... 106, 301
\downmordent ... 106
\downprall ... 106
\draw-circle ... 218, 621
\draw-hline ... 622
\draw-line ... 218, 622
\drummode ... 161
\dynamic ... 114, 598
\dynamicDown ... 111
\dynamicNeutral ... 111
\dynamicUp ... 111
\easyHeadsOff ... 35
\easyHeadsOn ... 35
\epsfile ... 219, 622
\espressivo ... 106, 110
\expandFullBarRests ... 55
\expandFullBarRests ... 56
\eyeglasses ... 634
\f ... 109
\featherDurations ... 86
\fermata ... 106
\fermataMarkup ... 55
\fermataMarkup ... 56
\fermataMarkup ... 106
\ff ... 109
\fff ... 109
\ffff ... 109
\fffff ... 109
\fill-line ... 215, 608
\fill-with-pattern ... 609
\filled-box ... 218, 623
\finalis ... 405
\finger ... 191, 599
\flageolet ... 106
\flat ... 627
\flexa ... 414
\fontCaps ... 599
\fontsize ... 211, 599
\footnote ... 634
\fp ... 109
\fraction ... 634
\frenchChords ... 380
\fret-diagram ... 322, 630
\fret-diagram-terse ... 324, 630
\fret-diagram-verbose ... 325, 631
\fromproperty ... 634
\funkHeads ... 37
\funkHeadsMinor ... 37
\general-align ... 215, 609
\germanChords ... 380
\glissando ... 122
\grace ... 99
\halfopen ... 106
\halign ... 214, 610
\harmonic ... 301, 309
\harmonicByFret ... 309
\harmonicByRatio ... 309
\harp-pedal ... 632
\hbracket ... 217, 623
\hcenter-in ... 611
\header ... 431
\hideKeySignature ... 361
\hideNotes ... 193
\hideSplitTiedTabNotes ... 308
\hideStaffSwitch ... 292
\hspace ... 611
\huge ... 189, 213, 599
\improvisationOff ... 40, 72
\improvisationOn ... 40, 72
\in ... 539

\inclinatum ... 407, 414
\include ... 441
\instrumentSwitch ... 181
\inversion ... 13
\ionian ... 20
\italianChords ... 380
\italic ... 211, 599
\justified-lines ... 222, 639
\justify ... 216, 612
\justify-field ... 612
\justify-string ... 613
\keepWithTag ... 444
\key ... 20
\key ... 37
\killCues ... 187
\label, ... 439
\laissezVibrer ... 49
\large ... 189, 213, 600
\larger ... 211, 213, 600
\layout ... 431, 471
\left-align ... 213, 614
\left-brace ... 634
\left-column ... 614
\lheel ... 106
\line ... 614
\linea ... 414
\lineprall ... 106
\locrian ... 20
\longa ... 41
\longa ... 51
\longfermata ... 106
\lookup ... 635
\lower ... 214, 614
\ltoe ... 106
\lydian ... 20
\lyricmode ... 227, 228
\lyricsto ... 228, 230, 231
\magnify ... 211, 600
\major ... 20
\makeClusters ... 145
\makeStringTuning ... 318
\marcato ... 106
\mark ... 97, 204
\markalphabet ... 635
\markletter ... 635
\markup ... 204, 208, 209, 210
\markuplines ... 209, 222
\maxima ... 41
\maxima ... 51
\medium ... 600
\melisma ... 235
\melismaEnd ... 235
\mergeDifferentlyDottedOff ... 149
\mergeDifferentlyDottedOn ... 149
\mergeDifferentlyHeadedOff ... 149
\mergeDifferentlyHeadedOn ... 149
\mf ... 109
\midi ... 431
\minor ... 20
\mixolydian ... 20
\mm ... 539
\modalInversion ... 15
\modalTranspose ... 14
\mordent ... 106
\mp ... 109
\musicglyph ... 98, 627
\name ... 522
\natural ... 627
\new ... 515
\noBeam ... 84
\noBreak ... 475
\noPageBreak ... 476
\noPageTurn ... 477
\normal-size-sub ... 601
\normal-size-super ... 601
\normal-text ... 601
\normalsize ... 189, 213, 602
\note ... 628
\note-by-number ... 627
\null ... 214, 635
\number ... 602
\numericTimeSignature ... 59
\octaveCheck ... 9
\on-the-fly ... 636
\once ... 530, 532
\oneVoice ... 146
\open ... 106, 301
\oriscus ... 407, 414
\ottava ... 21
\override ... 531, 636
\override in \lyricmode ... 227
\override rückgängig machen ... 532
\override, nur einmal ... 532
\override-lines ... 639
\overrideTimeSignatureSettings ... 59
\p ... 109
\pad-around ... 217, 615
\pad-markup ... 217, 615
\pad-to-box ... 217, 615
\pad-x ... 217, 616
\page-link ... 636
\page-ref ... 636
\page-ref ... 439
\pageBreak ... 476
\pageTurn ... 477
\paper ... 431, 437
\paper ... 462
\parallelMusic ... 158
\parenthesize ... 195, 623
\partcombine ... 154
\partial ... 66, 129, 130
\path ... 624
\pattern ... 636
\pes ... 414
\phrasingSlurDashed ... 119
\phrasingSlurDashPattern ... 119
\phrasingSlurDotted ... 119
\phrasingSlurDown ... 119
\phrasingSlurHalfDashed ... 119
\phrasingSlurHalfSolid ... 119
\phrasingSlurNeutral ... 119
\phrasingSlurSolid ... 119
\phrasingSlurUp ... 119
\phrygian ... 20
\pitchedTrill ... 127
\portato ... 106
\postscript ... 219, 625
\powerChords ... 346
\pp ... 109
\ppp ... 109

\pppp ... 109
\ppppp ... 109
\prall ... 106
\pralldown ... 106
\prallmordent ... 106
\prallprall ... 106
\prallup ... 106
\predefinedFretboardsOff ... 341
\predefinedFretboardsOn ... 341
\pt ... 539
\put-adjacent ... 616
\quilisma ... 407, 414
\quoteDuring ... 181, 185
\raise ... 214, 616
\relative ... 2, 5, 13, 291
\RemoveEmptyStaves ... 175
\RemoveEmptyStaves ... 176
\removeWithTag ... 444
\repeat ... 129
\repeat percent ... 138
\repeat tremolo ... 140
\repeatTie ... 48, 131, 251
\rest ... 51
\retrograde ... 13
\reverseturn ... 106
\revert ... 532
\revertTimeSignatureSettings ... 60
\rfz ... 109
\rheel ... 106
\right-align ... 213, 616
\right-brace ... 637
\right-column ... 617
\rightHandFinger ... 343
\roman ... 602
\rotate ... 617
\rounded-box ... 217, 625
\rtoe ... 106
\sacredHarpHeads ... 37
\sacredHarpHeadsMinor ... 37
\sans ... 602
\scale ... 625
\scaleDurations ... 47, 68
\score ... 427, 431, 628
\segno ... 106
\semiflat ... 629
\semiGermanChords ... 380
\semisharp ... 629
\sesquiflat ... 629
\sesquisharp ... 629
\set ... 77, 529
\sf ... 109
\sff ... 109
\sfz ... 109
\sharp ... 629
\shiftOff ... 149
\shiftOn ... 149
\shiftOnn ... 149
\shiftOnnn ... 149
\shortfermata ... 106
\showKeySignature ... 361
\showStaffSwitch ... 292
\signumcongruentiae ... 106
\simple ... 603
\skip ... 53, 250
\slashed-digit ... 637
\slurDashed ... 116
\slurDashPattern ... 117
\slurDotted ... 116
\slurDown ... 116
\slurHalfDashed ... 116
\slurHalfSolid ... 116
\slurNeutral ... 116
\slurSolid ... 116
\slurUp ... 117
\small ... 189, 213, 603
\smallCaps ... 603
\smaller ... 211, 213, 603
\snappizzicato ... 106
\sostenutoOff ... 294
\sostenutoOn ... 294
\southernHarmonyHeads ... 37
\southernHarmonyHeadsMinor ... 37
\sp ... 109
\spp ... 109
\staccatissimo ... 106
\staccato ... 106
\startGroup ... 200
\startStaff ... 170, 171
\startTrillSpan ... 126
\stemDown ... 196
\stemNeutral ... 196
\stemUp ... 196
\stencil ... 637
\stopGroup ... 200
\stopped ... 106
\stopStaff ... 170, 171, 175
\stopTrillSpan ... 126
\storePredefinedDiagram ... 333
\stropha ... 407, 414
\strut ... 638
\sub ... 212, 604
\super ... 212, 604
\sustainOff ... 294
\sustainOn ... 294
\tabChordRepetition ... 307
\tabFullNotation ... 306
\table-of-contents ... 440, 639
\tag ... 444
\taor ... 361
\teeny ... 189, 213, 604
\tempo ... 63
\tenuto ... 106
\text ... 605
\textLengthOff ... 56
\textLengthOff ... 202
\textLengthOn ... 56
\textLengthOn ... 202
\textSpannerDown ... 203
\textSpannerNeutral ... 203
\textSpannerUp ... 203
\thumb ... 106, 191
\tied-lyric ... 630
\tieDashed ... 49
\tieDotted ... 49
\tieDown ... 49
\tieNeutral ... 49
\tieSolid ... 49
\tieUp ... 49
\time ... 58, 77
\times ... 43

\times ... 68
\tiny ... 189, 213, 605
\tocItem ... 440
\translate ... 215, 617
\translate-scaled ... 215, 618
\transparent ... 638
\transpose ... 5, 10, 13
\transposedCueDuring ... 188
\transposition ... 22, 181
\treCorde ... 294
\triangle ... 218, 626
\trill ... 106, 126
\tupletDown ... 43
\tupletNeutral ... 43
\tupletUp ... 43
\turn ... 106
\tweak ... 533
\type ... 522
\typewriter ... 605
\unaCorda ... 294
\underline ... 211, 605
\unfoldRepeats ... 454
\unHideNotes ... 193
\unset ... 530
\upbow ... 106, 301
\upmordent ... 106
\upprall ... 106
\upright ... 606
\varcoda ... 106
\vcenter ... 618
\verbatim-file ... 638
\verylongfermata ... 106
\virga ... 407, 414
\virgula ... 405
\voiceFourStyle ... 149
\voiceNeutralStyle ... 149
\voiceOne ... 146
\voiceOne ... \voiceFour ... 146
\voiceOneStyle ... 149
\voiceThreeStyle ... 149
\voiceTwoStyle ... 149
\vspace ... 618
\walkerHeads ... 37
\walkerHeadsMinor ... 37
\whiteout ... 638
\with ... 519
\with-color ... 194, 638
\with-dimensions ... 639
\with-link ... 639
\with-url ... 626
\woodwind-diagram ... 632
\wordwrap ... 216, 619
\wordwrap-field ... 619
\wordwrap-internal ... 639
\wordwrap-lines ... 222, 639
\wordwrap-string ... 620
\wordwrap-string-internal ... 640

|

| ... 96

~

~ ... 48

1

15ma ... 21

8

8va ... 21
8ve ... 21

A

a due-Stellen ... 154
Abbildungen im Text ... 217
Abschnitte definieren, Notenabstände ... 499
Abschnitte markieren ... 97
absolute Lautstärke ... 109
Absolute Spezifikation von Oktaven ... 1
Absoluter Modus: Tonhöhen ... 1
Abstand vergrößern, Gesangstext ... 245
Abstand von Hilfslinien ... 169
Abstand zwischen Notensystemen ... 482
Abstand zwischen Systemen in Klaviernoten ... 292
Abstände, absolut ... 539
Abstände, skaliert ... 539
Abstände, vertikal ... 481
Abstrich ... 106, 640
Accentus ... 640
accepts ... 522, 523, 524
acciaccatura ... 673
addChordShape ... 333
addChordShape ... 673
adding a white background to text ... 638
addInstrumentDefinition ... 181
addInstrumentDefinition ... 673
Additionen in Akkorden ... 374
addlyrics ... 231
addQuote ... 181
addQuote ... 673
aeolian ... 20
Aeolisch ... 20
afterGrace ... 99
afterGrace ... 673
Aiken-Notenköpfe ... 37
aikenHeads ... 37
aikenHeadsMinor ... 37
Akkolade ... 162
Akkord, eine Noten verändern ... 533
Akkord, gebrochen ... 123
Akkord-Diagramme ... 330
Akkordbezeichnungen ... 371, 376
Akkordbezeichnungen und Bunddiagramme ... 331
Akkorddiagramm ... 321
Akkorddiagramme, automatisch ... 340
Akkorde ... 142, 371
Akkorde über zwei Systeme ... 292
Akkorde und relativer Modus ... 4
Akkorde und Überbindungen ... 48
Akkorde, Entfernen von Tönen ... 375
Akkorde, relative Tonhöhe ... 143
Akkorde, Unterdrückung wiederholt ... 378
Akkorde, zwischen Systemen mit \autochange ... 291

Akkorde: farbige Noten ... 195
Akkorde: Fingersatz ... 191
Akkorde: Versetzungszeichen ... 30
Akkordeigenschaften ... 372
Akkordeon ... 295
Akkordeon, Diskant-Symbole ... 295
Akkordeon, Register ... 295
Akkordformen für bundierte Saiteninstrumente ... 333
Akkordformen für Bundinstrumente ... 333
Akkordglissando ... 309
Akkordmodi ... 372
Akkordmodus ... 371
Akkordstufen, Alteration ... 375
Akkordstufen, Veränderung ... 375
Akkordsymbole ... 376
Akkordsymbole in MIDI ... 454
Akkordsymbole, anpassen ... 379
Akkordtabulatur ... 321
Akzent ... 106, 640
Akzidentien ... 5, 24
al niente ... 112
`alias` ... 522
`alignAboveContext` ... 524
`alignBelowContext` ... 524
alist ... 643
`allowPageTurn` ... 673
Alte Schlüssel ... 16
alternative Endungen und Gesangstext ... 250
alternative Schlüsse in ausgeschriebenen Wiederholungen ... 136
Alternative Schlüsse mit Bindebogen ... 131
alternativer Gesangstext ... 254
alternativer Schluss ... 129
Altschlüssel ... 16
Ambitus ... 31
Analyse ... 200
andere Stimmen zitieren ... 185
Ändern von Instrumentenbezeichnungen ... 180
Ändern von Schriftarten für das gesamte Dokument ... 225
Anfänger, Notenlernen ... 35
Anführungsstriche im Text ... 210
Anführungsstriche, Gesangstext ... 234
Anführungszeichen, Gesangstext ... 227
Angabe der Oktave: absolut ... 1
Anmerkung, Blase ... 197
`annotate-spacing` ... 509
Anordnung, horizontal ... 497
Anpassen von Akkordsymbolen ... 379
Anpassen von Bunddiagrammen ... 327
Anpassen von staff symbol ... 540
Anpassungen im Gesangstextmodus ... 227
Anstrich ... 106
Anzahl der Notenlinien einstellen ... 169
Anzahl der Wiederholung, ändern ... 134
Äolisch ... 20
`applyContext` ... 673
`applyMusic` ... 673
`applyOutput` ... 673
`appoggiatura` ... 673
arabische Musik ... 420
arabische Musik, Beispiel ... 424
arabische Notenbezeichnungen ... 420
Arabische Taktarten ... 422
arabische Tonarten ... 421
arabische Vorzeichen ... 421
arabisches Halb-B Versetzungszeichen ... 420
`arpeggio` ... 123
Arpeggio ... 123
Arpeggio über Systeme im Klammernstil ... 126
Arpeggio-Symbole, besondere ... 124
`arpeggioArrowDown` ... 123
`arpeggioArrowUp` ... 123
`arpeggioBracket` ... 124
`arpeggioNormal` ... 123
`arpeggioParenthesis` ... 124
`arpeggioParenthesisDashed` ... 124
`arranger` ... 433
`arrow-head` ... 218
Art der Übungszeichen ... 98
Arten von Notenköpfen ... 597
articulation-event ... 184
Artikulationszeichen ... 106
Artikulationszeichen, greg. Choral ... 406
`ascendens` ... 407
`assertBeamQuant` ... 673
`assertBeamSlope` ... 673
`associatedVoice` ... 228
assoziative Liste ... 643
Atemzeichen ... 120
`auctum` ... 407
Aufführungsanweisung: Tempo ... 63
Aufklappen von wiederholten Noten ... 136
Auflösungszeichen ... 5
Aufstrich ... 640
Auftakt ... 66
Auftakt in Wiederholung ... 130
Aufteilen von Noten ... 71
Aufteilen von Pausen ... 71
`aug` ... 373
Ausdehnen von Noten ... 47
Ausdrück, Text ... 210
Ausgabe von Akkordbezeichnungen ... 376
ausgeschriebene Wiederholungen ... 136
Ausklingen lassen ... 49
Ausklingen lassen, Bögen ... 49
Auslassen von Noten im Gesangstext ... 250
Ausnahmen, Akkordsymbole ... 381
Ausrichten an Kadenz ... 104
Ausrichtung an Objekten ... 556
Ausrichtung von Gesangstext ... 230
Ausrichtung von Taktlinien ... 94
Ausrichtung von Text ... 213
Ausrichtung von Text, Befehle ... 217
Ausrichtung, Papier ... 462
Aussehen von Taktnummern ... 93
Auswahl von Schriftgröße (Notation) ... 189
`auto-first-page-number` ... 470
auto-knee-gap ... 75
autobeam ... 77
`autoBeaming` ... 77
`autoBeamOff` ... 74
`autoBeamOn` ... 74
`autoBeamSettings` ... 81
`autochange` ... 290
`autochange` ... 673
`automaticBars` ... 550
automatische Ausrichtung von Silben ... 230
automatische Bebalkung ... 74
automatische Bunddiagramme ... 340

automatische Kombination von Stimmen 154
Automatische Versetzungszeichen 24
Automatischer Systemwechsel 290
automatischer Systemwechsel und relativer Modus 291
automatisches Aufteilen von Noten 71

B

B 5
backslashed digits 633
Balken in Kadenzen 67
Balken in polymetrischer Notation 68
Balken mit Knie 75
Balken und Gesangstext 77
Balken und Melismen 75
Balken zwischen Systemen 289
Balken, \partcombine und \autoBeamOff 76
Balken, automatisch 74
Balken, eigene Regeln 74
Balken, Einstellungen 74
Balken, gespreizt 86
Balken, letzter in Partitur 82
Balken, letzter in polyphoner Stimme 82
Balken, manuell 84
Balken, Taktartstandard 59
Balken, Unterteilung 80
Balken, Zeilenumbrüche 75
Balkenpausen, mehrtaktig 56
Ballon 197
`Balloon_engraver` 197
`balloonGrobText` 197
`balloonGrobText` 673
`balloonLengthOff` 197
`balloonLengthOn` 197
`balloonText` 197
`balloonText` 673
`banjo-c-tuning` 348
`banjo-modal-tuning` 348
`banjo-open-d-tuning` 348
`banjo-open-dm-tuning` 348
Banjo-Stimmung 348
Banjo-Tabulatur 303
Banjo-Tabulaturen 348
`bar` 87, 92
`bar` 673
`barCheckSynchronize` 96
Baritonschlüssel 16
`BarNumber` 93
`barNumberCheck` 96, 673
`barNumberVisibility` 93
Barré, anzeigen für bundierte Saiteninstrumente 345
Barré, anzeigen für Bundinstrumente 345
Barré, Gitarre 321
Bartók-Pizzicato 302
`bartype` 92
`base-shortest-duration` 498
`baseMoment` 77
Bassnote in Akkorden 375
Basso continuo 384
Bassschlüssel 16
`beamExceptions` 77
`beatStructure` 77
Bebalken in taktloser Musik 67
Bebalkung in Kadenzen 67
Bebalkung in polymetrischer Notation 68
Bebalkung, automatisch, Einstellungen 77
Bebalkung, Taktartstandard 59
Beenden eines Notensystems 169
Beenden eines Systems 170
Beenden von Notenlinien 170
Befehle zur Textausrichtung 217
Beginn eines Notensystems 161
Beginn von Wiederholung 134
Beginnen eines Notensystems 169
Beginnen von Notenlinien 170
Beispiel der arabischen Musik 424
`bendAfter` 121
`bendAfter` 673
Beschriftung 106
Beschriftung ausrichten 213
Beschriftung über mehrere Seiten 222
Beschriftung über Mehrtaktpausen 55
Beschriftung, Blocksatz 216
Beschriftung, mehrzeilig 215
Beschriftung, Notation einfügen 221
Beschriftung, Notationsobjekte einfügen 219
Beschriftung, Sonderzeichen 210
Beschriftung, Text 210
Beschriftung, Zentrieren auf der Seite 215
besondere Arpeggio-Symbole 124
besondere Notenköpfe 34
besondere Zeichen, Text 210
Bézier-Kurven 559
Bezifferter Bass 384
Bilder einbinden 219
Bindebogen 48
Bindebogen in alternativem Schluss 131
Bindebogen in Wiederholung 131
Bindebögen und Akkorde 48
Bindebogen und Wiederholung 133
Bindebögen wiederholen 48
Bindebögen, Aussehen 49
Bindebögen, durchgehend 49
Bindebögen, gepunktet 49
Bindebogen, Gesangstext 234
Bindebögen, gestrichelt 49
Bindebögen, verändern 559
Binderand 467
`binding-offset` 468
Bison 645
`blank-after-score-page-force` 469
`blank-last-page-force` 469
`blank-page-force` 469
Blase 197
Blasinstrumente 358
Blöcke, Text 215
Blocksatz, Text 216
BNF 645
Bogen zur Phrasierung 119
Bogen, Anzeige 301
Bögen, gleichzeitig 116
Bögen, gleichzeitige Phrasierung 119
Bogen, halb durchgehend, halb gestrichelt 119
Bogen, halb gestrichelt, halb durchgehend 116
Bögen, laissez vibrer 49
Bögen, manuelle Platzierung 116
Bögen, mehrfach 116
Bögen, Phrasierung 116

Bogen, Strichelung definieren ... 117
Bögen, über Noten ... 116
Bögen, unter Noten ... 116
Bögen, verändern ... 559
bold ... 211
bookOutputName ... 673
bookOutputSuffix ... 673
bookTitleMarkup ... 436
bottom-margin ... 463
box ... 217
bracket ... 114, 217, 294
Bratschenschlüssel ... 16
break-align-symbols ... 556
break-visibility ... 547
breakable ... 75
breakbefore ... 433
breathe ... 120
breathe ... 674
breve ... 41
breve ... 51
Brevis-Pause ... 51
Bund ... 306
Bunddiagramm-Beschriftung ... 322
Bunddiagramme ... 321, 330
Bunddiagramme und Akkordbezeichnungen ... 331
Bunddiagramme, anpassen ... 327
Bunddiagramme, ausführlicher Stil ... 325
Bunddiagramme, automatisch ... 340
Bunddiagramme, eigene ... 321
Bunddiagramme, eigene definieren ... 332
Bunddiagramme, Fingersatz ... 341
Bunddiagramme, knapper Stil ... 324
Bunddiagramme, Transposition ... 331
Bunddiagramme, Ukulele ... 330
bundierte Saiteninstrumente, Akkordformen ... 333
bundierte Saiteninstrumente, Fingersatz der rechten Hand ... 343
bundierte Saiteninstrumente, Flageolett ... 345
bundierte Saiteninstrumente, gedämpfte Noten ... 345
bundierte Saiteninstrumente, Position und Barré anzeigen ... 345
bundierte Saiteninstrumente, Saitenstimmung ... 317
Bundinstrumente, Akkordformen ... 333
Bundinstrumente, Fingersatz der rechten Hand ... 343
Bundinstrumente, Flageolett ... 345
Bundinstrumente, gedämpfte Noten ... 345
Bundinstrumente, Position und Barré anzeigen ... 345
Bundinstrumente, Saitenstimmung ... 317
Bundsteg ... 467

C

C-Schlüssel ... 16
cadenzaOff ... 67
cadenzaOn ... 67
caesura ... 121
caesura ... 405
callback ... 643
Capo ... 325
cavum ... 407
center-align ... 213
center-column ... 215
centering a column of text ... 606
change ... 289
changing direction of text columns ... 608
chants ... 274
check-consistency ... 467
Chor-Tenorschlüssel ... 17
chord-Akkorde ... 371
chordChanges ... 378
chordGlissando ... 674
chordmode ... 5, 13, 331
chordNameExceptions ... 380
chordNameLowercaseMinor ... 380
ChordNames ... 331, 376
chordNameSeparator ... 380
chordNoteNamer ... 380
chordPrefixSpacer ... 380
chordRootNamer ... 380
Chornoten ... 263
Chorsystem ... 162
Christian Harmony-Notenköpfe ... 37
circle ... 217
circling text ... 621
Circulus ... 640
clef ... 16
clef ... 674
closure ... 643
Cluster ... 145, 376
Coda ... 98, 106, 640
Coda am Taktstrich ... 204
color ... 194
coloring text ... 638
column ... 215
combine ... 218
common-shortest-duration ... 498
Completion_heads_engraver ... 71
Completion_rest_engraver ... 71
composer ... 433
compound time signatures ... 62
compoundMeter ... 674
compressFullBarRests ... 55
compressFullBarRests ... 56
concatenating text ... 607
consists ... 522
context ... 515
contextStringTuning ... 674
contextStringTunings ... 318
Continuo, Generalbass ... 384
controlling general text alignment ... 609
controlpitch ... 9
copyright ... 434
Copyright ... 436
Copyright-Zeichen ... 448
cr ... 109
creating empty text objects ... 635
creating horizontal spaces in text ... 611
creating text fractions ... 634
creating vertical spaces in text ... 618, 638
cresc ... 110
Crescendo ... 109
Crescendo-Klammer ... 109
Crescendoklammern, gedreht ... 552
crescHairpin ... 110
crescTextCresc ... 110
cross ... 34
cross-staff ... 292
cueClef ... 674
cueClefUnset ... 674

`cueDuring` ... 185
`cueDuring` ... 674
`cueDuringWithClef` ... 674
`currentBarNumber` ... 92
`currentBarNumber` ... 105
Custodes ... 394
Custos ... 394

D

D'al Segno ... 106
D.S. al Fine ... 98
Dal Segno ... 98
Dämpfung, bundierte Saiteninstrumente ... 345
Dämpfung, Bundinstrumente ... 345
Dateien einfügen ... 441
Dateistruktur ... 431
Dauer ... 41
Dauer, Standard ... 41
Dauern skalieren ... 47
Daumenbezeichnung ... 106, 640
`deadNote` ... 674
`decr` ... 109
`decresc` ... 110
Decrescendo ... 109
`dedication` ... 433
`default` ... 24
`default-staff-staff-spacing` ... 482
`defaultBarType` ... 92
`defaultNoteHeads` ... 674
`defaultTimeSignature` ... 59
Definieren von eigenen Bunddiagrammen ... 332
`deminutum` ... 407
`denies` ... 522, 523, 524
`descendens` ... 407
Dicke der Notenlinien einstellen ... 169
didaktischer Versetzungszeichenstil ... 29
`dim` ... 110, 373
`dimHairpin` ... 110
Diminuendo ... 109
`dimTextDecr` ... 110
`dimTextDecresc` ... 110
`dimTextDim` ... 110
Diskantsymbole, Akkordeon ... 295
`displayLilyMusic` ... 674
`displayMusic` ... 674
Divisi, Gesangstext ... 254
divisio ... 405
`divisioMaior` ... 405
`divisioMaxima` ... 405
`divisioMinima` ... 405
divisiones ... 405
`dodecaphonic` ... 29
dodekaphoner Versetzungszeichenstil ... 29
doits ... 121
Doppel-B ... 5
Doppelkreuz ... 5
Doppellinie ... 87
Doppelpraller ... 640
Doppelpunktierung ... 41
Doppelschlag ... 106
doppelte Taktartensymbole ... 68
Doppelter Taktstrich ... 87
`dorian` ... 20
Dorisch ... 20
`dotsDown` ... 42
`dotsNeutral` ... 42
`dotsUp` ... 42
`draw-circle` ... 218
`draw-line` ... 218
drawing a line across a page ... 622
drawing beams within text ... 621
drawing boxes with rounded corners ... 623
drawing boxes with rounded corners around text ... 625
drawing circles within text ... 621
drawing lines within text ... 622
drawing paths ... 624
drawing solid boxes within text ... 623
drawing triangles within text ... 626
Drehen von Objekten ... 552
Dreiklänge ... 372
Drucken von Sonderzeichen ... 210
Druckreihenfolge ... 547
`drummode` ... 161
Drums ... 349
`DrumStaff` ... 161
Dudelsack ... 361
Dur ... 20
durchgehender Legatobogen ... 116
durchgestrichener Hals ... 101
durchsichtig, Objekte ... 546
durchsichtige Noten ... 193
`dynamic` ... 114
dynamic-event ... 184
`dynamicDown` ... 111
`dynamicNeutral` ... 111
`dynamicUp` ... 111
Dynamik ... 109
Dynamik, mehrere Zeichen an einer Note ... 110
Dynamik, vertikale Position ... 111
Dynamik, zentriert für Tasteninstrumente ... 288
Dynamikzeichen, Anmerkung ... 114
Dynamikzeichen, eigene ... 114
Dynamikzeichen, Klammer ... 114

E

`easyHeadsOff` ... 35
`easyHeadsOn` ... 35
Ebenen (layer) ... 547
editorische Dynamikzeichen ... 114
editorische Noten ... 195
eigene Balkenregeln ... 74
eigene Bunddiagramme ... 321, 327
Eigene Bunddiagramme definieren ... 332
eigene Dynamikzeichen ... 114
eigene Kontexte erstellen ... 515
Eigene Saitenstimmung, Tabulatur ... 318
eigene Tabulaturen ... 317
Eigenschaften ... 529
Eigenschaften von Grob ... 531
ein System, Mehrstimmigkeit ... 146
Einbinden von Graphik ... 219
einfache closure ... 643
Einfärben von Objekten ... 194, 547
Einfärben von Stimmen ... 149
einfügen von Dateien ... 441
Einfügen von Notationsobjekten ... 219

Eingabe von Noten parallel ... 158
Eingabedatei, Struktur ... 431
eingebundene Graphik im Text ... 217
Einmal verändern von Kontexten ... 532
Einstellung von Hilfslinien ... 169
einzelnes Notensystem ... 161
Einzug ... 179
encapsulated postscript output ... 450
enclosing text in a box with rounded corners ... 625
enclosing text within a box ... 598
Ende von Wiederholung ... 134
`endSpanners` ... 674
Engraver, in Kontexte einfügen ... 522
Entfernen eines Stencil ... 546
Entfernen von Stichnoten ... 187
Entfernen von Stufen in Akkorden ... 375
Entfernen von Tönen aus Akkorden ... 374
EPS-Ausgabe ... 450
`epsfile` ... 219
Erinnerungsvorzeichen ... 6
Erklärungsblase ... 197
erste Klammer ... 129
erweiterte Akkorde ... 374
`espressivo` ... 110
Espressivo ... 106, 640
Espressivo-Artikulation ... 110
`evenFooterMarkup` ... 437
`evenHeaderMarkup` ... 437
`expandFullBarRests` ... 55
`expandFullBarRests` ... 56
`explicitClefVisibility` ... 549
`explicitKeySignatureVisibility` ... 549
`extra-offset` ... 482

F

`f` ... 109
F-Schlüssel ... 16
Fähnchen, Mensuralnotation ... 399
falls ... 121
Farbe ... 194
Farbe, RGB ... 194
Färben von Objekten ... 547
Färben von Stimmen ... 149
Farben, Liste ... 577
farbige Noten ... 194
farbige Noten in Akkorden ... 195
`featherDurations` ... 86
`featherDurations` ... 674
`fermataMarkup` ... 55
`fermataMarkup` ... 56
Fermate ... 98, 106, 640
Fermate an Taktstrich ... 204
Fermate über Mehrtaktpausen ... 55
Feta font ... 578
`ff` ... 109
`fff` ... 109
`ffff` ... 109
`fffff` ... 109
Figuren, Namen ... 267
`fill-line` ... 215
`filled-box` ... 218
`finalis` ... 405
`finger` ... 191
Fingersatz ... 191, 640
Fingersatz der rechten Hand, bundierte Saiteninstrumente ... 343
Fingersatz in Bunddiagrammen ... 341
Fingersatz und Mehrtaktpausen ... 58
Fingersatz versus Saitenzahl ... 303
Fingersatz: Akkorde ... 191
Fingersatz: Daumen-Zeichen ... 191
Fingerwechsel ... 191
`first-page-number` ... 470
`flag-style` ... 292
Flageolet ... 106, 640
Flageolett ... 301
Flageolett in Tabulaturen ... 309
Flageolett, bundierte Saiteninstrumente ... 345
Flageolett, Bundinstrumente ... 345
Flageolett, künstliches ... 302
Flageolett-Notenköpfe ... 34
Folgen einer Stimmen in anderes System ... 292
`followVoice` ... 292
font ... 643
Font, Feta ... 578
Font, Größe ändern für Notation ... 189
`font-interface` ... 190, 223
`font-size` ... 189, 190
`fontsize` ... 211
`fontSize` ... 189
`footnote` ... 674
`footnoteGrob` ... 674
`forget` ... 30
forget-Versetzungszeichenstil ... 30
Form-Notenköpfe ... 37
Formatierung von Gesangstext ... 227
Formatierung von Textstreckern ... 203
Formatierung von Triolen ... 43
Formatierung von Übungszeichen ... 98
`four-string-banjo` ... 348
`fp` ... 109
Fragmente ... 181, 185
Französischer Violinschlüssel ... 16
fret (Bunddiagramme) ... 322
Fret (Bunddiagramme) ... 321
`fret-diagram` ... 322
`fret-diagram-interface` ... 327
`fret-diagram-terse` ... 324
fret-diagram-terse-Markup ... 324
`fret-diagram-verbose` ... 325
fret-diagram-verbose-Markup ... 325
`FretBoards` ... 330
Fülllinie ... 238
Füllung um Text ... 217
Funk-Formnotenköpfe ... 37
`funkHeads` ... 37
`funkHeadsMinor` ... 37
Fußbezeichnung ... 640
Fußzeile ... 437

G

G-Schlüssel ... 16
Ganztaktpausen ... 51, 54
Ganztaktpausen und Fingersatz ... 58
Gebrochene Akkorde ... 123
gedämpft ... 106
Gedämpft ... 640
gedämpfte Noten, bundierte Saiteninstrumente ... 345

gedämpfte Noten, Bundinstrumente ... 345
gedrehte Crescendoklammern ... 552
Geisternoten ... 195
`general-align` ... 215
Generalbass ... 384
Generalbass Fortsetzungslinie ... 387
gepunkteter Legatobogen ... 116
gepunkteter Phrasierungsbogen ... 119
gerundeter Kasten, Graphik ... 217
Gesangsstimmen ... 263
Gesangstext ... 227
Gesangstext und Balken ... 77
Gesangstext und tweak-Befehl ... 534
Gesangstext und übergebundene Noten ... 251
Gesangstext und Verzierungsnoten ... 257
Gesangstext, alternativ ... 254
Gesangstext, an einer sporadischen Melodie ausrichten ... 517
Gesangstext, an Melodie ausgerichtet ... 228
Gesangstext, Auslassen von Noten ... 250
Gesangstext, Ausrichtung ... 230
Gesangstext, einer Stimme zugewiesen ... 146
Gesangstext, Formatierung ... 227
Gesangstext, innerhalb des Randes behalten ... 202
Gesangstext, Note überspringen ... 53
Gesangstext, Platz zwischen Silben ... 245
Gesangstext, Platzierung ... 240
Gesangstext, Variablen ... 239
Gesangstext, Wiederholungen ... 247
Gesangstext, Wiederholungen mit alternativen Endungen ... 250
geschweifte Klammer ... 162
geschweifte Klammern, Schachteln ... 166
gespreizte Balken ... 86
gestopft ... 106
gestrichelter Legatobogen ... 116
gestrichelter Phrasierungsbogen ... 119
geteilte Stimmen ... 265
getrennter Gesangstext ... 254
Gitarren-Akkordnotation ... 72
Gitarrengriffsymbole ... 321
Gitarrennotenköpfe ... 34
Gitarrenschlagrhythmus, Notation ... 72
Gitarrentabulatur ... 303
Gitterlinien ... 198
gleichzeitige Bögen ... 116
gleichzeitige Noten: Versetzungszeichen ... 30
gleichzeitige Phrasierungsbögen ... 119
Gleiten in Tabulaturen ... 309
Gleiten nach oben/unten ... 121
`glissando` ... 122
Glissando ... 122
Glissando, Akkorde ... 309
Glissando, nach oben ... 121
Glissando, nach unten ... 121
Glissando, unbestimmt ... 121
glyph ... 643
Glyphe ... 643
`grace` ... 99
`grace` ... 674
Grammatik von LilyPond ... 645
graphical objects ... 643
Graphik einbinden ... 219
Graphik, eingebunden ... 217
Graphische Notation ... 218
graphische Objekte ... 643
graphische Objekte, Eigenschaften ... 531
graphische Objekte, Schnittstellen ... 644
Gregorianische quadratische Neumenligaturen ... 407
Gregorianischer Choral, Transkription ... 161
`GregorianTranscriptionStaff` ... 161
`Grid_line_span_engraver` ... 198
`Grid_point_engraver` ... 198
`gridInterval` ... 198
Griff: Fingersatz ... 191
Griffsymbole, bundierte Saiteninstrumente ... 321
Griffsymbole, Bundinstrumente ... 321
grob ... 643
Grob ... 526
Grob-Eigenschaften ... 531
grob-interface ... 644
Grobs, Sichtbarkeit ... 546
Grobs, verändern ... 547
Größe der Schriftart ... 211
Größe von Notensystem verändern ... 171
Größe, Papier ... 461
Größe, Seite ... 461
`grow-direction` ... 86
Grundton eines Akkordes ... 374
Grundton eines Akkords ... 372

H

Halb-B ... 5, 8
Halb-B-Versetzungszeichen, arabische Musik ... 420
halber Takt ... 66
Halbkreuz ... 5, 8
Halboffen ... 640
`halign` ... 214
Hals ... 196
Hals nach oben ... 196
Hals nach unten ... 196
Hals neutral ... 196
Hals, mit Schrägstrich ... 101
Hals, Richtung ... 196
Hals, Richtung von ... 196
Hals, unsichtbar ... 196
Hälse über zwei Systeme ... 292
Haltepedal, Stile ... 294
Harfe ... 299
Harfenpedal ... 299
`harmonic` ... 301
Harmonica Sacra-Notenköpfe ... 37
`harmonicByFret` ... 675
`harmonicByRatio` ... 675
`harmonicNote` ... 675
`harmonicsOn` ... 675
harmonische Obertöne (Flageolett) ... 301
`hbracket` ... 217
`hideKeySignature` ... 361
`hideNotes` ... 193
`hideStaffSwitch` ... 292
Hilfe, Blase ... 197
Hilfslinien, Abstände ... 169
Hilfslinien, Einstellungen ... 169
Hinzufügen von Tönen in Akkorden ... 374
hochgestellt ... 212
hochkant, Papier ... 462
`horizontal-shift` ... 468
`Horizontal_bracket_engraver` ... 200

horizontale Abstände ... 499
horizontale Anordnung ... 497
horizontale Ausrichtung von Text ... 213
horizontale Klammer ... 200
horizontale Notenabstände ... 499
horizontale Notenabstände, Abschnitte definierten ... 499
horizontale Platzierung ... 497
horizontally centering text ... 606
Hufnagel ... 392, 393
`huge` ... 189, 213
hymns ... 274

I

Ictus ... 640
Illustrationen im Text ... 217
immutable-Eigenschaften ... 644
immutable-Objekte ... 644
importing stencils into text ... 637
Improvisation ... 40
`improvisationOff` ... 40, 72
`improvisationOn` ... 40, 72
`inclinatum` ... 407
`include-settings` ... 447
`indent` ... 179, 468, 501
inlining an Encapsulated PostScript image ... 622
`inner-margin` ... 468
inserting music into text ... 628
inserting PostScript directly into text ... 625
inserting URL links into text ... 626
`instrument` ... 433
Instrumentbezeichnungen ... 452
Instrumente, transponierende ... 10
Instrumentenbezeichnung, Notation ... 179
Instrumentenbezeichnungen ... 178
Instrumentenbezeichnungen zu anderen Kontexten hinzufügen ... 180
Instrumentenbezeichnungen, wechseln ... 180
Instrumentengruppe ... 162
Instrumentenwechsel ... 181
`instrumentSwitch` ... 181
`instrumentSwitch` ... 675
interface ... 644
Internals Reference ... 513
`inversion` ... 13
`inversion` ... 675
`ionian` ... 20
Ionisch ... 20
`italic` ... 211

J

Justierung von Notensystemen ... 169
`justified-lines` ... 222
`justify` ... 216
justifying lines of text ... 639
justifying text ... 612

K

Kadenz ... 67
Kadenz und Seitenumbruch ... 67
Kadenz und Zeilenumbruch ... 67
Kadenz, Ausrichten an ... 104
Kadenzen, Bebalkung ... 67
Kapo ... 325
Kasten, Graphik ... 217
`keepWithTag` ... 444
`keepWithTag` ... 675
`key` ... 20
`key` ... 37
`killCues` ... 187
`killCues` ... 675
Kirchenpausen ... 56
Kirchentonarten ... 20
Klammer, Crescendo ... 109
Klammer, erste (Wiederholung) ... 129
Klammer, geschweift ... 162
Klammer, vertikal ... 162
Klammer, Wiederholung ... 134
Klammer, Wiederholung mit Text ... 135
Klammer-Arpeggio über Systeme ... 126
Klammern ... 200
Klammern um Noten ... 195
Klammern um Vorzeichen ... 6
Klammern, Analyse ... 200
Klammern, Crescendo, schräg ... 552
Klammern, Graphik ... 217
Klammern, spitze ... 142
Klammern, unterschiedliche Größen ... 223
Klammern, Verschachteln ... 166
Klang ... 451
Klavier, Pedalbezeichnung ... 294
Klavier-Versetzungszeichenstil ... 27
Klavier: Warnungsversetzungszeichen ... 28
Klaviermusik, zentrierte Dynamik ... 288
Klaviersystem ... 162, 288
kleinere Noten ... 188
Knall-Pizzicato ... 302
Kollisionen, vertikal, vermeiden ... 496
Kombinieren von Stimmen ... 154
Komma-Intervalle ... 425
Komprimieren von Noten ... 47
Kontexte erstellen ... 515, 516
Kontexte, am Leben erhalten ... 516
Kontexte, einmal verändern ... 532
Kontexte, Lebensdauer ... 516
Kontexte, neue definieren ... 522
Kontexte, Reihenfolge ... 539
Kontexte, verschachtelt ... 524
Kontextveränderungen rückgängig machen ... 532
Kontroll-Tonhöhe ... 9
Kontrollpunkte und tweak ... 534
Kontrollpunkte, Bézier-Kurven ... 559
Kopfzeile ... 437
Krebs ... 13
Kreuz ... 5
Kreuznotenköpfe ... 34
künstliches Flageolett ... 302
kurze Instrumentenbezeichnungen ... 178

L

`label` ... 675
Laissez vibrer ... 49
`laissezVibrer` ... 49
`landscape` ... 462

Länge von Zeilen ... 501
language ... 675
languageRestore ... 675
languageSaveAndChange ... 675
large ... 189, 213
larger ... 211, 213
last-bottom-spacing ... 466
Lautstärke ... 109
layer (Ebenen) ... 547
Layout der Seite ... 437
layout file ... 472
layout objects ... 643
Layout, Partitur ... 471
Layout-Schnittstelle ... 526
Layoutobjekte ... 643
leere Systeme verstecken ... 175
Leerzeichen ... 433
Leerzeichen, Gesangstext ... 227, 234
left aligning text ... 614
left-align ... 213
left-margin ... 467
Legatobögen ... 116
Legatobogen zur Phrasierung ... 119
Legatobogen, gepunktet ... 116
Legatobogen, gestrichelt ... 116
Legatobögen, manuelle Platzierung ... 116
Legatobogen, massiv ... 116
Legatobögen, verändern ... 559
Legatobogen-Stil ... 116
length ... 292
lexer ... 644
Ligaturen ... 394
Ligaturen der quadratischen Neumennotation ... 407
Ligaturen, weiße Mensuralnotation ... 402
ligatures in text ... 607
line-width ... 466, 501
linea ... 407
Linien zwischen Systemen ... 198
Linien, Gitter ... 198
Liste der Farben ... 577
Liste der vorhandenen Schriftarten ... 225
Literatur ... 379, 383
locrian ... 20
Lokrisch ... 20
longa ... 41
longa ... 51
Longa-Pause ... 51
lower ... 214
lowering text ... 614
ly:minimal-breaking ... 477
ly:optimal-breaking ... 476
ly:page-turn-breaking ... 476
lydian ... 20
Lydisch ... 20
lyrics und tweak-Befehl ... 534
lyricsto ... 231

M

m ... 373
magnify ... 211
magnifying text ... 600
magstep ... 189, 540
maj ... 373
major ... 20
major seven symbols ... 380
majorSevenSymbol ... 380
makam ... 425
makamlar ... 425
make-dynamic-script ... 114
make-pango-font-tree ... 225
makeClusters ... 145
makeClusters ... 675
makeDefaultStringTunings ... 675
makeStringTuning ... 318
makeStringTuning ... 675
Manuals ... 1
manuelle Balken ... 84
manuelle Balken, Richtung zuweisen ... 84
manuelle Balken, Verzierungen ... 84
manuelle Systemwechsel ... 289
manuelle Taktstriche ... 88
manuelle Wiederholungszeichen ... 134
manuelles Übungszeichen ... 98
Maqam ... 420
Marcato ... 106, 640
mark ... 97, 204
Marke ... 444
Markieren von Abschnitten ... 97
markierte Noten behalten ... 444
markierte Noten entfernen ... 444
markup ... 204, 208, 209, 210
markup, Syntax ... 210
markup-markup-spacing ... 466
markup-system-spacing ... 465
markuplines ... 209, 222
massiver Legatobogen ... 116
Matrize (stencil) ... 646
Matrize, entfernen ... 546
max-systems-per-page ... 468
maxima ... 41
maxima ... 51
Maxima-Pause ... 51
measureLength ... 77
measureLength ... 105
measurePosition ... 66
measurePosition ... 105
Medicaea, Editio ... 392, 393
mehre Dynamikzeichen an einer Note ... 110
mehrere Phrasierungsbögen ... 119
mehrere Stimmen ... 149
mehrfache Bögen ... 116
mehrnotiger Vorschlag ... 103
mehrseitiger Text ... 222
Mehrstimmigkeit ... 146
Mehrstimmigkeit, ein System ... 146
Mehrtaktpause mit Fermate ... 55
Mehrtaktpausen ... 51, 54
Mehrtaktpausen und Fingersatz ... 58
Mehrtaktpausen, ausschreiben ... 55
Mehrtaktpausen, Beschriftung ... 55
Mehrtaktpausen, komprimieren ... 55
Mehrtaktpausen, Positionierung ... 57
Mehrtaktpausen, Text hinzufügen ... 55
mehrzeiliger Text ... 215
Melisma ... 235, 238
Melismen, Balken ... 75
Melodierhythmus: Anzeige ... 72
Melodietransformation, Krebs ... 13

Melodietransformation, Umkehrung 13
Melodietransformationen, modal 14
Melodietransposition, modal 14
Melodieumkehrung, modal 15
Mensur 397
Mensuralligaturen 402
Mensuralmusik, Transkription 165
Mensuralnotation 392
`MensuralStaff` 161, 395
MensuralStaffContext 395
Mensuralstil 393
`MensuralVoice` 395
MensuralVoiceContext 395
Mensurstriche 165
`mergeDifferentlyDottedOff` 149
`mergeDifferentlyDottedOn` 149
`mergeDifferentlyHeadedOff` 149
`mergeDifferentlyHeadedOn` 149
merging text 607
`meter` 433
Metronombezeichnung 63
Metrum 58
Metrum, Noten ohne 67, 105
Metrum, polymetrisch 68
Mezzosopranschlüssel 16
`mf` 109
MIDI 22, 451
MIDI und Wiederholungen 454
MIDI, Akkordsymbole 454
MIDI, Mikrotöne 454
MIDI, Rhythmen 454
MIDI, Tonhöhen 454
MIDI, Vierteltöne 454
MIDI-Instrumentenbezeichnungen 576
MIDI-Kontextdefinitionen 453
MIDI-Transposition 22
MIDI-Umgebung 453
Mikrotöne 5, 8
Mikrotöne in MIDI 454
`min-systems-per-page` 468
`minimum-Y-extent` 482
`minimumFret` 306, 342
`minimumPageTurnLength` 477
`minimumRepeatLengthForPageTurn` 477
`minor` 20
mirroring markup 625
`mixed` 294
`mixolydian` 20
Mixolydisch 20
modale Transformationen 14
modale Transposition 14
modale Umkehrung 15
`modalInversion` 15
`modalInversion` 675
`modalTranspose` 14
`modalTranspose` 675
`modern` 26
`modern-cautionary` 26
`modern-voice` 27
`modern-voice-cautionary` 27
modern-Warnung-Versetzungszeichenstil 26
moderne Versetzungszeichen 27
Moderner Stil, Versetzungszeichen 26
moderner Tabulatur-Schlüssel 320
moderner Versetzungszeichenstil 26, 27
moderner Versetzungszeichenstil mit Warnungen .. 26
moderner Versetzungszeichenstil mit Warnungen für Stimmen 27
`moderntab` 320
Modi, in Akkorden 372
Modifikatoren, Akkorde 372
Modus 20
Moll 20
Mordent 106, 640
`mp` 109
`MultiMeasureRestText` 55
Musica ficta 401
`musicglyph` 98
`musicMap` 675
Musik komprimieren 47
Musik ohne Metrum, Umbrüche 67
Musikanalyse 200
Musikbuchstaben 98
Musikobjekte, Einfügen 219
musikwissenschaftliche Analyse 200
mutable-Objekte 644

N

N-tole, Formatierung 43
N-tolen 43
N.C.-Symbol 377
Nachschlag 99
`name` 522
Name von Sänger 256
Namen von Figuren 267
`neo-modern` 28
`neo-modern-cautionary` 28
neo-modern-cautionary-Versetzungszeichenstil 28
`neo-modern-voice` 28
`neo-modern-voice-cautionary` 29
neo-moderner Versetzungszeichenstil 28
neo-moderner Versetzungszeichenstil pro Stimme 28
neo-moderner Versetzungszeichenstil pro Stimme mit Warnungen 29
Neomensuralstil 393
neue Dynamikzeichen 114
neue Kontexte 515
neues Notensystem 161
`new` 515
nicht metrische Musik, Umbrüche 67
Nicht-ASCII-Zeichen 447
Nicht-Textschriftarten in Beschriftung 223
nichtmusikalische Symbole 218
niente, al 112
`no-reset` 29
`noBeam` 84
`nonstaff-nonstaff-spacing` 482
`nonstaff-relatedstaff-spacing` 482
`nonstaff-unrelatedstaff-spacing` 482
`noPageBreak` 675
`noPageTurn` 675
normale Wiederholung 129
`normalsize` 189, 213
Notation für Streicher 300
Notation innerhalb von Beschriftung 221
Notation innerhalb von Text 221
Notation, Aiken 37

Notation, Erklärungen ... 197
Notation, graphische ... 218
Notationsobjekte, Einfügen ... 219
note-event ... 184
`Note_heads_engraver` ... 71
Noteknöpfe, einfache Notation ... 35
Noten ausdehnen ... 47
Noten in Klammern ... 195
Noten komprimieren ... 47
Noten ohne Metrum ... 105
Noten ohne Takt ... 67, 105
Noten verkleinern ... 188
Noten verschmelzen ... 149
Noten verstecken ... 193
Noten wiederholt schreiben ... 136
Noten, Aufteilen ... 71
Noten, doppelpunktiert ... 41
Noten, durchsichtig ... 193
Noten, farbig ... 194
Noten, farbige in Akkorden ... 195
Noten, kleiner ... 188
Noten, parlato ... 34
Noten, punktiert ... 41
Noten, Schriftgröße ... 189
Noten, Stichnoten ... 185
Noten, transponieren ... 10
Noten, unsichtbar ... 193
Noten, Wechsel zwischen Systemen ... 289
Noten-Schriftzeichen ... 98
Notenabstände, Abschnitte definieren ... 499
Notenabstände, horizontal ... 499
Notenbezeichnungen, arabisch ... 420
Notenbezeichnungen, Deutsch ... 5
Notenbezeichnungen, Holländisch ... 5
Notenbezeichnungen, Standard ... 5
Notenbezeichungen, andere Sprachen ... 7
Notencluster ... 145
Notendauer, Standard ... 41
Noteneingabe: relative Oktavbestimmung ... 2
Notengruppenklammer ... 200
Notenhals, durchgestrichen ... 101
Notenhals, Richtung ... 196
Notenhals, Richtung von ... 196
Notenhals, unsichtbar ... 196
Notenhälse über zwei Systeme ... 292
Notenkopfarten ... 597
Notenköpfe ... 189
Notenköpfe für Anfänger ... 35
Notenköpfe zum Lernen ... 35
Notenköpfe, besondere ... 34
Notenköpfe, Christian Harmony ... 37
Notenköpfe, Flageolett ... 34
Notenköpfe, Formen ... 37
Notenköpfe, Funk ... 37
Notenköpfe, Gitarre ... 34
Notenköpfe, Harmonica Sacra ... 37
Notenköpfe, Improvisation ... 40
Notenköpfe, Kreuz ... 34
Notenköpfe, Mensuralnotation ... 398
Notenköpfe, Raute ... 34
Notenköpfe, rautenförmig ... 301
Notenköpfe, sacred harp ... 37
Notenköpfe, Übung ... 35
Notenköpfe, Walker ... 37
Notenlänge ... 41
Notenlinien, Anzahl ... 169
Notenlinien, beenden ... 170
Notenlinien, beginnen ... 170
Notenlinien, Dicke ... 169
Notenlinien, Einstellungen ... 169
Notenlinien, Erstellen ... 169
Notenschlüssel ... 16
Notensystem beginnen ... 169
Notensystem stoppen ... 169
Notensystem, anpassen ... 540
Notensystem, beenden ... 170
Notensystem, Größe verändern ... 171
Notensystem, Klavier ... 288
Notensystem, neu ... 161
Notensystem, Tasteninstrumente ... 288
Notensystemabstand ... 482
Notensysteme in Text einfügen ... 221
Notensysteme, gruppieren ... 162
Notensysteme, mehrere ... 162
Notensysteme, Modifikation ... 169
Notensystemgruppe ... 162
Notenzusammenstöße ... 149
notes within text by log and dot-count ... 627
notes within text by string ... 628
`null` ... 214
`numericTimeSignature` ... 59
Nummerierung von Saiten ... 303
Nummerierung von Takten ... 92
Nummerierung, Strophen ... 255
nur Text ... 208

O

oberste Ebene, Text ... 208
Objekte verändern ... 547
Objekte, Drehen ... 552
Objekte, einfärben ... 547
Objekte, farbig ... 194
Objekte, Graphik im Text ... 217
Objekte, Sichtbarkeit ... 546
`octaveCheck` ... 9
`octaveCheck` ... 676
`oddFooterMarkup` ... 437
`oddHeaderMarkup` ... 437
offen ... 106
Offen ... 640
Offene Saite, anzeigen ... 301
Oktavbestimmung, relativ ... 2
Oktavenmodus (relativ) und Akkorde ... 4
Oktavenüberprüfung ... 9
oktavierte Schlüssel, Sichtbarkeit ... 550
Oktavierung ... 21
Oktavierungskorrektur ... 9
Oktavtransposition ... 17
Oktavwechsel: Tonhöhe ... 1
`once` ... 530, 532
`oneVoice` ... 146
Optimieren ... 533
Optimierung innerhalb einer Variable ... 534
`Optischer Ausgleich` ... 498
`opus` ... 433
Oratorium ... 263
Orchester, Streicher ... 300
Orgelpedal-Bezeichnung ... 106
Orgelpedalbezeichnung ... 640

oriscus ... 407
Ornament ... 106
Ornamente ... 99
Osmanische Musik ... 425
ossia ... 176
Ossia ... 171, 524
Ossia-Systeme ... 171
ottava ... 21
ottava ... 676
outer-margin ... 468
output-def ... 644
outside-staff-horizontal-padding ... 496
outside-staff-padding ... 496
outside-staff-priority ... 496
override ... 531
Override im Gesangstextmodus ... 227
override rückgängig machen ... 532
overrideProperty ... 676
overrideTimeSignatureSettings ... 59
overrideTimeSignatureSettings ... 676
overriding properties within text markup ... 636

P

p ... 109
pad-around ... 217
pad-markup ... 217
pad-to-box ... 217
pad-x ... 217
pädagogische Notenköpfe ... 35
padding text ... 615
padding text horizontally ... 616
page-breaking ... 469
page-breaking-system-system-spacing ... 469
page-count ... 469
page-spacing-weight ... 470
pageBreak ... 676
pageTurn ... 676
palmMute ... 676
palmMuteOn ... 676
Pango ... 223
paper-height ... 463
paper-width ... 466
Papier, Ausrichtung ... 462
Papierformat ... 462
Papiergröße ... 461
Parallele Notation, Eingabe ... 158
parallelMusic ... 158
parallelMusic ... 676
parenthesize ... 195
parenthesize ... 676
Parlato ... 273
Parlato-Notenköpfe ... 34
parser ... 645
partcombine ... 154
partcombine ... 676
partcombineForce ... 676
partial ... 66
partieller Takt ... 66
Partitur ... 162
Partitur, Layout ... 471
paths, drawing ... 624
Pausen ... 51
Pausen verschieben, automatisch ... 149
Pausen, Aufteilen ... 71
Pausen, Ganztakt- ... 54
Pausen, ganztaktig ... 51
Pausen, Kirchenstil ... 56
Pausen, mehrere Takte ausschreiben ... 55
Pausen, mehrere Takte komprimieren ... 55
Pausen, Mehrtakt- ... 54
Pausen, mehrtaktig ... 51
Pausen, Mensuralnotation ... 400
Pausen, unsichtbar ... 53
Pausen, vertikale Position festlegen ... 51
Pausen, Zusammenfalten ... 58
Pausen, Zusammenstöße ... 58
Pausendauern ... 51
Pausenzeichen ... 120
Pedal, Harfe ... 299
Pedal, sostenuto ... 294
Pedal-Bezeichnung ... 106
Pedalbezeichnung ... 294
Pedalbezeichnung, Klammer ... 294
Pedalbezeichnung, Stile ... 294
Pedalbezeichnung, Text ... 294
Pedaldiagramme, Harfe ... 299
pedalSustainStyle ... 294
percent ... 138
Percussionsnotensystem ... 161
Perkussion ... 349, 351
Perkussionsnotensystem ... 161
Petrucci ... 392
Petrucci-Stil ... 393
Phrasierung, Gesang ... 235
Phrasierungsbögen ... 116, 119
Phrasierungsbögen, gepunktet ... 119
Phrasierungsbögen, gestrichelt ... 119
Phrasierungsbögen, gleichzeitig ... 119
Phrasierungsbogen, halb durchgehend, halb gestrichelt ... 119
Phrasierungsbögen, mehrfach ... 119
Phrasierungsbogen, Strichelmuster definieren ... 119
Phrasierungsklammern ... 200
Phrasierungszeichen ... 119
phrasingSlurDashed ... 119
phrasingSlurDashPattern ... 119
phrasingSlurDashPattern ... 676
phrasingSlurDotted ... 119
phrasingSlurDown ... 119
phrasingSlurHalfSolid ... 119
phrasingSlurNeutral ... 119
phrasingSlurSolid ... 119
phrasingSlurUp ... 119
phrygian ... 20
Phrygisch ... 20
piano ... 27
Piano, Pedalbezeichnung ... 294
piano-cautionary ... 28
Piano-System ... 288
Piano-Versetzungszeichenstil ... 27
PianoStaff ... 288, 290
piece ... 433
pipeSymbol ... 96
Pitch_squash_engraver ... 72
pitchedTrill ... 127
pitchedTrill ... 676
Pizzicato, Bartók ... 302
Pizzicato, Knall- ... 302

placing horizontal brackets around text 623
placing parentheses around text 623
placing vertical brackets around text 621
Platz innerhalb von Systemgruppen 482
Platz um Text 217
Platz zwischen Notensystemen 482
Platzhalternoten 53
Platzierung von Gesangstext 240
Platzierung, Layouteinstellungen 509
PNG-Ausgabe 450
poet 433
pointAndClickOff 677
pointAndClickOn 677
Polymetrische Notation und Balken 68
polymetrische Partitur 520
polymetrische Taktarten 68
Polyphonie 146, 149
Polyphonie, ein System 146
Portato 106, 640
Position und Barré für bundierte Saiteninstrumente 345
Position und Barré für Bundinstrumente 345
Position von Mehrtaktpausen 57
Positionierung, vertikal 481
postscript 219
Postscript, Graphik 219
Powerakkorde 346
powerChords 346
Powerchords 346
pp 109
ppp 109
pppp 109
ppppp 109
Praller 106, 640
Prallermordent 640
predefinedFretboardsOff 341
predefinedFretboardsOn 341
Prima volta 129
print-all-headers 436, 470
print-first-page-number 470
print-page-number 470
Prozent-Wiederholungen 138
psalms 274
Punktierung 41
putting space around text 615

Q

Quadratische Neumenligaturen 407
Quelldatei, Struktur 431
quer, Papier 462
quilisma 407
quotedCueEventTypes 184
quotedEventTypes 184
quoteDuring 181, 185
quoteDuring 677

R

r 51
R 54
ragged-bottom 464
ragged-last 467, 501
ragged-last-bottom 464
ragged-right 467, 501
Rahmen, Text 217
railroad tracks 121
raise 214
raising text 616
Rand um Text 217
Rand, überhängender Text 202
Ratisbona, Editio 393
rautenförmige Notenköpfe 301
Rautennotenköpfe 34
rechte Hand, Fingersatz für bundierte Saiteninstrumente 343
rechte Hand, Fingersatz für Bundinstrumente 343
referencing page labels in text 639
referencing page numbers in text 636
Referenz der Interna 513
regelmäßige Zeilenumbrüche 475
Relativ 2
relative 2, 5, 13, 291
Relative Oktavbestimmung 2
relative Tonhöhe, Akkorde 143
relativer Modus und Akkorde 4
relativer Modus und automatischer Systemwechsel 291
Relativer Oktavenmodus und Transposition 5
religious music 274
RemoveEmptyStaves 175
RemoveEmptyStaves 176
removeWithTag 444
removeWithTag 677
Renaissancemusik 165
repeatCommands 134
repeatTie 48
repetitive Musik 136
resetRelativeOctave 677
rest 51
rest-event 184
retrograde 13
retrograde 677
revert 532
revertTimeSignatureSettings 60
revertTimeSignatureSettings 677
rfz 109
rgb-color 194
RGB-Farbe 194
Rhythmen in MIDI 454
RhythmicStaff 161
Rhythmische Aufteilungen 43
rhythmisches Notensystem 161
Rhythmus der Melodie anzeigen 72
Richtung von Notenhälsen 196
right aligning text 616
right-align 213
right-margin 467
rightHandFinger 343
rightHandFinger 677
rotating text 617
rounded-box 217
rückgängig machen von Kontextveränderungen 532

S

s 53
Sackpfeife 361

sacred harp-Notenköpfe ... 37
sacredHarpHeads ... 37
sacredHarpHeadsMinor ... 37
Saite, offen ... 301
Saitenstimmung für Bundinstrumente ... 317
Saitenzahl ... 303
Sängername ... 256
SATB ... 263
Sätze, mehrere ... 428
Satzzeichen ... 227
scalable vector graphics output ... 450
scaleDurations ... 47, 68
scaleDurations ... 677
scaling markup ... 625
scaling text ... 618
Schachtelung von Systemen ... 166
Scheme objekt ... 646
Schlaggruppen ... 81
Schlagrhythmus, Gitarre ... 72
Schlagzeug ... 349, 351
schließende Taktstriche ... 87
Schluss, alternativer in Wiederholung ... 129
Schlüssel ... 5, 16
Schlüssel Alter Musik ... 16
Schlüssel, C ... 16
Schlüssel, F ... 16
Schlüssel, G ... 16
Schlüssel, greg. Choral ... 404
Schlüssel, Mensuralnotation ... 396
Schlüssel, modern, Tabulatur ... 320
Schlüssel, Sichtbarkeit der Oktavierung ... 550
Schlüssel, Sichtbarkeit nach expliziter Änderung ... 549
Schlüssel, transponierend ... 17
Schnittstelle von graphischen Objekten ... 644
Schnittstelle, Layout- ... 526
Schottischer Dudelsack ... 361
schräge Crescendoklammern ... 552
schräge Notenköpfe ... 40
Schriftart verändern ... 211
Schriftarten, für das gesamte Dokument ändern ... 225
Schriftarten, Hintergrundinformation ... 223
Schriftarten, Liste zum Auswählen ... 225
Schriftarten, Nicht-Text in Beschriftung ... 223
Schriftarten, vorhandene auflisten ... 225
Schriftartenfamilien, definieren ... 225
Schriftfamilie ... 643
Schriftfamilien ... 212
Schriftgröße ... 211
Schriftgröße (Notation) ändern ... 189
Schriftgröße (Notation), Standard ... 190
Schriftgröße, Einstellung ... 472
Schriftschnitt verändern ... 211
Schriftschnitte ... 212
Schriftzeichen, Notenschrift ... 98
score-markup-spacing ... 465
score-system-spacing ... 466
scoreTitleMarkup ... 436
Seconda volta ... 129
segno ... 89
Segno ... 98, 106, 640
Segno an Taktstrich ... 204
Seitegröße ... 461
Seitenformat ... 462
Seitenlayout ... 437
Seitenrand, überhängender Text ... 202
Seitenumbruch, erzwingen ... 433
Seitenumbrüche ... 475, 501
Seitenumbrüche in Kadenzen ... 67
Seitenumbrüche in nicht metrischer Musik ... 67
Seitenzahlen, automatische Nummerierung ... 470
Seitenzahlen, unterdrücken ... 470
self-alignment-X ... 482
Semai-Form ... 422
Semicirculus ... 640
separater Text ... 208
Septakkorde ... 372
sesqui-B ... 8
sesqui-Kreuz ... 8
set ... 77, 529
set-accidental-style ... 24
set-octavation ... 21
setting extent of text objects ... 639
setting horizontal text alignment ... 610
setting subscript in standard font size ... 601
setting superscript in standard font size ... 601
Setzen von Sonderzeichen ... 210
Setzen von Text ... 210
sf ... 109
sff ... 109
sfz ... 109
shiftDurations ... 677
shiftOff ... 149
shiftOn ... 149
shiftOnn ... 149
shiftOnnn ... 149
short-indent ... 179, 468
show-available-fonts ... 225
showFirstLength ... 449
showKeySignature ... 361
showLastLength ... 449
showStaffSwitch ... 292
Sichtbarkeit von Objekten ... 546
Sichtbarkeit von oktavierten Schlüsseln ... 550
Sietenzahlen, erste definieren ... 470
signum congruentiae ... 640
Silben spreizen ... 245
simple text strings ... 603
simple text strings with tie characters ... 630
simultane Noten und Versetzungszeichen ... 30
skalierbare Vektorgraphik-Ausgabe ... 450
Skalieren von Dauern ... 47
skip ... 53
Skip ... 53
skipTypesetting ... 449
slashed digits ... 637
Slide in Tabulaturen ... 309
slurDashed ... 116
slurDashPattern ... 117
slurDashPattern ... 677
slurDotted ... 116
slurDown ... 116
slurHalfDashed ... 116
slurHalfSolid ... 116
slurNeutral ... 116
slurSolid ... 116
slurUp ... 117
small ... 189, 213
smaller ... 211, 213
smob ... 646

Solesmes ... 392
solo-Stellen ... 154
Sonderzeichen in Textbeschriftungen ... 210
Sopranschlüssel ... 16
Sopranschlüssel in C ... 16
sos. ... 294
sostenuto-Pedal ... 294
`sostenutoOff` ... 294
`sostenutoOn` ... 294
Southern-Harmony-Notenköpfe ... 37
`southernHarmonyHeads` ... 37
`southernHarmonyHeadsMinor` ... 37
`sp` ... 109
`spacing` ... 498
`spacingTweaks` ... 677
spitze Klammern ... 142
`spp` ... 109
Sprache, Tonhöhenbezeichnungn in anderer ... 7
Sprechgesang ... 273
Spreizen von Silben ... 245
Springen zwischen Systemen ... 289
Staccatissimo ... 106
Staccato ... 106, 640
stacking text in a column ... 607
Staff symbol, Erstellen ... 169
`staff-affinity` ... 482
`staff-staff-spacing` ... 482
`Staff.midiInstrument` ... 452
`Staff_symbol_engraver` ... 175
`staffgroup-staff-spacing` ... 482
Standard Notendauer ... 41
Standard-Schriftgröße (Notation) ... 190
Standard-Versetzungszeichenstil ... 24, 25
Standardnotenbezeichnungen ... 5
Standardnotendauer ... 41
Standardtakteinstellungen ... 59
Standardtaktstrich, Änderung ... 92
`start-repeat` ... 134
`startGroup` ... 200
`startStaff` ... 170, 171
`startTrillSpan` ... 126
`Stem` ... 292
`stem-spacing-correction` ... 498
`stemDown` ... 196
`stemLeftBeamCount` ... 85
`stemNeutral` ... 196
Stempel (stencil), entfernen ... 546
`stemRightBeamCount` ... 85
`stemUp` ... 196
stencil ... 646
stencil, entfernen ... 546
Stichnoten ... 181, 185, 269
Stichnoten innerhalb von rhythmischer Kombination ... 46
Stichnoten, entfernen ... 187
Stichnoten, Formatierung ... 185
Stil von Legatobögen ... 116
Stil von Taktangaben ... 59
Stil von Übungszeichen ... 98
Stile, Notenköpfe ... 34, 597
Stile, Stimmen ... 149
Stimme ... 146
Stimme folgen ... 292
Stimme-Versetzungszeichenstil ... 26
Stimmen kombinieren ... 154
Stimmen verschieben ... 149
Stimmen, \autoBeamOff und \partcombine ... 76
Stimmen, farbige Unterscheidung ... 149
Stimmen, geteilt ... 265
Stimmen, mehrere ... 149
Stimmen, Stile ... 149
Stimmen, Versetzungszeichen für ... 27
Stimmen, Versetzungszeichenstil mit Warnung für Stimmen ... 27
Stimmen, zitieren ... 181
Stimmfolgestriche ... 292
Stimmgruppe ... 162
Stimmkreuzung ... 292
Stimmumfang ... 31
Stimmung, Banjo ... 348
`stopGroup` ... 200
`stopStaff` ... 170, 171, 175
`stopTrillSpan` ... 126
`storePredefinedDiagram` ... 333
`storePredefinedDiagram` ... 677
Strecker, Text ... 202
Strecker, Text-, Formatierung ... 203
Streicher ... 300
Streicher, Bogenanzeige ... 301
Striche zur Stimmverfolgung ... 292
Striche: Notenköpfe ... 40
Strichnotenköpfe ... 40
`stringTunings` ... 330
`StringTunings` ... 317
`stropha` ... 407
Strophennummer ... 255
Struktur, Datei ... 431
`styledNoteHeads` ... 677
`sub` ... 212
Subbassschlüssel ... 16
`subdivideBeams` ... 80
subscript text ... 604
`subsubtitle` ... 433
`subtitle` ... 433
Subtraktion in Akkorden ... 374
`suggestAccidentals` ... 401
`super` ... 212
superscript text ... 604
`sus` ... 375
`sustainOff` ... 294
`sustainOn` ... 294
SVG-Ausgabe ... 450
Symbole auf der Taktstrich ... 204
Symbole, Akkord- ... 376
Symbole, Akkordeon ... 295
Symbole, nicht musikalische ... 218
Synchronisation von Verzierungen ... 103
System querende Hälse ... 292
System, beenden ... 170
System, Chor ... 162
System, geschachtelt ... 166
System, Größe verändern ... 171
`system-count` ... 469
`system-separator-markup` ... 470
`system-system-spacing` ... 466
System-Trennzeichen ... 168
SystemBeginnBegrenzer, geschachtelt ... 166
Systeme verstecken ... 175
Systeme, leere ... 175
Systeme, mehrere ... 162

Systeme, Tremolo zwischen ... 142
Systeme, Zusammenstöße beim Stimmenwechsel ... 289
Systemgröße, Einstellung ... 472
Systemgruppe ... 162
Systemgruppen, Abstände innerhalb ... 482
Systemgruppen, Verschachtelung ... 166
`systems-per-page` ... 469
Systemwechsel von Stimmen ... 292
Systemwechsel, automatisch ... 290
Systemwechsel, manuell ... 289

T

`tabChordRepetition` ... 677
`tabFullNotation` ... 306
`TabStaff` ... 161, 305
Tabulatur ... 161, 303
Tabulatur und Flageolett ... 309
Tabulatur, Banjo ... 317, 348
Tabulatur, Bassgitarre ... 317
Tabulatur, Bratsche ... 317
Tabulatur, Cello ... 317
Tabulatur, eigene Saitenstimmung ... 318
Tabulatur, Geige ... 317
Tabulatur, Gitarre ... 317
Tabulatur, Grundlegendes ... 305
Tabulatur, Kontrabass ... 317
Tabulatur, Mandoline ... 317
Tabulatur, moderner Schlüssel ... 320
Tabulatur, Saitenstimmung ... 317
Tabulatur, Ukulele ... 317
Tabulaturen und Gleiten ... 309
Tabulaturen, eigen ... 317
Tabulatursystem ... 161
`TabVoice` ... 305
`tag` ... 444
`tag` ... 677
Tag ... 444
`tagline` ... 434
Tagline ... 436
Takt unterteilen ... 81
Takt, Noten ohne ... 105
Taktangabe ... 58
Taktangabe, Sichtbarkeit ... 58
Taktangaben-Stile ... 59
Taktart, Mensuralnotation ... 397
Taktart, Noten ohne ... 67
Taktart, Standardeigenschaften wiederherstellen ... 60
Taktart, Standardeinstellung ... 59
Taktarten, arabisch ... 422
Taktarten, mehrere in Partitur ... 520
Taktarten, polymetrisch ... 68
Taktarten, unterschiedliche per System ... 520
Taktartensymbole, doppelt ... 68
Taktartensymbole, unterteilt ... 68
Takte verkürzen ... 66
Taktgruppen ... 81
Taktlänge ändern ... 66
Taktlinie, manuell ... 88
Taktlinie, Wiederholung ... 134
Taktlinien ... 87
Taktlinien, Ausrichtung ... 94
Taktlinien, ausschalten ... 67
Taktlinien, unsichtbar ... 88
Taktlose Musik, Bebalkung ... 67
Taktnummer ... 105
Taktnummer, Form ... 93
Taktnummern ... 92
Taktnummern, ausschalten ... 67
Taktnummern, Zusammenstöße ... 96
Taktposition und Wiederholung ... 133
Taktschläge gruppieren ... 81
Taktstrich, doppelt ... 87
Taktstrich, Symbole anfügen ... 204
Taktstriche ... 87
Taktstriche, Änderung von Standard ... 92
Taktstriche, manuell ... 88
Taktstriche, schließend ... 87
Taktstriche, unsichtbar ... 88
Taktstriche, unterdrücken ... 550
Taktüberprüfung ... 96
Taktweise Wiederholungen ... 138
Taktzahlen ... 92
Taktzahlen, gleichmäßige Abstände ... 93
`taor` ... 361
taqasim ... 422
Tasteninstrumente, Notensystem ... 288
Tasteninstrumente, zentrierte Dynamik ... 288
`teaching` ... 29
teaching-Versetzungszeichenstil ... 29
`teeny` ... 189, 213
`tempo` ... 63
Tempo ... 63
Tempobezeichnung ... 63
Tempobezeichnungen innerhalb von N-tolen-Klammern ... 46
Tenorschlüssel ... 16
Tenorschlüssel, Chor ... 17
Tenuto ... 106, 640
`text` ... 294
Text alleine ... 208
Text am Taktstrich ... 204
Text auf der Seite zentrieren ... 215
Text außerhalb des Randes ... 202
text columns, left-aligned ... 614
text columns, right-aligned ... 617
Text einrahmen ... 217
Text in Voltaklammer ... 135
Text mit Sonderzeichen ... 210
Text über mehrere Seiten ... 222
Text über Mehrtaktpausen ... 55
Text und Balken ... 77
Text verzieren ... 217
Text, an Melodie ausgerichtet ... 228
Text, andere Sprachen ... 201
Text, Ausrichtung ... 213
Text, Blocksatz ... 216
Text, horizontale Ausrichtung ... 213
Text, innerhalb des Randes behalten ... 202
Text, mehrere Zeilen ... 215
Text, Notation innerhalb ... 221
Text, oberste Ebene ... 208
Text, Rand außen ... 217
Text, Syntax ... 210
Text, vertikale Ausrichtung ... 214
Textarten ... 201
textartige Zeichen ... 204
Textausrichtung, Befehle ... 217

Textausrichtungsbefehle ... 217
Textbeschriftung ... 210
Textbeschriftung ausrichten ... 213
Textbeschriftung über mehrere Seiten ... 222
Textbeschriftung, Blocksatz ... 216
Textbeschriftung, mehrzeilig ... 215
Textbeschriftung, Notationsobjekte einfügen ... 219
Textbeschriftung, Sonderzeichen ... 210
Textbeschriftungs-Ausdrücke ... 210
Textblasen ... 197
Textblöcke ... 215
Textelemente, nicht leer ... 201
Textgröße ... 211
`textLengthOff` ... 56
`textLengthOff` ... 202
`textLengthOn` ... 56
`textLengthOn` ... 202
`textSpannerDown` ... 203
`textSpannerNeutral` ... 203
`textSpannerUp` ... 203
Textstrecker ... 202
Textstrecker, Formatierung ... 203
Textzeichen ... 204
`thumb` ... 191
thumb-script ... 191
`tieDashed` ... 49
`tieDashPattern` ... 677
`tieDotted` ... 49
`tieDown` ... 49
tiefergestellt ... 212
`tieNeutral` ... 49
ties, placement ... 49
`tieSolid` ... 49
`tieUp` ... 49
`time` ... 58, 77
time signature, compound ... 62
`times` ... 43
`times` ... 68
`timeSignatureFraction` ... 68
`tiny` ... 189, 213
Titel ... 437
`title` ... 433
`tocItem` ... 677
Tonart ... 5, 20
Tonart, Mensuralnotation ... 400
Tonart, Sichtbarkeit nach expliziter Änderung ... 549
Tonarten, greg. Choral ... 405
Tonhöhe: Wechsel der Oktave ... 1
Tonhöhen in MIDI ... 454
Tonhöhen, transponieren ... 10
Tonhöhenbezeichnungen ... 1
Tonhöhenbezeichnungen, andere Sprachen ... 7
`Top` ... 1
`top-margin` ... 463
`top-markup-spacing` ... 466
`top-system-spacing` ... 466
Transformationen, modal ... 14
Transkription von Mensuralmusik ... 165
`translate` ... 215
`translate-scaled` ... 215
translating text ... 617, 618
transparent, Noten ... 193
transparent, Objekte ... 546
Transponieren ... 10
transponierende Instrumente ... 10
transponierende Schlüssel ... 17
Transponierendes Instrument ... 22
`transpose` ... 5, 10, 13
`transposedCueDuring` ... 188
`transposedCueDuring` ... 678
`transposition` ... 22, 181
`transposition` ... 678
Transposition ... 10
Transposition und relativer Modus ... 5
Transposition von Bunddiagrammen ... 331
Transposition, Instrumente ... 22
Transposition, MIDI ... 22
Transposition, modal ... 14
tre corde ... 294
`treCorde` ... 294
`tremolo` ... 140
Tremolo ... 140
Tremolo über Systeme ... 142
Tremolobalken ... 140
`tremoloFlags` ... 141
Tremolozeichen ... 141
Trennstriche, Gesangstext ... 238
Trennzeichen ... 168
`triangle` ... 218
`trill` ... 126
Triller ... 106, 126, 640
Triller mit Tonhöhe ... 127
Triller mit Tonhöhe und Versetzungszeichen ... 127
Triole, Formatierung ... 43
Triolen ... 43
Triolenklammer, Platzierung ... 43
Triolennummer, Änderung ... 44
`tupletDown` ... 43
`tupletNeutral` ... 43
`TupletNumber` ... 44
`tupletNumberFormatFunction` ... 43
`tupletSpannerDuration` ... 43
`tupletUp` ... 43
Türkische Musik ... 425
türkische Notenbezeichnungen ... 425
`tweak` ... 533
`tweak` ... 678
tweak und Kontrollpunkte ... 534
tweak-Befehl in einer Variable ... 534
tweaks-Befehl in Gesangstext ... 534
`two-sided` ... 467
`type` ... 522
typeface ... 643

U

U.C. ... 294
Überbindung ... 48
Überbindung in Wiederholung ... 131
Überbindung und Wiederholungen ... 48
Überbindung, Versetzungszeichen ... 6
Überbindungen und Akkorde ... 48
Überschriften ... 437
Überspringen von Noten im Gesangstext ... 250
Überspringen von Zeichen ... 53
Übungszeichen ... 97
Übungszeichen formatieren ... 98
Übungszeichenstil ... 98
Übungszwecke, Notenköpfe ... 35

Ukulele ... 322
Umbrechen von Seiten ... 501
Umbruch von Text ... 216
Umbrüche in Kadenzen ... 67
Umbrüche in nicht metrischer Musik ... 67
Umbrüche von Zeilen ... 473
Umbrüche, Seite ... 475
Umkehrung ... 13
Umkehrung, modal ... 15
Umkehrungen ... 372, 375
una corda ... 294
`unaCorda` ... 294
`underline` ... 211
underlining text ... 605
`unfold` ... 136
`unfoldRepeats` ... 678
`unHideNotes` ... 193
Unicode ... 447
`unset` ... 530
unsichtbar, Objekte ... 546
unsichtbare Noten ... 193
Unsichtbare Pausen ... 53
unsichtbare Taktstriche ... 88
unsichtbarer Notenhals ... 196
unterschiedlicher Gesangstext ... 254
Unterteilen von Takten ... 81
unterteilte Taktarten ... 68
UTF-8 ... 447

V

Varcoda ... 106, 640
Variable, tweak-Befehl benutzen ... 534
Variablen ... 432
Variablen, Benutzung ... 442
Variablen, Gesangstext ... 239
Vaticana, Editio ... 392
`VaticanaStaff` ... 161, 403
VaticanaStaffContext ... 403
`VaticanaVoice` ... 403
VaticanaVoiceContext ... 403
veränderbare (mutable) Objekte ... 644
Verändern der Schriftart ... 211
Verändern der Schriftgröße ... 472
Verändern der Systemgröße ... 472
Verändern von automatischer Bebalkung ... 77
Verändern von Eigenschaften ... 529
verändern von Objekten ... 547
veränderte Akkorde ... 374
Veränderung des Notensystems ... 540
Veränderung von Kontexten nur einmal ... 532
Veränderung von Verzierungsnoten ... 100
Vermeidung von vertikalen Zusammenstößen ... 496
verschachtelte Kontexte ... 524
Verschachtelte Musik ... 158
verschachtelte Systemklammern ... 166
verschachtelte Wiederholung ... 133
Verschachtelung von Systemen ... 166
Verschieben von Noten ... 149
Verschieben von Pausen, automatisch ... 149
Verschiebung ... 527
Verschiebung von Gesangstext ... 240
Verschmelzen von Noten ... 149
Verschwinden von leeren Systemen ... 175
Versetzungszeichen ... 5
Versetzungszeichen an übergebundener Note ... 6
Versetzungszeichen für Klavier ... 27
Versetzungszeichen in Akkorden ... 30
Versetzungszeichen pro Stimme ... 27
Versetzungszeichen und gleichzeitige Noten ... 30
Versetzungszeichen, automatisch ... 24
Versetzungszeichen, Deutsch ... 5
Versetzungszeichen, Erinnerung ... 6
Versetzungszeichen, für Triller ... 127
Versetzungszeichen, greg. Choral ... 405
Versetzungszeichen, Mensuralnotation ... 400
Versetzungszeichen, moderne Stile ... 26
Versetzungszeichen, moderner Stil mit Warnungen ... 26
Versetzungszeichen, musica ficta ... 401
Versetzungszeichen, piano cautionary ... 28
Versetzungszeichen, Standard ... 24
Versetzungszeichen, Viertelton ... 7
Versetzungszeichen, Vierteltöne ... 5
Versetzungszeichen, Warnung ... 6
Versetzungszeichenstil ... 24
Versetzungszeichenstil forget ... 30
Versetzungszeichenstil Klavier mit Warnungen ... 28
Versetzungszeichenstil modern ... 26
Versetzungszeichenstil neo-modern mit Warnungen ... 28
Versetzungszeichenstil teaching ... 29
Versetzungszeichenstil Vergessen ... 30
Versetzungszeichenstil, modern ... 27
Versetzungszeichenstil, modern mit Warnung für Stimmen ... 27
Versetzungszeichenstil, modern-cautionary ... 26
Versetzungszeichenstil, neo-modern ... 28
Versetzungszeichenstil, neo-modern-voice ... 28
Versetzungszeichenstil, neo-modern-voice-cautionary ... 29
Versetzungszeichenstil, no reset ... 29
Versetzungszeichenstil, piano ... 27
Versetzungszeichenstil, Standard ... 25
Versetzungszeichenstil, Stimme ... 26
Versetzungszeichenstil, Zwölftonmusik ... 29
Versetzungszeichenstil: nicht zurücksetzen ... 29
Verstecken von Noten ... 193
Verstecken von Rhythmus-Systemen ... 176
Verstecken von Systemen ... 175
Verstecken von Systemen der Alten Musik ... 176
versteckte Notensysteme ... 171
`VerticalAxisGroup` ... 482
vertically centering text ... 618
vertikale Ausrichtung von Text ... 214
vertikale Linien zwischen Systemen ... 198
vertikale Position von Dynamik ... 111
vertikale Positionierung ... 481
vertikale Zusammenstöße, vermeiden ... 496
Verwaltung der Zeiteinheiten ... 105
verwendbare Schriftarten auflisten ... 225
Verzierung innerhalb von rhythmischer Kombination ... 46
Verzierung innerhalb von Triole ... 46
Verzierung, danach ... 99
Verzierungen ... 99
Verzierungen verändern ... 100
Verzierungen, Aussehen verändern ... 100
Verzierungen, manuelle Bebalkung ... 84
Verzierungen, Synchronisation ... 103

Verzierungsnoten und Gesangstext ... 257
viele Stimmen ... 149
Vierteltöne ... 5
Vierteltöne in MIDI ... 454
Vierteltonversetzungszeichen ... 7
Violinschlüssel ... 16
`virga` ... 407
`virgula` ... 405
`voice` ... 24, 26
`Voice` ... 146
Voice-Stile ... 149
Voice-Versetzungszeichenstil ... 26
`voiceOne` ... 146
Volta ... 129
Volta und Überbindung ... 48
Volta-Klammer mit Text ... 135
Volta-Klammern und Wiederholungen ... 48
Voltaklammer, ändern ... 134
Vorhalt ... 99
vorhandene Schriftarten auflisten ... 225
Vorlage, arabische Musik ... 424
Vorschlag ... 99
Vorschlag, mehrere Noten ... 103
Vorzeichen ... 20
Vorzeichen in Klammern ... 6
Vorzeichen, Erinnerung ... 6
Vorzeichen, greg. Choral ... 405
Vorzeichen, Mensuralnotation ... 400
Vorzeichen, Vierteltöne ... 5

W

Walker-Formnotenköpfe ... 37
`walkerHeads` ... 37
`walkerHeadsMinor` ... 37
Warnungsversetzungszeichen für Klavier ... 28
Warnungsversetzungszeichen, neo-modern ... 28
Warnungsvorzeichen ... 6
Wechsel der Oktave ... 1
Wechsel des Systems, automatisch ... 290
Wechsel des Systems, manuell ... 289
Wechsel von Instrument ... 181
Wechsel zwischen Systemen ... 292
Wechseln von Instrumentenbezeichnungen ... 180
Weiße Mensuralligaturen ... 402
weit außeinander liegende Balken ... 75
`whichBar` ... 92
Wiederherstellen von Taktart-Standardeigenschaften ... 60
wiederholte Musik ... 136
Wiederholung mit alternativem Schluss ... 129
Wiederholung mit Auftakt ... 130
Wiederholung und Bindebögen ... 48
Wiederholung und Bindebogen ... 133
Wiederholung und Zählzeit ... 133
Wiederholung von Gesangstext bei alternativen Endungen ... 250
Wiederholung, alternative Schlüsse ... 134
Wiederholung, aufklappen ... 136
Wiederholung, Beginn ... 134
Wiederholung, Ende ... 134
Wiederholung, kurz ... 138
Wiederholung, manuell ... 134
Wiederholung, mehrdeutig ... 133
Wiederholung, Prozent ... 138
Wiederholung, taktweise ... 138
Wiederholung, Tremolo ... 140
Wiederholung, verschachtelt ... 133
Wiederholung, Voltaklammer ... 134
Wiederholungen ... 90, 129
Wiederholungen in MIDI ... 454
Wiederholungen mit Überbindung ... 131
Wiederholungen und Gesangstext ... 247
Wiederholungen, ausgeschrieben ... 136
Wiederholungsklammer mit Text ... 135
Wiederholungstaktlinie ... 134
Wiederholungszeichen ... 87
wirkliche Tonhöhe ... 5
`with` ... 519
`with-color` ... 194
`withMusicProperty` ... 678
`wordwrap` ... 216
`wordwrap-lines` ... 222

X

`X-offset` ... 482
`x11-color` ... 194, 195
x11-Farbe ... 195
X11-Farben ... 194
`xNote` ... 678
`xNotesOn` ... 678

Z

Zahl der Notenlinien einstellen ... 169
Zahl eines Taktes ... 92
Zahl von Saiten ... 303
Zählzeit und Wiederholung ... 133
Zeichen ... 106
Zeichen, textartige ... 204
Zeichen, Übung: Formatierung ... 98
Zeichnen im Text ... 217
Zeilenlänge ... 501
Zeilenumbruch, Balken ... 75
Zeilenumbrüche ... 88, 473
Zeilenumbrüche in Intervallen ... 475
Zeilenumbrüche in Kadenzen ... 67
Zeilenumbrüche in nicht metrischer Musik ... 67
Zeit (in der Partitur) ... 105
Zentrieren von Text auf der Seite ... 215
zentrierte Dynamik für Klaviermusik ... 288
Ziernoten ... 99
Zitieren von anderen Stimmen ... 181, 185
zitierter Text ... 201
Zurücksetzen von Taktart-Standardeigenschaften ... 60
Zusammenfalten von Pausen ... 58
Zusammenstöße ... 149
Zusammenstöße zwischen Systemen ... 289
Zusammenstöße, Taktnummern ... 96
Zusammenstöße, vertikal, vermeiden ... 496
zweite Klammer ... 129
Zwischensystem-Tremolo ... 142
Zwischensysteme-Klammer-Arpeggio ... 126
Zwölftonmusik, Versetzungszeichenstil ... 29