Nouvelles fonctionnalités de la version 2.20 (depuis 2.18)

Nouveautés en matière de notation musicale

Améliorations de la représentation des hauteurs

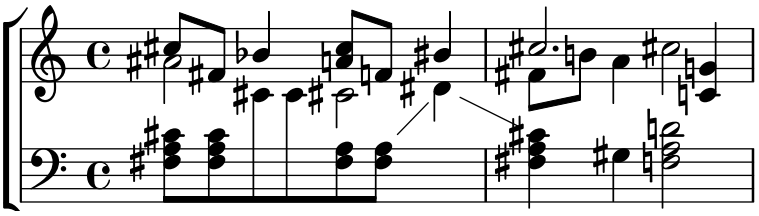
- Dans la langue de notes `english`, le nom développé des notes altérées comprend désormais un trait d'union, pour une meilleure lisibilité. Il faut donc maintenant saisir

  ```
  \key a-flat \major
  ```

  au lieu de

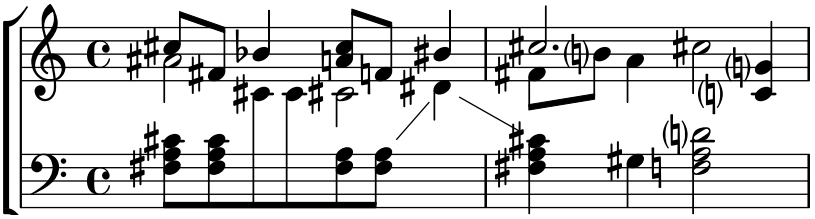
  ```
  \key aflat \major
  ```

  Les altérations doubles ne prennent pas de trait d'union supplémentaire ; le `cisis` batave s'écrit `c-sharpsharp` en anglais développé.
- Les règles en matière d'altération peuvent désormais se définir au niveau d'un contexte `ChoirStaff`.
- Deux nouvelles règles – `choral` et `choral-cautionary` – sont disponibles. Elles combinent les caractéristiques de `modern-voice` et `piano` ou leurs équivalents avec altérations de précaution.
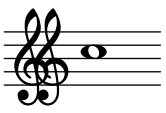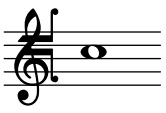


  Ceci constitue désormais le style par défaut pour un `ChoirStaff`.

  `choral-cautionary`

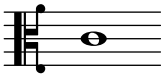


  Identique au style `choral`, à ceci près que les altérations supplémentaires apparaissent comme des altérations de précaution.

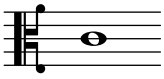
  Voir aussi : Section "Altérations accidentelles automatiques" dans *Manuel de notation*.
- Quatre nouveaux glyphes de clef sont désormais disponibles : *GG* ou *double-G*, *Tenor G*, *varC*, ainsi que leur tessiture respective, et *Varpercussion*.

Exemple	**Résultat**	**Exemple**	**Résultat**
`\clef GG`		`\clef tenorG`	
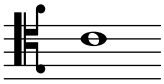
`\clef varC`



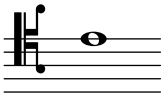
`\clef altovarC`
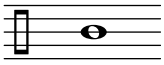


`\clef tenorvarC`



`\clef baritonevarC`
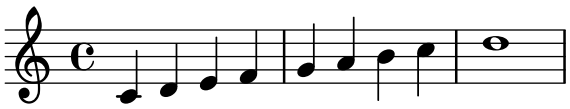


`\clef varpercussion`



Voir aussi : Section “Styles de clef” dans *Manuel de notation*.
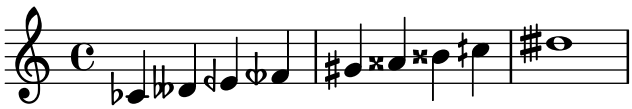
- Les noms de note en français sont maintenant définis spécifiquement, plutôt qu'en alias de l'italien. En plus de la syntaxe dérivée de l'italien, la hauteur *d* peut se saisir *ré* ou `re`, et un double-dièse par le suffice *-x*.

```
\language "français"
do ré mi fa | sol la si do | ré1
```



Les doubles dièses s'indiquent par l'adjonction d'un `x`.

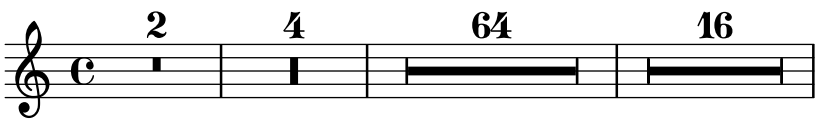
```
\language "français"
dob, rebb misb fabsb | sold ladd six dosd | rédsd1
```



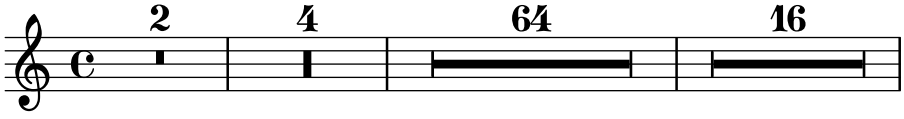
Améliorations en matière de rythme

- Les silences multimesures ont une longueur dépendant de leur durée totale, sous contrôle de `MultiMeasureRest.space-increment`. La valeur par défaut est fixée à `2.0`.

```
\compressEmptyMeasures
R1*2 R1*4 R1*64 R1*16
```
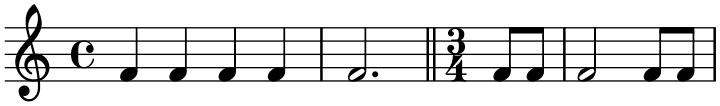



```
\compressEmptyMeasures
\override Staff.MultiMeasureRest.space-increment = 2.5
R1*2 R1*4 R1*64 R1*16
```



- La commande `\partial` a connu des améliorations notables afin d'éviter les problèmes en cas de contextes multiples ou parallèles.

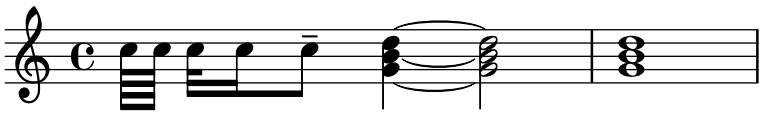
- Il est désormais possible d'utiliser `\time` et `\partial` de concert pour modifier une métrique en cours de mesure.

```
f f f f | f2. \bar "||"
\time 3/4 \partial 4
f8 8 | f2 f8 f |
```

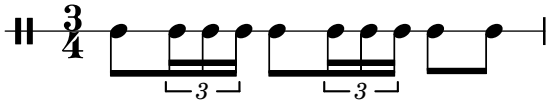


- Des durées isolées dans une séquence musicale sont désormais considérées comme des notes sans hauteur. Dans la partition finale, les hauteurs sont déterminées à partir de la note ou de l'accord qui précède. Ceci peut s'avérer utile pour affecter des rythmes à de la musique ou dans une fonction Scheme.

```
c64[ 64] 32 16 8^- <g b d>4~ 2 | 1
```



```
\new DrumStaff \with { \override StaffSymbol.line-count = 1 }
\drummode {
  \time 3/4
  tambourine 8 \tuplet 3/2 { 16 16 16 }
             8 \tuplet 3/2 { 16 16 16 } 8 8 |
}
```



- Les exceptions en matière de ligature peuvent désormais se libeller à l'aide de la fonction Scheme `\beamExceptions`. Ce qui auparavant se définissait par
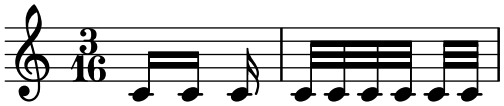
```
\set Timing.beamExceptions =
#'(                          ;début de la liste associative
   (end .                    ;entrée pour la terminaison des ligatures
    (                        ;début de la liste des terminaisons
     ((1 . 32) . (2 2 2))    ;règle pour les triples croches -- groupées à la double
    )))

\time #'(2 1) 3/16
c16 c c
\repeat unfold 6 { c32 }
```

se libelle, grâce à la fonction Scheme `\beamExceptions`

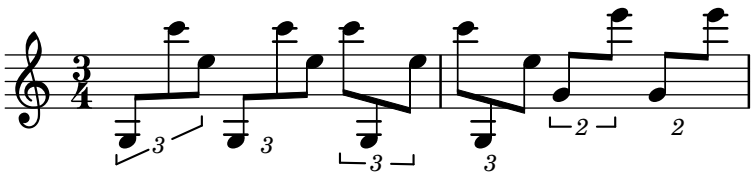
```
\set Timing.beamExceptions =
  \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }

\time #'(2 1) 3/16
c16 c c |
\repeat unfold 6 { c32 } |
```

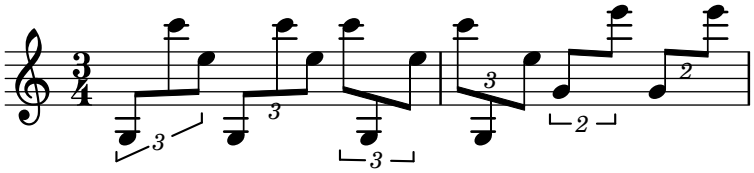


tout en séparant les exceptions par un `|` (contrôle de barre de mesure) – l'absence de hauteur dans les motifs d'exception n'est pas obligatoire.

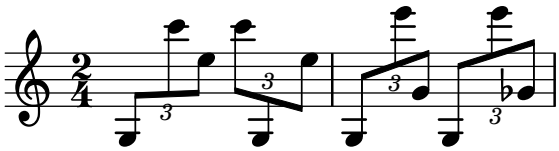
- Amélioration significative du positionnement du nombre des n-olets dans le cas de ligatures coudées. Jusqu'à présent, ce nombre était placé selon la position du crochet même lorsque ce dernier n'était pas imprimé, ce qui pouvait amener à un positionnement disgracieux :



Le nombre est désormais positionné plus près du coude en présence d'un tronçon de ligature approprié à son placement et en l'absence de crochet.
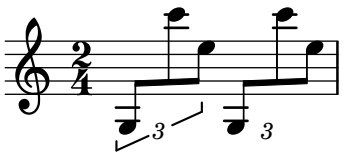


- De plus, la détection de collision ajoutée décalera horizontalement le nombre s'il était trop proche d'un empilement adjacent, tout en préservant son écartement de la ligature. Dans le cas où ce nombre serait trop large pour tenir dans l'espace disponible, LilyPond reviendra au positionnement basé sur le crochet. Dans le cas d'une collision avec, par exemple, une altération accidentelle, le nombre sera plutôt écarté verticalement.



L'ancien comportement des n-olets en présence de ligature coudée reste disponible au moyen d'une nouvelle propriété `knee-to-beam` affectée àl'objet de rendu `TupletNumber`.

```
\time 2/4
\override Beam.auto-knee-gap = 3
\override TupletNumber.knee-to-beam = ##f
\override TupletBracket.bracket-visibility = ##t
\tuplet 3/2 4 { g8 c'' e, }
\once \override TupletBracket.bracket-visibility = ##f
\tuplet 3/2 4 { g,,8 c'' e, }
```



Améliorations en matière d'expressivité
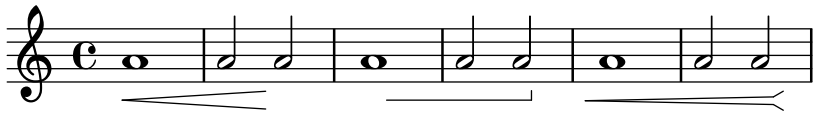
- Les terminaisons des soufflets peut désormais s'affiner à l'aide de la propriété de `grob` `shorten-pair`. Celle-ci n'affectait auparavant que les extensions de type textuel tels que `TupletBracket` et `OttavaBracket`.

Les valeurs positives décalent vers la droite, les négatives vers la gauche.

```
\once \override Hairpin.shorten-pair = #'(0 . 2)
a1\< | a2 a\!

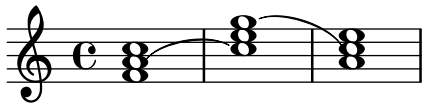
\once \override Hairpin.shorten-pair = #'(2 . 0)
\once \override Hairpin.stencil = #constante-hairpin
a1\< | a2 a\!

\once \override Hairpin.shorten-pair = #'(-1 . -1)
\once \override Hairpin.stencil = #flared-hairpin
a1\< | a2 a\!
```

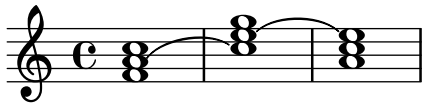


- Les liaisons, qu'elles soient d'articulation ou de phrasé, peuvent désormais débuter sur une hauteur particulière d'un accord.

```
<f a( c>1 | <c') e g(> | <a c) e>
```



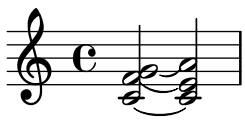
```
<f( a\( c>1 | <c'\) e\( g> | <a c e\)>
```



- Le nouvelle commande `\=` permet de spécifier le `spanner-id` (identificateur d'extension) pour des liaisons d'articulation ou de phrasé simultanées ; l'identificateur est soit un entier positif, soit un symbole.

  Ceci s'avère utile en présence de multiples liaisons ou lorsqu'une liaison en recouvre une autre ou bien encore lorsque plusieurs courtes laisons sont englobées dans une plus grande.
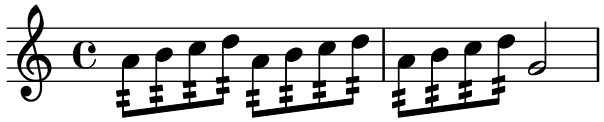
```
\fixed c' {
  <c~ f\=1( g\=2( >2 <c e\=1) a\=2) >
}
```



Voir aussi : Section "Signes d'interprétation sous forme de courbe" dans *Manuel de notation*.

Améliorations en matière de reprises

- Le style visuel des traits de tremolo (allure, style et pente) est maintenant plus finement contrôlé.

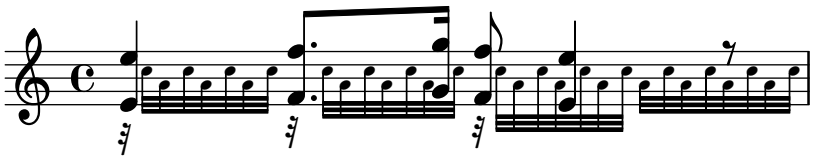


- La fonction musicale `\unfoldRepeats` peut maintenant prendre en argument optionnel une liste spécifiant le ou les types de musique à répéter dans le développement. Sont disponibles les arguments `percent`, `tremolo` et `volta`. En l'absence de liste optionnelle d'arguments, sera utilisé `repeated-music` et tout sera développé.

Améliorations en matière de notation sur la portée

- Ajout d'une nouvelle commande `\magnifyStaff`, qui échelonne de façon globale, au niveau d'un contexte `Staff`, sa taille, les lignes de portée, les barres de mesure, les hampes et l'espacement horizontal. Les lignes de la portée considérée ne seront toutefois pas plus fines que la taille par défaut dans la mesure où l'épaisseur des hampes, liaisons et autres est basée sur l'épaisseur des lignes de portée.
- La nouvelle commande `\magnifyMusic` permet de modifier la taille de la musique sans changer la taille de la portée, tout en ajustant automatiquement les hampes, ligatures et l'espacement horizontal.

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```



- La nouvelle commande `\RemoveAllEmptyStaves` agit exactement comme `\RemoveEmptyStaves`, à ceci près qu'elle supprime aussi les lignes du premier système de la partition.
- Ajout de la commande de *markup* `\justify-line`. Cette fonction est comparable à `\fill-line`, à ceci près qu'au lieu de répartir les *mots* en colonnes, la commande `\justify-line` répartit les vides de telle sorte qu'en présence de trois *mots* ou plus, ces blancs soient d'égale longueur.

```
\markup \fill-line {oooooo oooooo oooooo oooooo}
\markup \fill-line {ooooooooo ooooooooo oo ooo}
```

oooooo oooooo oooooo oooooo

ooooooooo oooooooo oo ooo

```
\markup \justify-line {oooooo oooooo oooooo oooooo}
\markup \justify-line {ooooooooo ooooooooo oo ooo}
```

oooooo oooooo oooooo oooooo

000000000 00000000 00 000

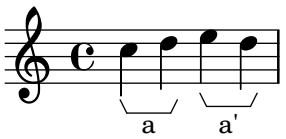
- L'utilisation de fontes alternatives à Emmentaler, la fonte par défaut de LilyPond, a été simplifiée. Voir Section "Changement des fontes musicales" dans *Manuel de notation* pour de plus amples informations.

Améliorations en matière d'annotations éditoriales

- Il est désormais possible d'ajouter du texte à un crochet d'analyse, grâce à l'objet `HorizontalBracketText`.

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

{
  \once \override HorizontalBracketText.text = "a"
  c''\startGroup d''\stopGroup
  e''-\tweak HorizontalBracketText.text "a'" \startGroup d''\stopGroup
}
```



Améliorations en matière de mise en forme du texte

- La prise en charge de fontes musicales alternative, autrement dit autres que Emmentaler, est facilitée. Voir Section "Changement des fontes musicales" dans *Manuel de notation* pour de plus amples informations.
- Les fontes textuelles par défaut – `Century Schoolbook L`, `sans-serif` et `monospace` – ont été modifiées.

  Pour le moteur `svg` :

Famille	**Fonte par défaut**
roman	`serif`
sans	`sans-serif`
typewriter	`monospace`

  `serif`, `sans-serif` et `monospace` sont des `generic-family` au titre des spécifications SVG et CSS.

  Pour les autres moteurs :

Famille	**Fonte par défaut (alias)**	**Listes de définition des alias**
roman	`LilyPond Serif`	TeX Gyre Schola, C059, Century SchoolBook URW, Century Schoolbook L, DejaVu Serif, ..., serif
sans	`LilyPond Sans Serif`	TeX Gyre Heros, Nimbus Sans, Nimbus Sans L, DejaVu Sans, ..., sans-serif
typewriter	`LilyPond Monospace`	TeX Gyre Cursor, Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, DejaVu Sans Mono, ..., monospace

`LilyPond Serif`, `LilyPond Sans Serif` et `LilyPond Monospace` sont des alias de fonte définis dans le fichier de configuration de FontConfig spécifique à LilyPond `00-lilypond-fonts.conf`. Lorsqu'un caractère est absent de la première fonte listée, il sera remplacé par celui de la fonte suivante. Pour plus de détails sur les définitions des alias, consultez le fichier `00-lilypond-fonts.conf` dans le répertoire d'installation.

- Le recours aux fontes OpenType permet d'utiliser certaines fonctionnalités de ces fontes. Toutefois, les fontes OpenType ne disposent pas toutes de l'intégralité de ces fonctions.

```
% Vraies petites capitales
\markup { Style normal : Hello HELLO }
\markup { \caps { Petites capitales : Hello } }
\markup { \override #'(font-features . ("smcp"))
          { Vraies petites capitales : Hello } }

% Styles numériques
\markup { Style numérique normal : 0123456789 }
\markup { \override #'(font-features . ("onum"))
          { Style numérique ancien : 0123456789 } }

% Alternatives stylistiques
\markup { \override #'(font-features . ("salt 0"))
          { Alternative stylistique 0 : εφπρθ } }
\markup { \override #'(font-features . ("salt 1"))
          { Alternative stylistique 1 : εφπρθ } }

% Fonctionnalités multiples
\markup { \override #'(font-features . ("onum" "smcp" "salt 1"))
          { Fonctionnalités multiples : Hello 0123456789 εφπρθ } }
```

Style normal : Hello HELLO

Petites capitales : Hello

Vraies petites capitales : Hello

Style numérique normal : 0123456789

Style numérique ancien : 0123456789

Alternative stylistique 0 : εφπρθ

Alternative stylistique 1 : ϵϕϖϱϑ

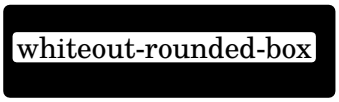
Fonctionnalités multiples : Hello 0123456789 ϵϕϖϱϑ

- Deux nouveaux styles de contours à blanc sont disponibles. Le style `outline` agit un peu comme un ombrage des glyphes, son galbe étant le résultat de multiples copies du glyphe. Le style `rounded-box` produit un rectangle aux coins arrondis. Pour tous les styles, y compris le style par défaut `box`, l'épaisseur (`thickness`) du contour, mesuré en épaisseur de ligne de portée, est adaptable.

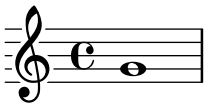
```
\markup {
```

```
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
    \override #'(thickness . 3)
    \whiteout whiteout-box
}
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'(style . rounded-box)
    \override #'(thickness . 3)
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'(style . outline)
    \override #'(thickness . 3)
    \whiteout whiteout-outline
}
\relative {
  \override Staff.Clef.whiteout-style = #'outline
  \override Staff.Clef.whiteout = 3
  g'1
}
```









- Une nouvelle commande de *markup*, `\with-dimensions-from`, rend plus aisée l'utilisation de `\with-dimensions` en adoptant les dimensions d'un objet *markup* fourni en premier argument.

```
\markup {
  \pattern #5 #Y #0 "x"
  \pattern #5 #Y #0 \with-dimensions-from "x" "f"
  \pattern #5 #Y #0 \with-dimensions-from "x" "g"
  \override #'(baseline-skip . 2)
  \column {
    \pattern #5 #X #0 "n"
    \pattern #5 #X #0 \with-dimensions-from "n" "m"
```

```
    \pattern #5 #X #0 \with-dimensions-from "n" "!"
  }
}
```

- La nouvelle commande de *markup* `\draw-squiggle-line` permet de tracer des lignes ondulées. Sont adaptables l'épaisseur du trait (`thickness`), l'amplitude (`angularity`), la hauteur (`height`) et l'orientation (`orientation`).

```
\markup
  \overlay {
    \draw-squiggle-line #0.5 #'(3 . 3) ##t

    \translate #'(3 . 3)
    \override #'(thickness . 4)
    \draw-squiggle-line #0.5 #'(3 . -3) ##t

    \translate #'(6 . 0)
    \override #'(angularity . -5)
    \draw-squiggle-line #0.5 #'(-3 . -3) ##t

    \translate #'(3 . -3)
    \override #'(angularity . 2)
    \override #'(height . 0.3)
    \override #'(orientation . -1)
    \draw-squiggle-line #0.2 #'(-3 . 3) ##t
  }
```

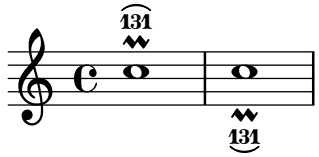


- Deux commandes de *markup* font leur apparition : `\undertie` et `\overtie`, ainsi qu'une version générique `\tie`.

```
\markup {
  \undertie "undertied"
  \overtie "overtied"
}

m = {
  c''1 \prall -\tweak text \markup \tie "131" -1
}

{ \voiceOne \m \voiceTwo \m }
```



Nouveautés en matière de notation spécialisée

Améliorations pour la musique vocale

- Création d'un gabarit flexible dans le domaine de la musique chorale. Il s'utilise pour de la musique chorale simple, avec ou sans accompagnement de piano, sur deux ou quatre portées. Contrairement aux autres gabarits, celui-ci est directement intégré à LilyPond ; il n'est donc pas besoin d'être recopié et édité, mais simplement appelé à l'aide d'un `\include` dans le fichier source. Pour de plus amples détails, voir Section "Gabarits préprogrammés" dans *Manuel d'initiation*.
- `\lyricsto` et `\addLyrics` ont été « harmonisés ». Tous deux acceptent désormais la même sorte de liste délimitée d'arguments, à l'instar de `\lyrics` et `\chords`. Une rétrocompatibilité a été ajoutée, de sorte à accepter des identificateurs musicaux (tel `\mus`) en tant qu'arguments. Une règle a été ajoutée à `convert-ly` visant à supprimer les utilisations redondantes de `\lyricmode` et réarranger les combinaisons avec les déclencheurs de contexte afin d'appliquer `\lyricsto` généralement en dernier (comme le ferait `\lyricmode`).

Améliorations pour les cordes frettées ou non

- Ajout d'un nouveau style de tête de note pour les tablature : `TabNoteHead.style = #'slash`.
- L'écartement entre les diagrammes de fret et entre les cordes peut désormais s'ajuster de manière indépendante, à l'aide des sous-propriétés de `fret-diagram-details` `fret-distance` et `string-distance`.

```
fretMrkp = \markup { \fret-diagram-terse "x;x;o;2;3;2;" }

\markuplist
\override #'(padding . 2)
\table #'(0 -1) {
  "default"

  \fretMrkp

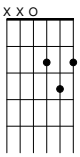
  "fret-distance"

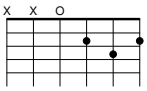
  \override #'(fret-diagram-details . ((fret-distance . 2)))
  \fretMrkp

  "string-distance"

  \override #'(fret-diagram-details . ((string-distance . 2)))
  \fretMrkp
}
```

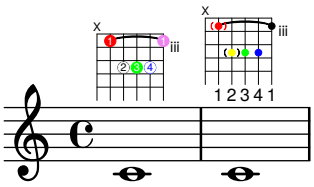
default 

fret-distance 

string-distance 

- Dans le cadre de l'utilisation de la commande de *markup* `\fret-diagram-verbose`, il est désormais possible de coloriser individuellement les points et les parenthèses des diagrammes de fret.

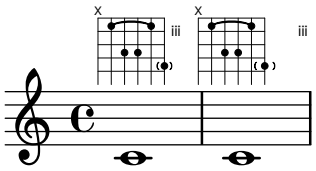
```
\new Voice {
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (finger-code . in-dot))) {
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1 red)
                               (place-fret 4 5 2 inverted)
                               (place-fret 3 5 3 green)
                               (place-fret 2 5 4 blue inverted)
                               (place-fret 1 3 1 violet)
                               (barre 5 1 3 ))
    }
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (finger-code . below-string))) {
      \fret-diagram-verbose #'((mute 6)
                              (place-fret 5 3 1 red parenthesized)
                              (place-fret 4 5 2 yellow
                                               default-paren-color
                                               parenthesized)
                              (place-fret 3 5 3 green)
                              (place-fret 2 5 4 blue )
                              (place-fret 1 3 1)
                              (barre 5 1 3))
    }
  }
}
```



- Deux propriétés sont ajoutées à `fret-diagram-details` pour une utilisation avec la commande de *markup* `\fret-diagram-verbose` : `fret-label-horizontal-offset` affecte le `fret-label-indication`, et `paren-padding` contrôle l'espacement entre le point et les parenthèses qui l'entourent.

```
\new Voice {
  c1^\markup {
    \fret-diagram-verbose #'((mute 6)
                             (place-fret 5 3 1)
                             (place-fret 4 5 2)
                             (place-fret 3 5 3)
                             (place-fret 1 6 4 parenthesized)
                             (place-fret 2 3 1)
```

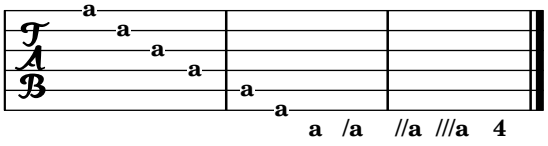
```
                                (barre 5 2 3))
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (fret-label-horizontal-offset . 2)
                 (paren-padding . 0.25))) {
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1)
                               (place-fret 4 5 2)
                               (place-fret 3 5 3)
                               (place-fret 1 6 4 parenthesized)
                               (place-fret 2 3 1)
                               (barre 5 2 3))
    }
  }
}
```



- Dans le cadre des tablatures pour luth sont désormais disponibles les cordes de basse additionnelles.

```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }
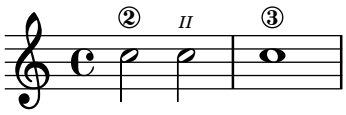
\score {
  \new TabStaff \m
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



- Les numéros de cordes peuvent désormais s'imprimer en chiffres romains, pour les instruments à cordes non frettées par exemple.

```
c2\2
\romanStringNumbers
c\2
```

```
\arabicStringNumbers
c1\3
```

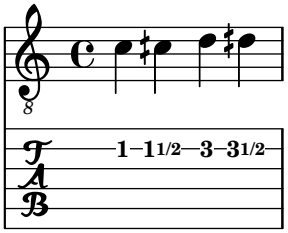


- Les microaltérations peuvent désormais s'imprimer aussi sur des `TabStaff`, notamment pour indiquer des *bends*.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

mus = \relative { c'4 cih d dih }

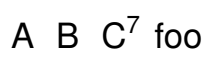
<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \mus
>>
```



Améliorations pour la notation des accords

- `\chordmode` prend désormais en charge les constructions `< >` et `<< >>`.
- Il est désormais possible de modifier la propriété `text` des noms d'accord.

```
<<
\new ChordNames \chordmode {
  a' b c:7
  \once \override ChordName.text = "foo"
  d
}
>>
```



Nouveautés en matière d'entrée et sortie

Améliorations pour la structure

- Les blocs introduits par `\header` peuvent être stockés dans des variables et utilisés en argument à la musique ou à des fonctions Scheme ainsi que dans le corps de constructions `#{...#}`. Ils sont représentés en tant que module Guile.

  Les blocs `\book`, `\bookpart`, `\score`, `\with`, `\layout`, `\midi` et `\paper` peuvent être passés de façon similaire, mais sont représentés par des types de donnée différents.

Améliorations pour les titrages et entêtes

- Les numéros de page peuvent désormais s'imprimer en chiffres romains, en ajustant la variable de papier `page-number-type`.

Améliorations pour les fichiers sources

- La nouvelle commande `\tagGroup` vient en complément des commandes `\keepWithTag` et `\removeWithTag` déjà existantes. Par exemple,

  ```
  \tagGroup #'(violinI violinII viola cello)
  ```

  déclare une liste de balises appartenant à un unique « groupe de balises ».

  ```
  \keepWithTag #'violinI
  ```

  n'est maintenant plus concerné que par les balises du groupe auquel « violinI » appartient.

  Tout élément balisé par une ou plusieurs balises du groupe, à l'exception de *violinI*, sera ignoré.

Améliorations pour les fichiers résultants

- Les fichiers sources LilyPond peuvent désormais être empaquetés dans les fichiers PDF générés. Cette fonctionnalité est pour l'instant désactivée par défaut car susceptible d'être considérée comme peu sure dans la mesure où des documents PDF comportant des fichiers cachés peuvent présenter des risques en matière de sécurité. Attention cependant : les lecteurs de PDF ne sont pas tous capables de gérer les fichiers joints ; si tel est le cas, le rendu PDF apparaîtra normalement mais les fichiers joints seront invisibles. Cette fonctionnalité n'est opérationnelle qu'avec le moteur PDF.
- La procédure `output-classic-framework` et l'option `-dclip-systems` sont désormais prises en charge par le moteur `SVG`.
- Le nouvel argument `-dcrop` permet de formater une sortie `SVG` ou `PDF` sans marges ni saut de page.
- La propriété d'objet graphique `output-attributes` remplace, dans les sorties SVG, la propriété de *grob* `id`. Ceci permet de définir, à l'aide d'une liste associative, de multiples attributs. Par exemple, `#'((id . 123) (class . foo) (data-whatever . « bar »))` produira dans un fichier SVG le groupe de balise : `<g id=« 123 » class=« foo » data-whatever=« bar »> ... </g>`.
- La fonctionnalité PostScript d'ajustement des traits ne s'applique plus de manière automatique ; elle est désormais laissée à l'appréciation du périphérique PostScript – Ghostscript l'utilise par défaut pour des résolutions inférieures à 150 dpi lorsqu'il génère des images *raster*. Lorsqu'elle est activée, un algorithme de dessin plus complexe tirant profit des ajustements de trait servira notamment pour les ligatures et barres de mesure.

  L'ajustement des traits peut se forcer, en ligne de commande, à l'aide de l'option '`-dstrokeadjust`'. En ce qui concerne la génération de fichiers `PDF`, ceci améliorera nettement la prévisualisation, au détriment cependant de la taille du fichier. La qualité d'impression à haute résolution n'est pas affectée.
- Une nouvelle fonction – `make-path-stencil` – supporte toutes les commandes `path`, tant relatives qu'absolues :

  `lineto`, `rlineto`, `curveto`, `rcurveto`, `moveto`, `rmoveto`, `closepath`. La fonction prend aussi en charge la syntaxe « lettre unique » utilisée dans les commandes de chemin du standard SVG :

  `L`, `l`, `C`, `c`, `M`, `m`, `Z` et `z`. Cette nouvelle commande est rétrocompatible avec la fonction originale `make-connected-path-stencil`. Voir aussi le fichier `scm/stencil.scm`.

Améliorations pour le MIDI

- La plupart des articulations communes sont réflétées dans le rendu MIDI. Accent et marcato donnent des notes plus fortes ; staccato, staccatissimo et portato abrègent les notes. Une marque de respiration raccourcit la note qui précède.

  Ce comportement est ajustable au travers des propriétés `midiLength` et `midiExtraVelocity` affectées à `ArticulationEvent`. Voir le fichier `script-init.ly` pour des exemples.

- Amélioration du rendu MIDI des respirations. Après une note liée, la respiration prend son temps uniquement sur la dernière note de la prolongation. Autrement dit, `{ c4~ c8 \breathe }` s'entendra comme `{ c4~ c16 r }` au lieu de `{ c4 r8 }`. Ceci est plus cohérent en matière d'articulation et avec la manière dont un instrumentiste interprète une respiration après une note prolongée par une liaison. Ceci permet aussi d'aligner plus facilement une respiration simultanée à plusieurs parties dont les notes différent dans leur durée.

- Il est désormais possible de contrôler le « niveau d'expression » des canaux MIDI à l'aide de la propriété de contexte `Staff.midiExpression`. Ceci permet d'altérer le volume perçu y compris des notes tenues, bien que légèrement. L'ajustement prend une valeur entre `0.0` et `1.0`.

```
\score {
  \new Staff \with {
    midiExpression = #0.6
    midiInstrument = "clarinet"
  }
  <<
    { a'1~ a'1 }
    {
      \set Staff.midiExpression = #0.7 s4\f\<
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.9 s4
      \set Staff.midiExpression = #1.0 s4

      \set Staff.midiExpression = #0.9 s4\>
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.7 s4
      \set Staff.midiExpression = #0.6 s4\!
    }
  >>
  \midi { }
}
```

- Lorsqu'il génère un fichier MIDI, LilyPond enregistre désormais le `title` défini dans le bloc `\header` d'une partition en tant que nom de la séquence MIDI du fichier MIDI. En l'absence de `title` au niveau `\score`, sera retenue la première définition trouvée dans l'ordre suivant : `\bookpart`, `\book` et enfin `\header` de premier niveau. De manière optionnelle, le nom de la séquence MIDI peut se définir à l'aide du nouveau champ de `\header` `midititle` indépendamment au cas où le champ `title` contiendrait du code *markup* qui ne serait pas rendu correctement en texte plat.

Améliorations en matière d'extraction de la musique

- `\displayLilyMusic` et ses fonctions Scheme sous-jacentes n'omettent plus les durées de note redondantes. Il est désormais plus facile et sûr de reconnaître et formater les durées isolées dans des expressions telles que

```
{ c4 d4 8 }
```

Nouveautés en matière d'espacements

Améliorations pour les sauts de page

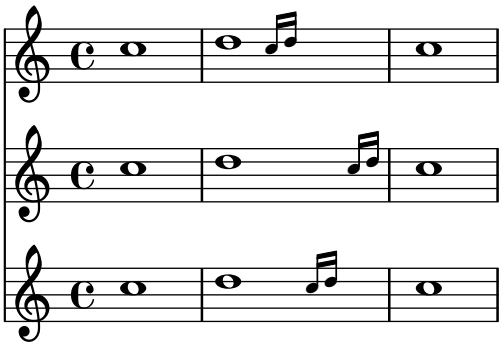
- Il est désormais possible de déplacer des systèmes relativement à leur positionnement actuel, à l'aide de la sous-propriété `extra-offset` de `NonMusicalPaperColumn.line-break-system-details`, que ce soit sur l'axe horizontal ou vertical. Cette fonctionnalité s'avère très utile pour effectuer un léger ajustement dans le positionnement vertical par défaut d'un système particulier. Voir Section "Positionnement explicite des portées et systèmes" dans *Manuel de notation* pour de plus amples explications.
- Amélioration de l'espacement visuel du « MI », tant à taille réduite que normale, dans les styles Funk et Walker, de telle sorte qu'il ait la même taille que les autres têtes de note profilées. Les « SOL » sont aussi améliorés dans les styles Aiken et Harpe sacrée normaux ou leurs variantes fines.
- `LeftEdge` dispose désormais d'un `Y-extent` (extension verticale) définissable. Voir Section "LeftEdge" dans *Référence des propriétés internes*.
- Les objets graphiques et leurs parents peuvent désormais s'aligner de manière indépendante, ce qui permet une flexibilité accrue dans le positionnement des *grobs*. Par exemple, le bord « gauche » d'un objet peut désormais s'aligner sur le « centre » de son parent.

Nouveautés en matière de modification des réglages par défaut

Un argument supplémentaire a été ajouté à la commande `\afterGrace`.

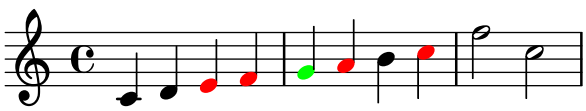
`\afterGrace` dispose à présent d'un argument permettant de spécifier, sous forme de fraction, l'espacement à respecter avant de positionner ses notes.

```
<<
\new Staff \relative {
  % Par défaut, valeur figée à (3/4)
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Modification manuelle à (15/16) de la valeur figée
  #(define afterGraceFraction (cons 15 16))
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Utilisation du nouvel argument (5/6)
  c''1 \afterGrace 5/6 d1 { c16[ d] } c1
}
>>
```



- Les différentes instructions `\override`, `\revert`, `\set` et `\unset` acceptent désormais le préfixe `\once` qui en réduit les effets à un seule occurrence.

```
\relative {
  c'4 d
  \override NoteHead.color = #red
  e4 f |
  \once \override NoteHead.color = #green
  g4 a
  \once \revert NoteHead.color
  b c |
  \revert NoteHead.color
  f2 c |
}
```
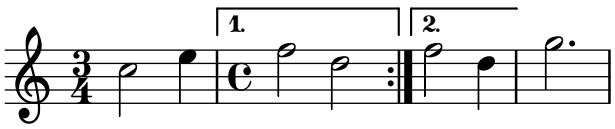


Nouveautés en matière d'interfaces et fonctions internes

- La propriété musicale et d'objet graphique `spanner-id`, qui permet de distinguer des liaisons d'articulation ou de phrasé simultanées, prend en argument une « clé » – entrier positif ou symbole – au lieu d'une chaîne.
- Les propriétés de contexte nommées dans la propriété '`alternativeRestores`' sont restaurées à la valeur qu'elles avaient juste avant la **première** alternative, ce pour toutes les suivantes.

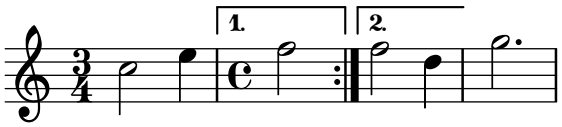
  Pour l'instant, le jeu par défaut restaure la « métrique courante » :

```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4 f2 d | }
  { f2 d4 | }
}
g2. |
```



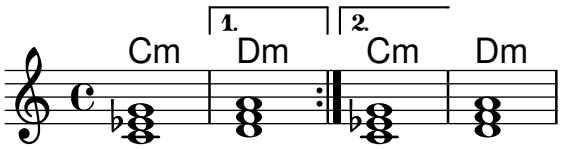
  la « position dans la mesure » :

```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4
    \set Timing.measurePosition = #(ly:make-moment -1/2)
    f2 | }
  { f2 d4 | }
}
g2. |
```



et les « changements d'accord »:

```
<<
  \new ChordNames {
    \set chordChanges = ##t
    \chordmode { c1:m d:m c:m d:m }
  }
  \new Staff {
    \repeat volta 2 { \chordmode { c1:m } }
    \alternative {
      { \chordmode { d:m } }
      { \chordmode { c:m } }
    }
  \chordmode { d:m }
}
>>
```



- Les fonctions LilyPond définies à l'aide de `define-music-function`, `define-event-function`, `define-scheme-function` et `define-void-function` peuvent désormais être appelées directement à partir de Scheme, comme s'il s'agissait de pures procédures Scheme. Le contrôle et la correspondance des arguments seront réalisés de manière identique à ce qui se passe lorsque la fonction est appelée au fil du code LilyPond. Ceci inclut l'insertion de valeurs par défaut pour des arguments optionnels qui ne correspondraient pas à leur prédicat. Dans la liste des arguments, il est possible d'utiliser `*unspecified*` au lieu de `\default` pour omettre explicitement une séquence d'argument optionnels.
- Les données `location` pour la saisie courante et `parser` sont désormais gérées directement dans les flux GUILE ; elles peuvent donc être référencées par des appels de fonction`(*location*)` et `(*parser*)`. Par voie de conséquence, nombre de fonctions ont vu disparaître leur argument `parser` explicite.

  Les fonctions définies par `define-music-function`, `define-event-function`, `define-scheme-function` et `define-void-function` n'ont désormais nul besoin d'argument `parser` ou `location`.

  Avec ces définitions particulières, LilyPond tentera de reconnaître l'utilisation héritée des arguments `parser` et `location`, fournissant ainsi une compatibilité ascendante de la sémantique pour un certain temps.
- Les définitions et identificateurs Scheme peuvent désormais s'utiliser en tant que définition de sortie.
- Les expressions Scheme peuvent désormais s'utiliser en tant que constituants d'un accord.
- Les fonctions, qu'elles soient musicales, Scheme ou fantômes, ainsi que les commandes de *markup* pour lesquelles le paramètre final est l'objet de dérogations en chaîne peuvent se définir en remplaçant l'expression à laquelle elle s'appliquera par `\etc`. Il en va de même pour les appels à une fonction ou une commande de *markup*.

```
bold-red-markup = \markup \bold \with-color #red \etc
highlight = \tweak font-size 3 \tweak color #red \etc
```
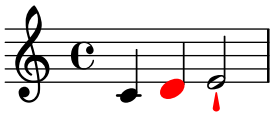
```
\markup \bold-red "text"
\markuplist \column-lines \bold-red { One Two }

{ c' \highlight d' e'2-\highlight -! }
```

text
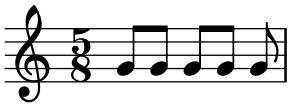
One

Two



- Les listes de symboles séparés par des points, à l'instar de `FretBoard.stencil` sont pris en charge depuis la version 2.18. Elles peuvent désormais contenir des entiers non signés, et leurs membres être séparés par des virgules. Ceci permet des libellés tels que

```
{ \time 2,2,1 5/8 g'8 8 8 8 8 }
```
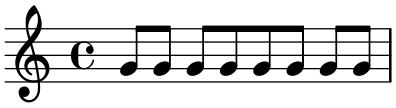


ou

```
\tagGroup violin,oboe,bassoon
```

- De telles listes peuvent aussi apparaître au sein d'expressions aux fins d'assignation, de définition ou de dérogation. Ceci permet des libellés tels que

```
{ \unset Timing.beamExceptions
  \set Timing.beatStructure = 1,2,1
  g'8 8 8 8 8 8 8 8 }
```



- Les éléments d'une liste associative pouvaient déjà se voir attribuer des valeurs individuellement, comme par exemple `system-system-spacing.basic-distance` pour les variables concernant le papier. Ils peuvent désormais être référencés de la même manière, comme ici

```
\paper {
  \void \displayScheme \system-system-spacing.basic-distance
}
```

Par extension à ces modifications, il est dorénavant possible de définir et faire référence à des pseudovariables telles que `violon.1`.

- La commande `\table`, pour gérer des listes de *markups* est disponible. Chaque colonne peut disposer de son propre alignement.

```
\markuplist {
    \override #'(padding . 2)
```

```
    \table
      #'(0 1 0 -1)
      {
        \underline { center-aligned right-aligned center-aligned left-aligned }
        one "1" thousandth "0.001"
        eleven "11" hundredth "0.01"
        twenty "20" tenth "0.1"
        thousand "1000" one "1.0"
      }
}
```

center-aligned	right-aligned	center-aligned	left-aligned
one	1	thousandth	0.001
eleven	11	hundredth	0.01
twenty	20	tenth	0.1
thousand	1000	one	1.0

- `InstrumentName` prend désormais en charge la `text-interface`.
- La propriété `thin-kern` du *grob* `BarLine` est renommée en `segno-kern`.
- Les objets `KeyCancellation` ignorent désormais les clefs de citation, à l'instar des objets `KeySignature`.
- Prise en charge de `\once \unset`