LilyPond

Le système de gravure musicale

Manuel d'initiation

L'équipe de développement de LilyPond

Ce document constitue le manuel d'initiation à GNU LilyPond 2.14.2.

Pour connaître la place qu'occupe ce manuel dans la documentation, consultez la page Section "Manuels" dans *Informations générales*.

Si vous ne disposez pas de certains manuels, la documentation complète se trouve sur http://www.lilypond.org/.

Copyright © 1999–2011 par les auteurs.

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

La traduction de la notice de droits d'auteur ci-dessous vise à faciliter sa compréhension par le lecteur non anglophone, mais seule la notice en anglais a valeur légale.

> Vous avez le droit de copier, distribuer et/ou modifier ce document selon les termes de la Licence GNU de documentation libre, version 1.1 ou tout autre version ultérieure publiée par la Free Software Foundation, "sans aucune section invariante". Une copie de la licence est fournie à la section "Licence GNU de documentation libre".
>
> 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".

Pour LilyPond version 2.14.2

Table des matières

1 Tutoriel ... 1

1.1 Compilation d'un fichier ... 1
1.1.1 Saisie de la musique ... 1
1.1.2 MacOS X ... 2
1.1.3 Windows ... 6
1.1.4 Ligne de commande ... 13
1.2 Composition d'un fichier source ... 13
1.2.1 Notation simple ... 13
1.2.2 Travail sur les fichiers d'entrée ... 18
1.3 Gestion des erreurs ... 19
1.3.1 Quand ça ne fonctionne pas ... 19
1.3.2 Erreurs courantes ... 19
1.4 Bien lire le manuel ... 19
1.4.1 Matériel incomplet ... 19
1.4.2 Exemples cliquables ... 20
1.4.3 Vue d'ensemble des manuels ... 20

2 Bases de notation musicale ... 21

2.1 Notation sur une seule portée ... 21
2.1.1 Contrôle de mesure ... 21
2.1.2 Altérations et armure ... 21
2.1.3 Liaisons ... 23
2.1.4 Articulations et nuances ... 24
2.1.5 Ajout de texte ... 25
2.1.6 Barres de ligature automatiques et manuelles ... 26
2.1.7 Commandes rythmiques avancées ... 26
2.2 Notes simultanées ... 27
2.2.1 Les expressions musicales en clair ... 27
2.2.2 Plusieurs portées ... 29
2.2.3 Regroupements de portées ... 30
2.2.4 Combinaison de notes en accords ... 31
2.2.5 Polyphonie sur une portée ... 31
2.3 Chansons ... 32
2.3.1 Écriture de chants simples ... 32
2.3.2 Alignement des paroles sur une mélodie ... 32
2.3.3 Paroles pour plusieurs portées ... 36
2.4 Dernières précisions ... 37
2.4.1 Organisation du code source avec des variables ... 37
2.4.2 Ajout de titres ... 38
2.4.3 Noms de note absolus ... 39
2.4.4 Après le tutoriel ... 40

3 Concepts fondamentaux ... 41

3.1 Organisation des fichiers LilyPond ... 41
3.1.1 Introduction à la structure de fichier LilyPond ... 41
3.1.2 La partition est une (unique) expression musicale composée ... 43
3.1.3 Expressions musicales imbriquées ... 46

3.1.4 Non-imbrication des crochets et liaisons ... 47
3.2 Les voix contiennent la musique ... 48
3.2.1 J'entends des Voix ... 48
3.2.2 Instanciation explicite des voix ... 53
3.2.3 Voix et paroles ... 56
3.3 Contextes et graveurs ... 59
3.3.1 Tout savoir sur les contextes ... 59
3.3.2 Création d'un contexte ... 60
3.3.3 Tout savoir sur les graveurs ... 62
3.3.4 Modification des propriétés d'un contexte ... 63
3.3.5 Ajout et suppression de graveurs ... 68
3.4 Extension des modèles ... 70
3.4.1 Soprano et violoncelle ... 71
3.4.2 Partition pour chœur à quatre voix mixtes ... 74
3.4.3 Écriture d'une partition à partir de zéro ... 79
3.4.4 Économie de saisie grâce aux identificateurs et fonctions ... 84
3.4.5 Conducteurs et parties ... 86

4 Retouche de partition ... 88
4.1 Retouches élémentaires ... 88
4.1.1 Introduction aux retouches ... 88
4.1.2 Objets et interfaces ... 88
4.1.3 Conventions de nommage des objets et propriétés ... 89
4.1.4 Méthodes de retouche ... 89
4.2 Le manuel des références internes ... 93
4.2.1 Propriétés des objets de rendu ... 93
4.2.2 Propriétés listées par interface ... 97
4.2.3 Types de propriétés ... 98
4.3 Apparence des objets ... 99
4.3.1 Visibilité et couleur des objets ... 99
4.3.2 Taille des objets ... 104
4.3.3 Longueur et épaisseur des objets ... 107
4.4 Positionnement des objets ... 108
4.4.1 Comportement automatique ... 108
4.4.2 Objets inclus dans la portée ... 109
4.4.3 Objets hors de la portée ... 112
4.5 Collisions d'objets ... 118
4.5.1 Déplacement d'objets ... 118
4.5.2 Correction des collisions d'objets ... 121
4.5.3 Exemple concret ... 126
4.6 Autres retouches ... 135
4.6.1 Autres utilisations des retouches ... 135
4.6.2 Utilisation de variables dans les retouches ... 137
4.6.3 Feuilles de style ... 138
4.6.4 Autres sources de documentation ... 142
4.6.5 Options ralentissant le traitement ... 144
4.6.6 Retouches avancées avec Scheme ... 144

Annexe A Modèles ... 146
A.1 Portée unique ... 146
A.1.1 Notes seules ... 146
A.1.2 Notes et paroles ... 146
A.1.3 Notes et accords ... 147
A.1.4 Notes, paroles et accords ... 148
A.2 Modèles pour piano ... 148
A.2.1 Piano seul ... 148
A.2.2 Chant et accompagnement ... 149
A.2.3 Piano et paroles entre les portées ... 150
A.2.4 Piano et nuances entre les portées ... 151
A.3 Quatuor à cordes ... 152
A.3.1 Quatuor à cordes ... 152
A.3.2 Parties pour quatuor à cordes ... 154
A.4 Ensemble vocal ... 156
A.4.1 Partition pour chœur à quatre voix mixtes ... 156
A.4.2 Partition pour chœur SATB avec réduction pour piano ... 158
A.4.3 Partition pour chœur SATB avec alignement des contextes ... 160
A.4.4 Partition pour chœur SATB, sur quatre portées ... 161
A.4.5 Couplet pour solo et refrain à deux voix ... 163
A.4.6 Hymnes et cantiques ... 165
A.4.7 Psalmodie ... 167
A.5 Orchestre ... 170
A.5.1 Orchestre, chœur et piano ... 170
A.6 Exemples de notation ancienne ... 173
A.6.1 Transcription de musique mensurale ... 173
A.6.2 Transcription du grégorien ... 178
A.7 Autres modèles ... 179
A.7.1 Symboles de jazz ... 179

Annexe B GNU Free Documentation License ... 186

Annexe C Index de LilyPond ... 193

1 Tutoriel

Ce tutoriel est une introduction au langage musical utilisé par LilyPond, qui vous permettra de faire fonctionner le logiciel pour produire une partition.

1.1 Compilation d'un fichier

Nous allons ici parler de la « compilation », ou comment LilyPond traite le fichier source que vous avez écrit, pour en faire quelque chose d'imprimable.
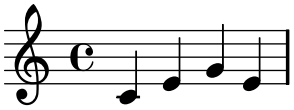
1.1.1 Saisie de la musique

Pour créer une partition avec LilyPond, on écrit un fichier texte, appelé fichier source, qui décrit la notation musicale. La *compilation* de ce fichier source par LilyPond produit un fichier graphique imprimable, et si on le désire un fichier MIDI qui peut être joué par un séquenceur.

Voici un premier exemple simple de fichier source LilyPond.

```
\version "2.14.2"
{
  c' e' g' e'
}
```

La compilation de ce fichier donnera quelque chose de sembable à l'image ci-dessous.



Il est aussi possible d'utiliser les noms de notes français « do re mi fa sol la si », en insérant au début du fichier la ligne `\include "italiano.ly"`.

> **Note :** Tout extrait de code LilyPond doit être entouré d'une `{` **paire d'accolades** `}`. De plus, pour éviter toute ambiguïté, il est préférable d'entourer les accolades par des espaces ou retours à la ligne. Bien que certains exemples de ce manuel ne comportent pas d'accolades, ne les oubliez pas dans vos partitions ! Pour plus d'informations sur l'affichage des exemples de cette documentation, consultez Section 1.4 [Bien lire le manuel], page 19.

De plus, LilyPond est **sensible à la casse** : le code '`{ c d e }`' est valide, alors que '`{ C D E }`' produira un message d'erreur.

Production de la partition

Dans cette section nous expliquerons quelles commandes exécuter et comment voir ou imprimer le résultat produit par LilyPond.

- Section 1.1.2 [MacOS X], page 2 Section 1.1.2 [MacOS X], page 2 (graphique)
- Section 1.1.3 [Windows], page 6 Section 1.1.3 [Windows], page 6 (graphique)
- Section 1.1.4 [Ligne de commande], page 13 Section 1.1.4 [Ligne de commande], page 13 (ligne de commande)

Notez qu'il existe plusieurs éditeurs de texte disponibles avec un bon support de LilyPond ; consultez Section "Facilités d'édition" dans *Informations générales*.

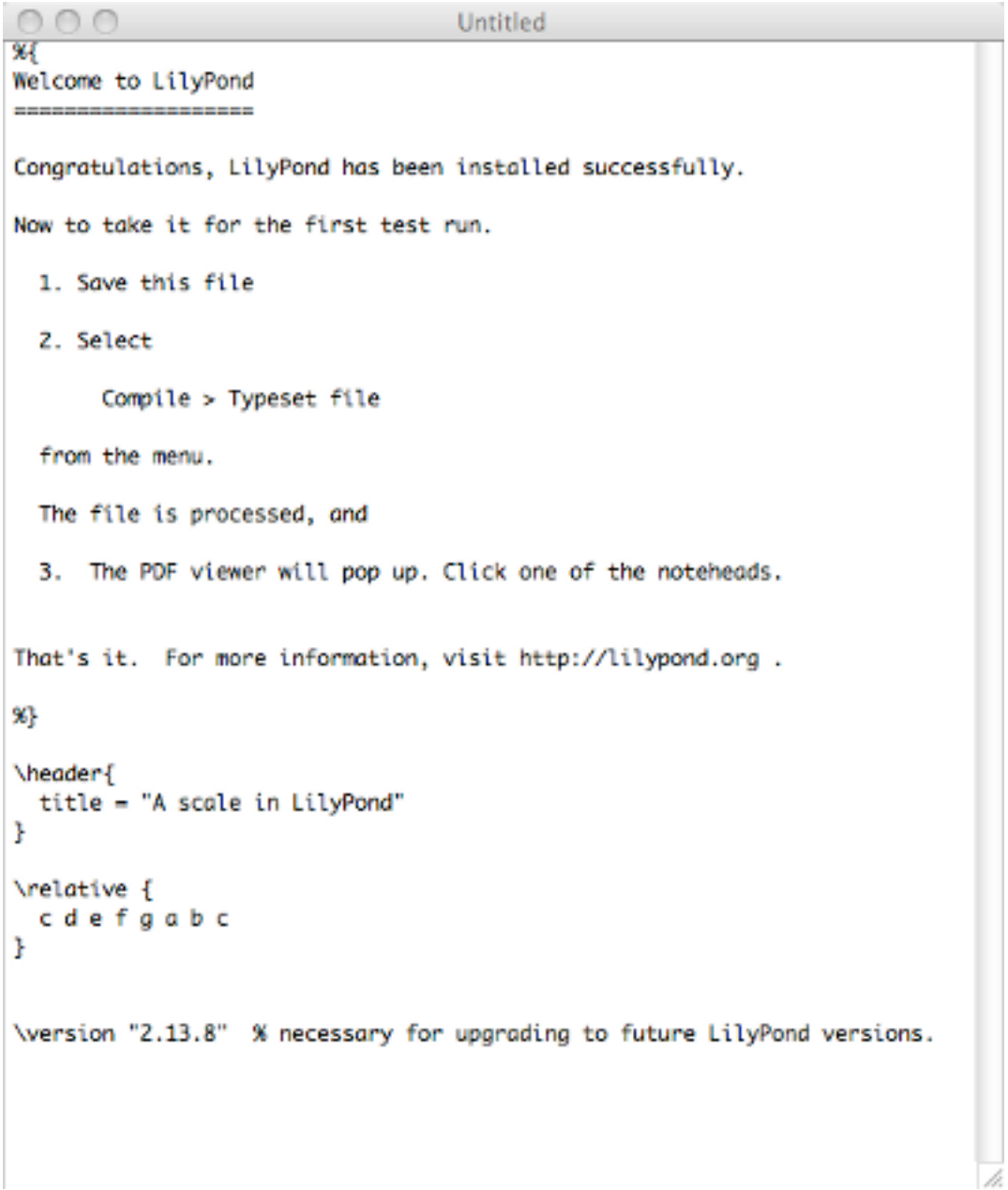
Note : Le premier démarrage de LilyPond peut prendre une minute ou deux, afin de générer la liste des polices du système. LilyPond démarre en principe plus rapidement lors des exécutions suivantes.

1.1.2 MacOS X

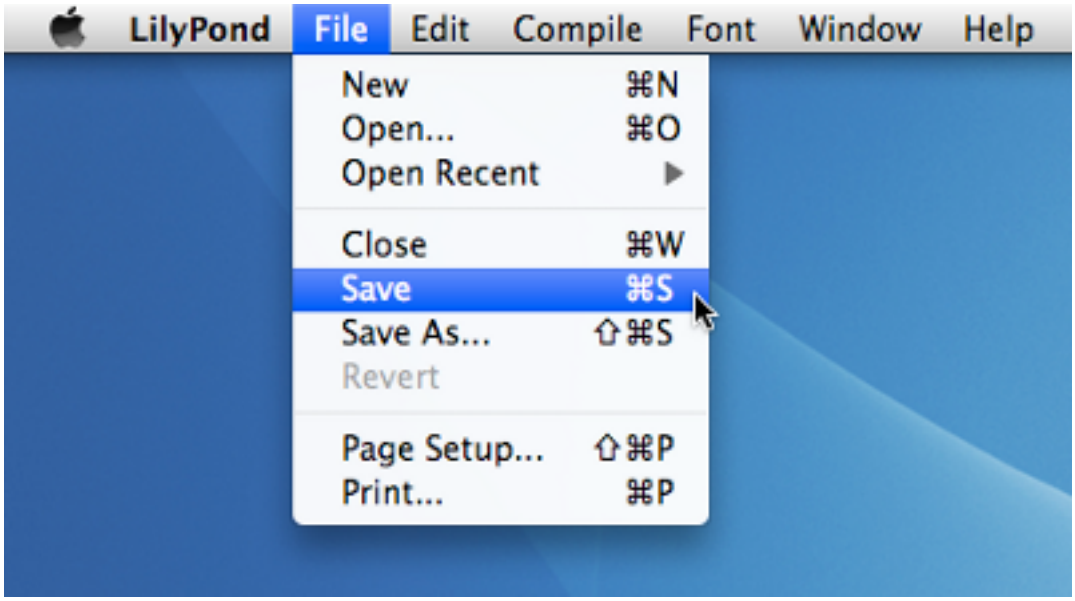
Note : Les instructions qui suivent concernent ceux qui utilisent le lanceur LilyPond. Si vous utilisez l'un des programmes mentionnés au chapitre Section "Facilités d'édition" dans *Informations générales*, référez-vous à leur documentation respective en cas de problème.

Étape 1. Création d'un fichier '`.ly`'

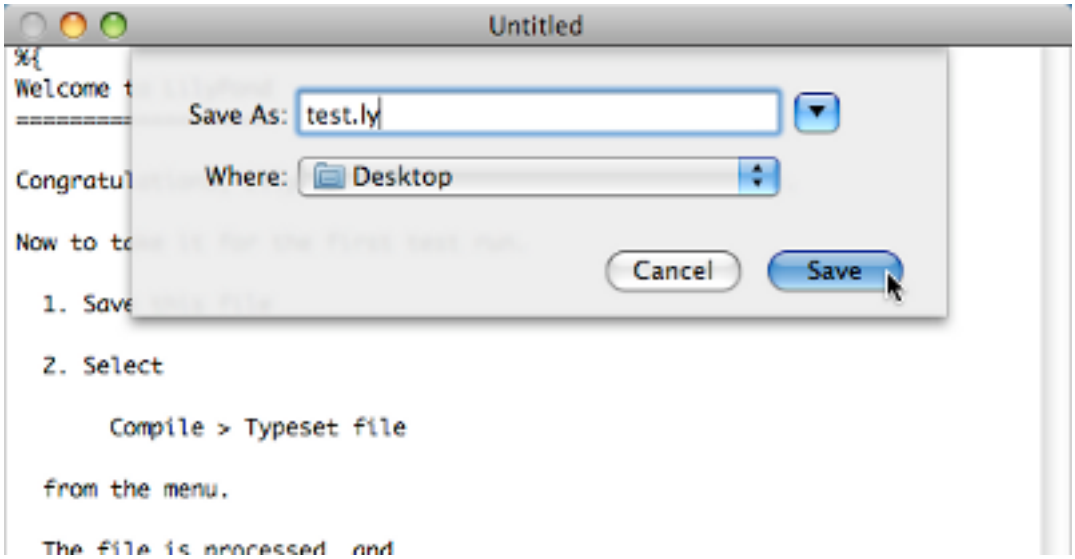
Lorsque vous faites un double clic sur `LilyPond.app`, un fichier exemple s'ouvre.



Dans le menu, en haut et à gauche de la fenêtre, sélectionnez `Fichier > Enregistrer`.

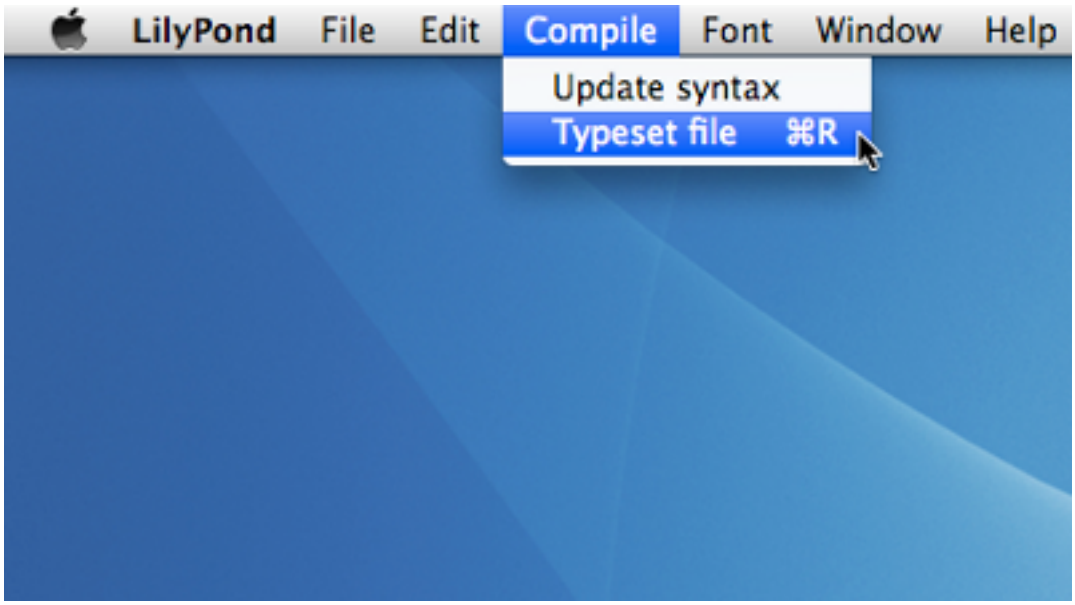


Attribuez un nom à votre fichier, par exemple '`test.ly`'.

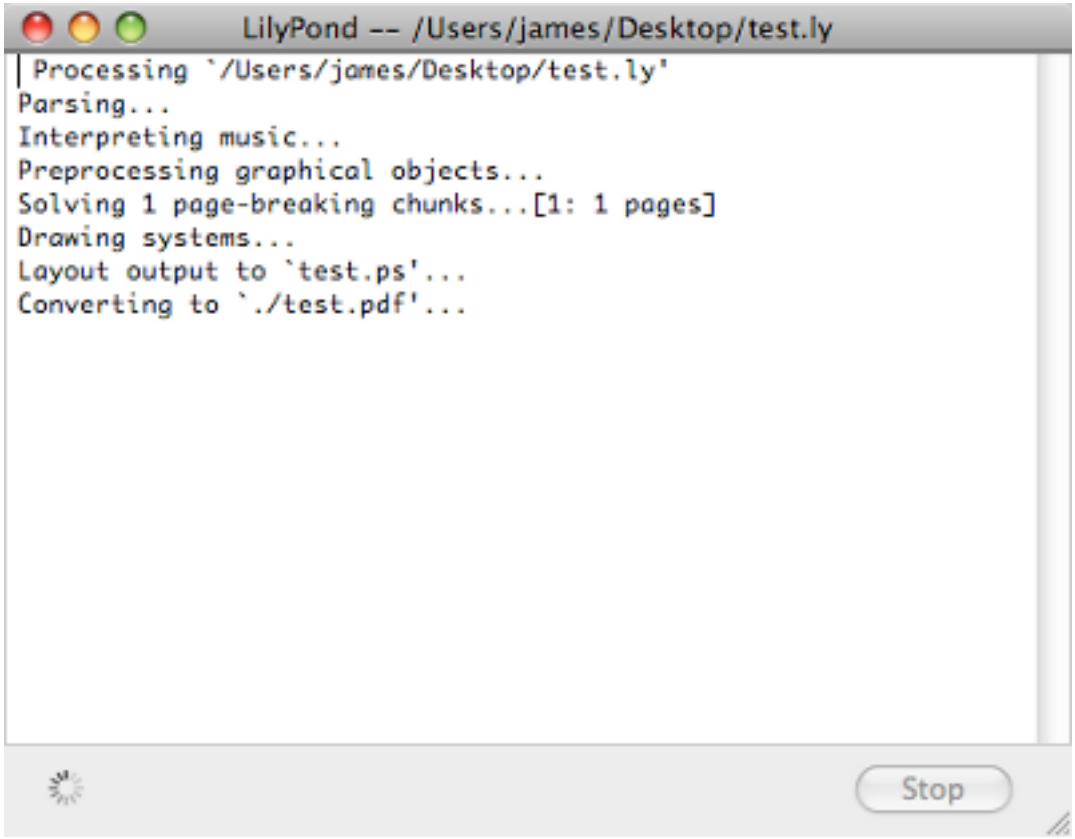


Étape 2. Compilation (avec LilyPad)

Dans le menu, sélectionnez `Compiler > Typeset`.

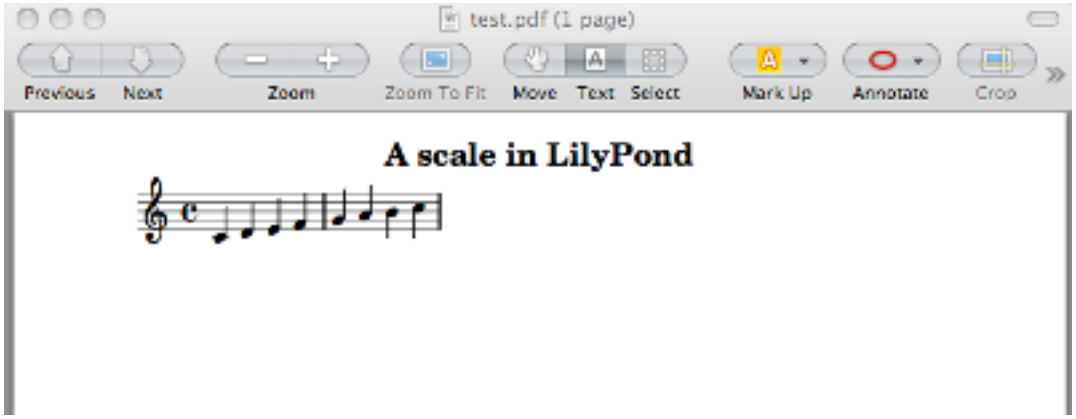


Une nouvelle fenêtre s'ouvre dans laquelle s'affiche le journal de compilation du fichier que vous venez de sauvegarder.



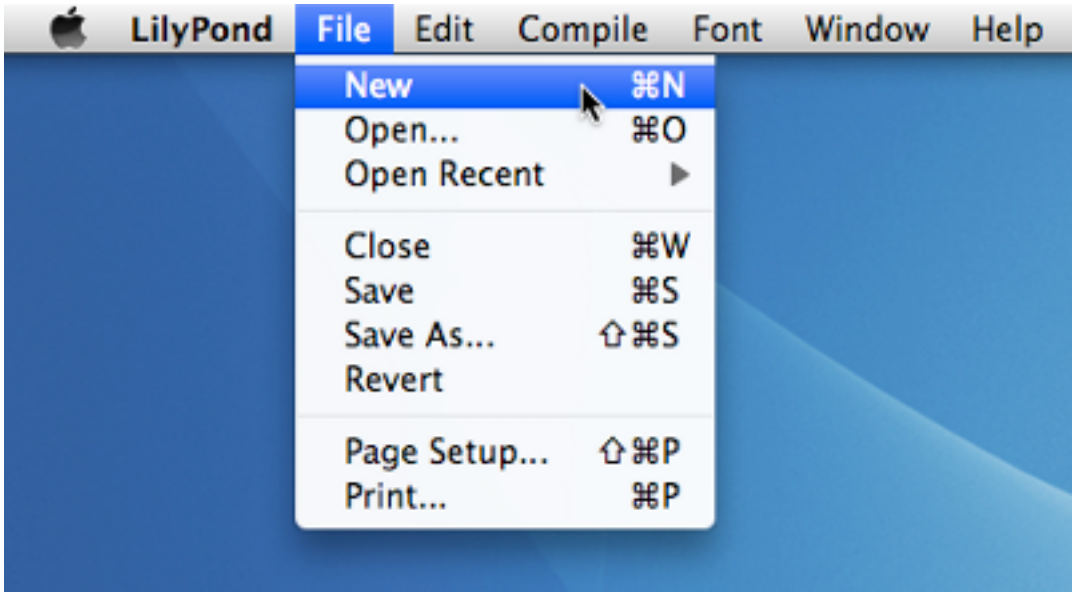
Étape 3. Visualisation du résultat

La compilation s'achève par la création d'un fichier PDF portant le même nom que le fichier source ; ce nouveau fichier sera automatiquement ouvert par votre lecteur PDF par défaut et affiché à l'écran.

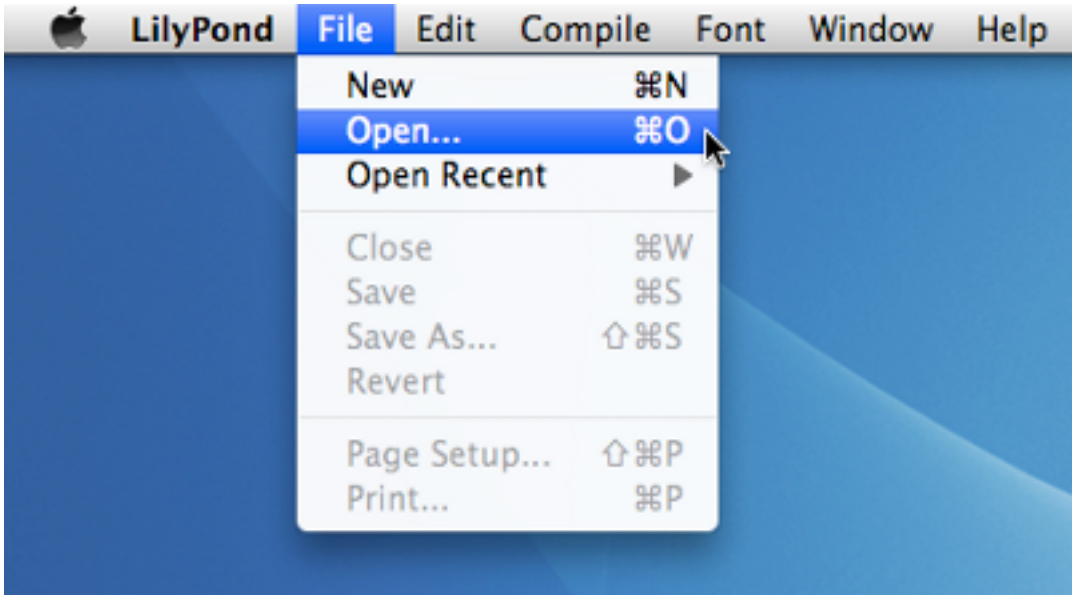


Autres commandes

Pour créer de nouveaux fichiers pour LilyPond, sélectionnez `Fichier > Nouveau`



ou `Fichier > Ouvrir` pour reprendre un fichier précédemment enregistré.



Pensez à toujours enregistrer votre travail avant de lancer l'option `Compile > Tyepset` du menu. Si le PDF n'apparaît pas, vérifiez que la fenêtre « log » ne comporte pas d'erreur.

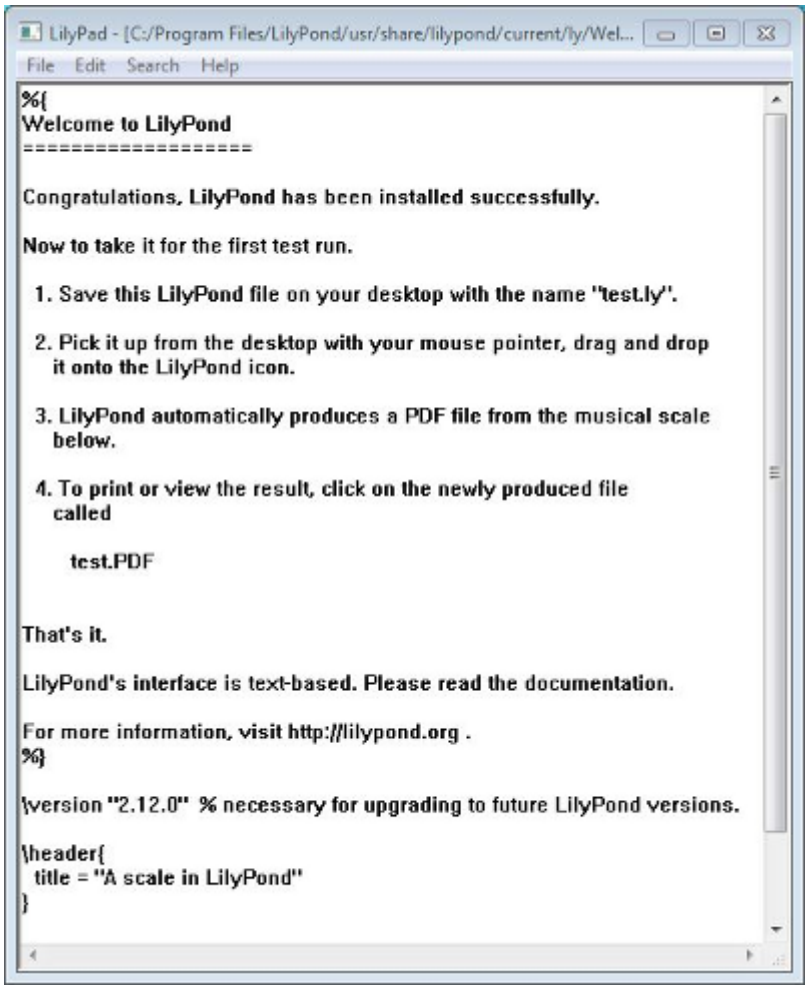
Si vous n'utilisez pas le lecteur de PDF par défaut de Mac OS et qu'un fichier résultant d'une précédente compilation est encore ouvert dans votre lecteur, la régénération de ce PDF peut bloquer tant que vous ne fermez pas le fichier ouvert.

1.1.3 Windows

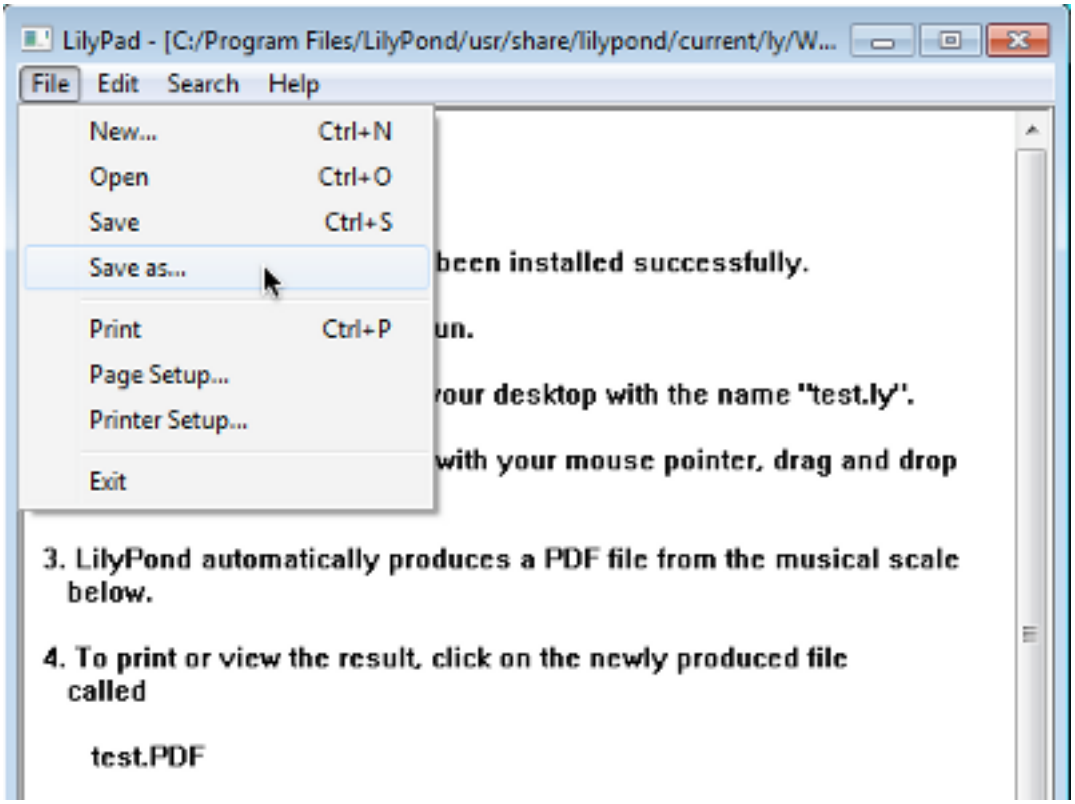
Note : Les instructions qui suivent partent du principe que vous utilisez l'éditeur LilyPad fourni. Si vous utilisez l'un des programmes répertoriés dans Section "Facilités d'édition" dans *Informations générales*, consultez sa documentation en cas de compilation infructueuse.

Étape 1. Création d'un fichier '.ly'

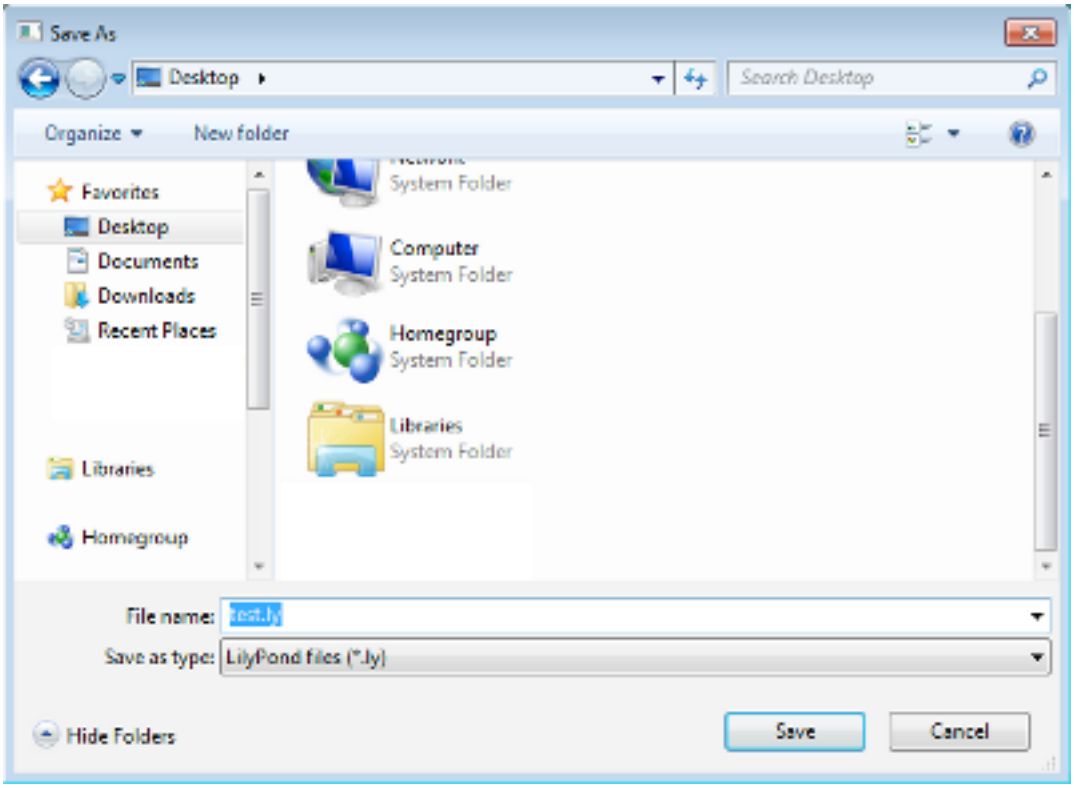
Double-cliquez sur l'icone LilyPond qui se trouve sur le bureau. S'ouvre alors un fichier d'exemple.



Dans le menu, sélectionnez `Fichier > Enregistrer sous`. Ne prenez pas l'option `Fichier > Enregistrer` pour ce fichier exemple : LilyPond attend un nom de fichier valide.

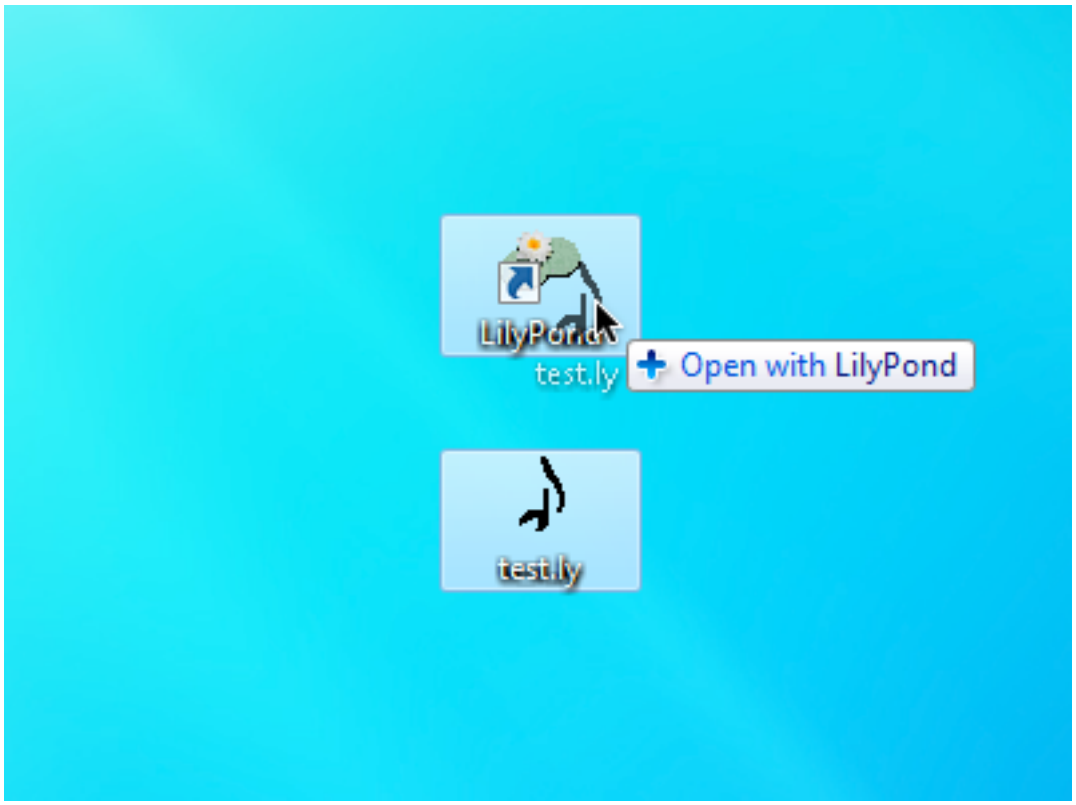


Affectez un nom à votre fichier, comme par exemple '`test.ly`'.

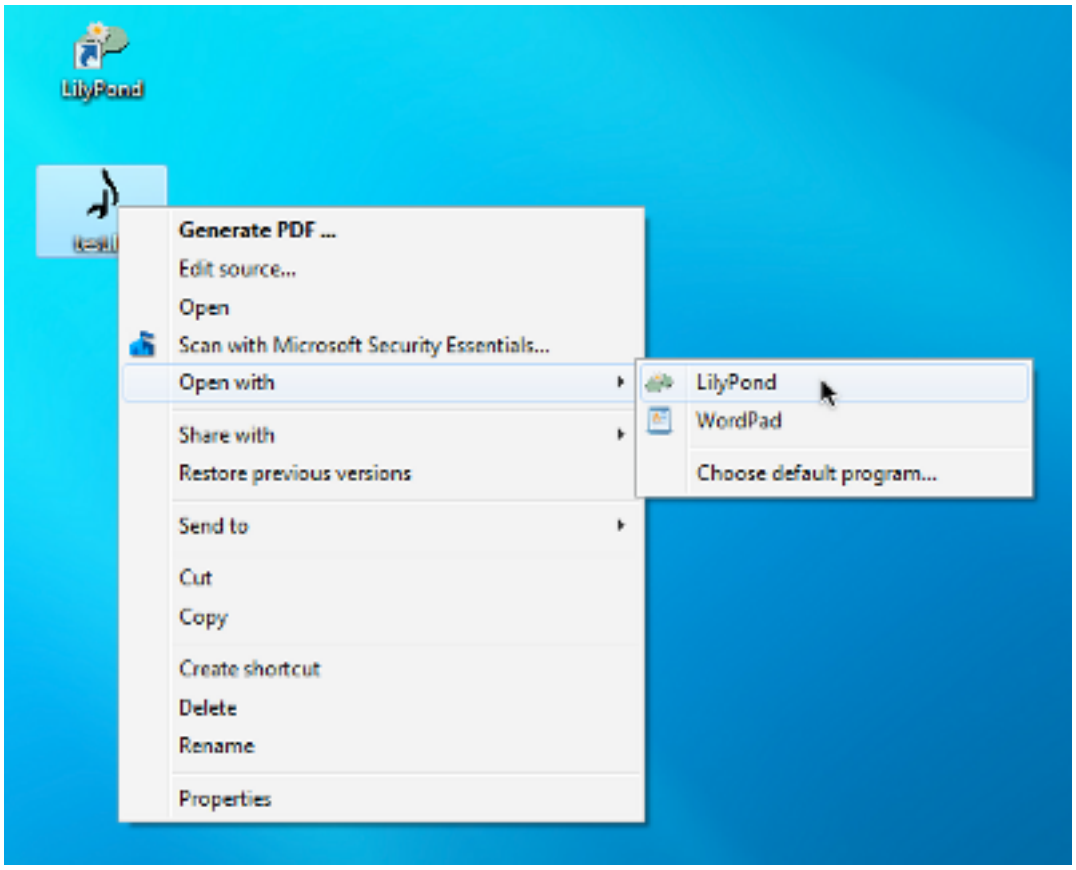


Étape 2a. Compilation par glisser-déposer

Selon votre préférence, vous pouvez compiler votre fichier en le faisant glisser puis en le déposant sur l'icone LilyPond



ou en ouvrant le menu contextuel par un clic-droit, puis en prenant l'option `Ouvrir avec > LilyPond`.

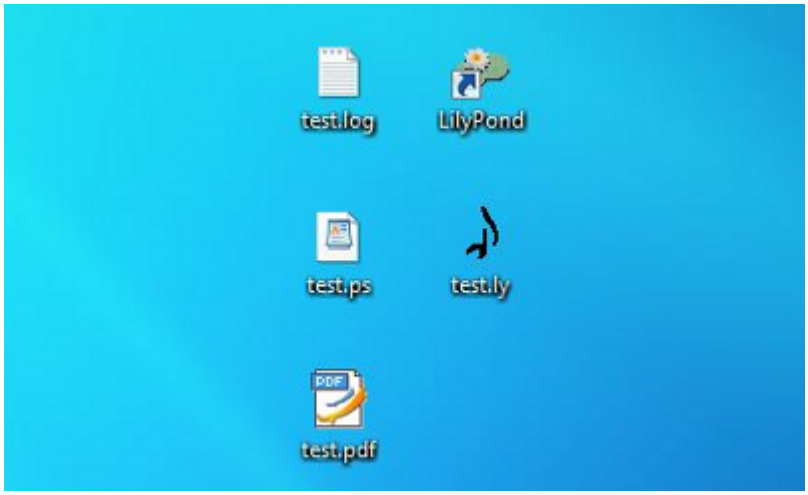


Étape 2b. Compilation par double-clic

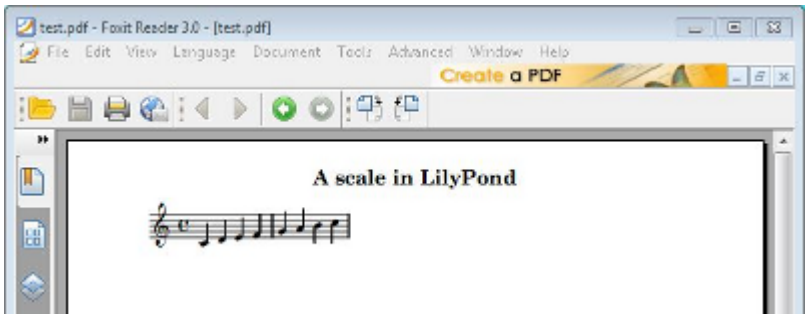
Vous pouvez aussi faire un double-clic sur le fichier '`test.ly`'.

Étape 3. Visualisation du résultat

Au cours de la compilation du fichier '`test.ly`', une fenêtre d'interpréteur de commande s'ouvre et se referme. Trois fichiers complémentaires seront générés pendant ce temps là.

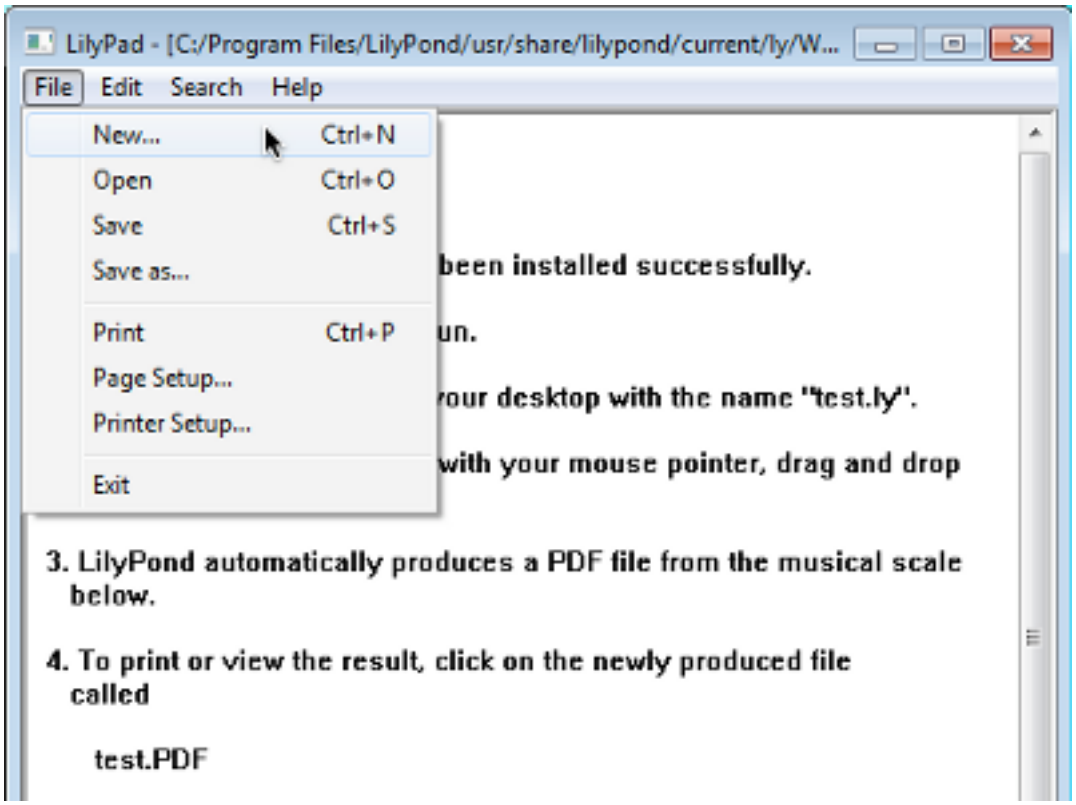


Le fichier PDF contient la gravure de votre fichier '`test.ly`'.

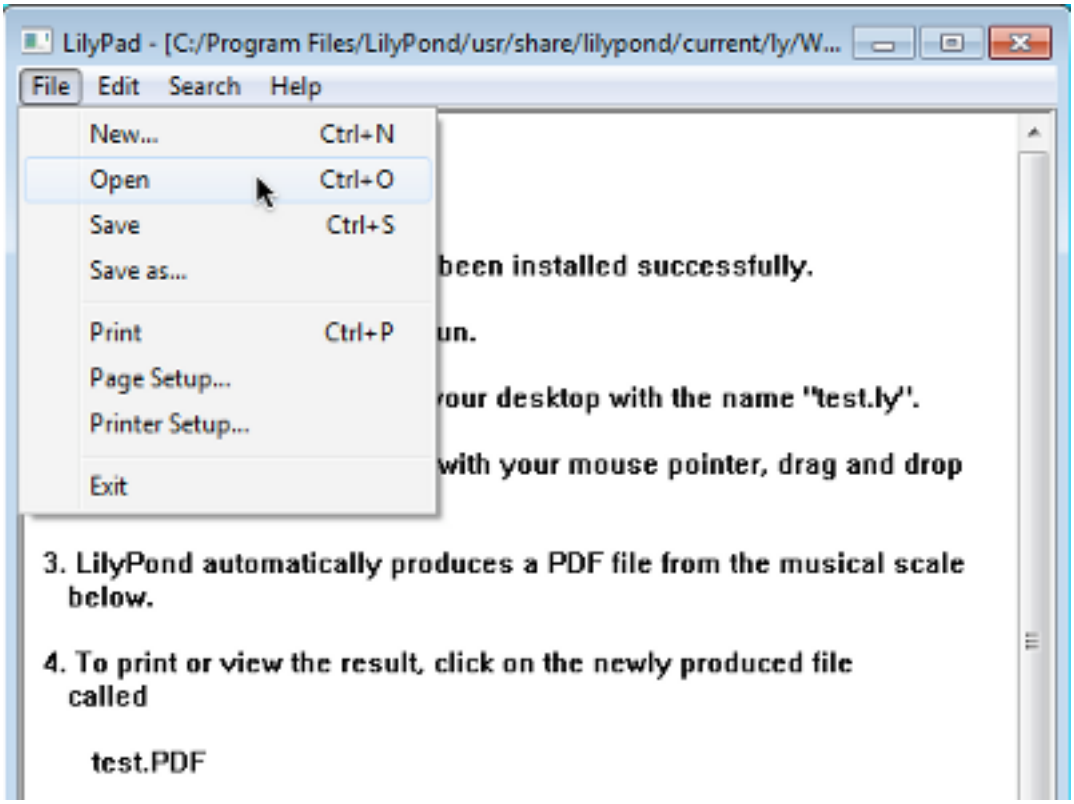


Autres commandes

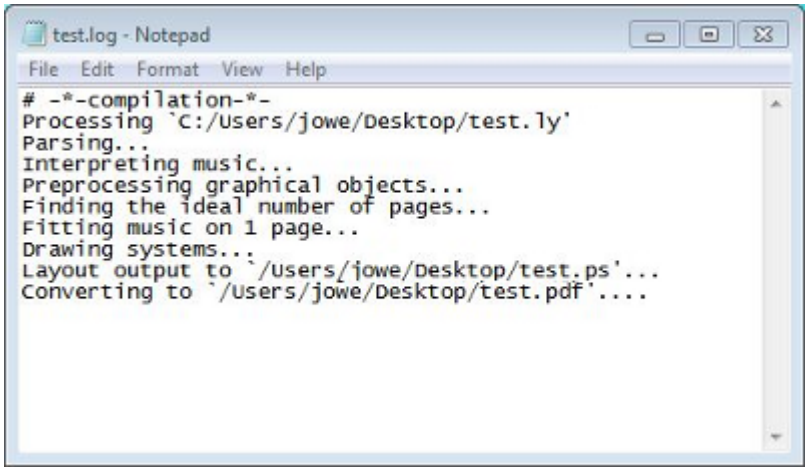
Pour créer un nouveau fichier, sélectionnez `Fichier > Nouveau` à parti de n'importe quel fichier déjà existant



ou bien `Fichier > Ouvrir` pour reprendre un fichier sauvegardé.



Pensez à toujours enregistrer votre travail avant de compiler votre fichier. Si LilyPond ne crée pas de PDF, consultez le fichier de journalisation – généré au fil du processus de compilation – et vérifiez qu'il ne comporte pas d'erreur.



Ce fichier journal est remplacé à chaque compilation de votre fichier LilyPond.

Le fichier PS est utilisé en interne par LilyPond pour créer le PDF. Il sera écrasé à chaque fois que vous relancerez la compilation de votre fichier.

Pensez à fermer le fichier dans votre lecteur de PDF à chaque fois que vous relancez la compilation, afin d'être sûr que celle-ci arrive à son terme.

1.1.4 Ligne de commande

> **Note :** Les instructions qui suivent supposent que vous êtes familier de la ligne de commande. Si vous utilisez l'un des programmes répertoriés dans Section "Facilités d'édition" dans *Informations générales*, consultez sa documentation en cas de compilation infructueuse.

Étape 1. Création d'un fichier '`.ly`'

Créez un fichier texte du nom de '`test.ly`' et saisissez :

```
\version "2.14.2"
{
c' e' g' e'
}
```

Étape 2. Compilation en ligne de commande

Pour traiter le fichier '`test.ly`', tapez ce qui suit à l'invite de commande :

```
lilypond test.ly
```

Vous verrez alors quelque chose qui ressemblera à :

```
GNU LilyPond 2.14.2
Traitement de « test.ly »
Analyse...
Interprétation en cours de la musique...
Pré-traitement des éléments graphiques...
Détermination du nombre optimal de pages...
Répartition de la musique sur une page...
Dessin des systèmes...
Sortie mise en page vers « test.ps »...
Conversion à « ./test.pdf »...
```

Suivant votre installation, ces messages peuvent être traduits ou non.

Étape 3. Visualisation du résultat

Vous pouvez à présent visualiser ou imprimer '`test.pdf`'.
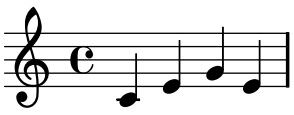
1.2 Composition d'un fichier source

This section introduces some basic LilyPond syntax to help get you started writing input files.

1.2.1 Notation simple

Il y a certains éléments graphiques de notation que LilyPond ajoute automatiquement. Dans l'exemple suivant, nous n'avons fourni que quatre hauteurs, mais LilyPond a ajouté une clef, un chiffre de mesure et du rythme.

```
{
  c' e' g' e'
}
```
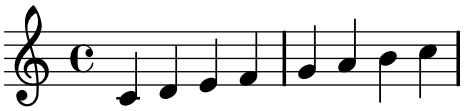


Ces valeurs automatiques simplifient la saisie du code source dans bien des cas ; nous verrons plus loin comment les indiquer explicitement.

Hauteurs

Glossaire musicologique : Section “hauteur” dans *Glossaire*, Section “intervalle” dans *Glossaire*, Section “gamme” dans *Glossaire*, Section “do central” dans *Glossaire*, Section “octave” dans *Glossaire*, Section “altération” dans *Glossaire*.
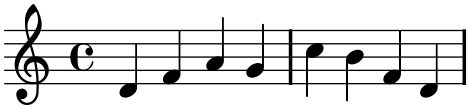
Le moyen le plus simple d’entrer des notes est d’utiliser le mode d’octaves relatives, ou mode `\relative`. Dans ce mode, l’octave de chaque note est déterminée automatiquement de façon à ce qu’elle soit le plus proche possible de la note précédente, c’est-à-dire de façon à ce que l’intervalle avec la note précédente soit au plus d’une quarte. Commençons par saisir une partition très simple, à savoir une gamme.

```
% set the starting point to middle C
\relative c' {
  c d e f
  g a b c
}
```



La note de départ est le *do central*. Chacune des notes qui suivent est placée à l’octave la plus proche de la note précédente – en d’autres termes, le premier `c` est le do central, entre la clef de sol et la clef de fa, puis est suivi par le *ré* le plus proche, et ainsi de suite. On peut bien sûr créer des mélodies avec de plus grands intervalles, toujours avec le mode `\relative` :
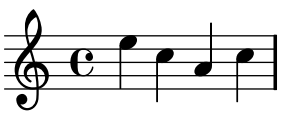
```
\relative c' {
  d f a g
  c b f d
}
```



Remarquez que cet exemple ne commence plus sur le *do central* : la première note – le `d` – est le ré qui en est le plus proche.
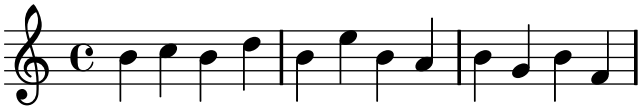
Dans l’exemple suivant, on remplace `c'` dans la commande `\relative c'` par `c''`, afin de calculer l’octave de la première note par rapport au do situé une octave au-dessus du do central :

```
% one octave above middle C
\relative c'' {
  e c a c
}
```



Le mode d’octaves relatives peut être déroutant au début, mais c’est souvent la façon la plus économique de saisir les hauteurs en utilisant le clavier de l’ordinateur de façon classique. Détaillons dans un exemple le calcul des octaves relatives. En partant d’un si sur la troisième ligne de la clé de sol, un do, un ré ou un mi sans indication d’octave particulière seront placés juste au-dessus du si, c’est-à-dire au plus à une quarte ascendante du si, alors qu’un la, un sol ou un fa seront placés juste en-dessous du si, c’est-à-dire au plus à une quarte descendante du si.
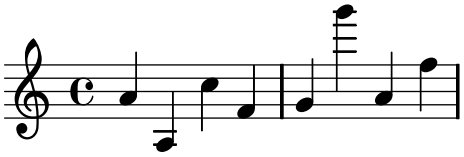
```
\relative c'' {
  b c  % c is 1 staff space up, so is the c above
  b d  % d is 2 up or 5 down, so is the d above
  b e  % e is 3 up or 4 down, so is the e above
  b a  % a is 6 up or 1 down, so is the a below
  b g  % g is 5 up or 2 down, so is the g below
  b f  % f is 4 up or 3 down, so is the f below
}
```



Notez que le calcul des octaves relatives **ne tient pas compte des altérations** des notes, dièse bémol ou bécarre.

Pour obtenir des intervalles supérieurs à une quarte, on peut ajouter des apostrophes `'` – qui font chacune monter la hauteur d'une octave – ou des virgules `,` – qui font chacune descendre la hauteur d'une octave – au nom de la note.

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



Pour déplacer une note deux octaves (ou davantage !) plus haut ou plus bas, il suffit de mettre deux (ou davantage) `'` ou `,` – attention cependant à bien mettre deux apostrophes `''`, et non un guillemet `"` ! C'est de cette même manière que l'on peut modifier la valeur de départ de `\relative c'`.

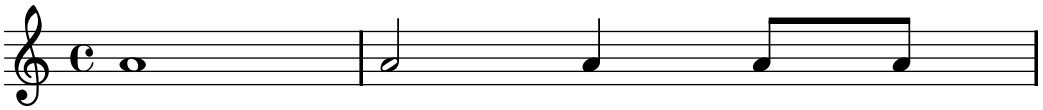
Durées et rythme
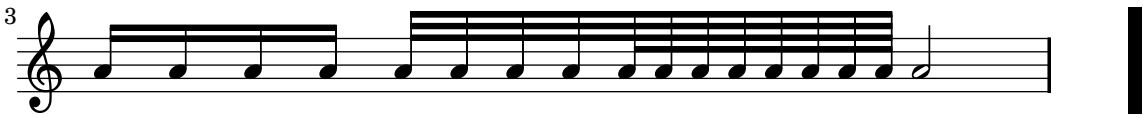
Glossaire musicologique : Section "ligature" dans *Glossaire*, Section "durée" dans *Glossaire*, Section "ronde" dans *Glossaire*, Section "blanche" dans *Glossaire*, Section "noire" dans *Glossaire*, Section "note pointée" dans *Glossaire*.

La *durée* d'une note est indiquée par un nombre qui suit sa hauteur : `1` pour une *ronde*, `2` pour une *blanche*, `4` pour une *noire* et ainsi de suite. Les *crochets* et *liens* sont ajoutés automatiquement.

Si aucune durée n'est indiquée pour une note, la dernière durée entrée est utilisée. En l'absence d'indication de durée, la première note est une noire.
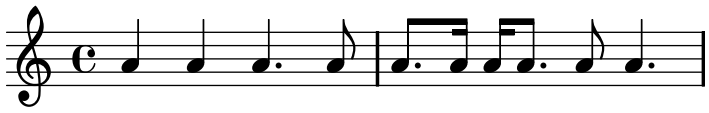
```
\relative c'' {
  a1
  a2 a4 a8 a
  a16 a a a a32 a a a a64 a a a a a a a2
}
```





Une *note pointée* s'obtient en ajoutant un point `.` à la valeur rythmique. Le point doit être précédé d'un nombre spécifiant la durée de base.

```
\relative c'' {
  a a a4. a8
  a8. a16 a a8. a8 a4.
}
```
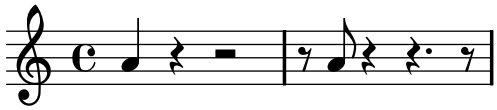


Silences

Glossaire musicologique : Section "silence" dans *Glossaire*.

On saisit un *silence* tout comme une note, mais avec la lettre `r` (pour *rest*).

```
\relative c'' {
  a r r2
  r8 a r4 r4. r8
}
```
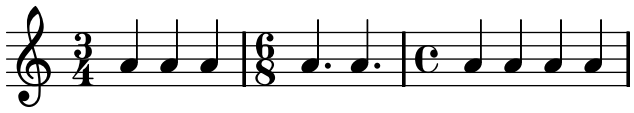


Métrique

Glossaire musicologique : Section "métrique" dans *Glossaire*.

La *métrique*, aussi appelée *chiffre de mesure*, peut être définie à l'aide de la commande `\time` :

```
\relative c'' {
  \time 3/4
  a4 a a
  \time 6/8
  a4. a
  \time 4/4
  a4 a a a
}
```
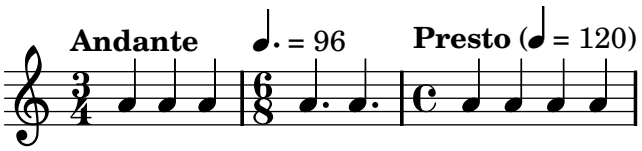


Indication de tempo

Glossaire musicologique : Section "indication de tempo" dans *Glossaire*, Section "métronome" dans *Glossaire*.

La commande `\tempo` permet de stipuler aussi bien le *tempo* que le métronome :

```
\relative c'' {
  \time 3/4
  \tempo "Andante"
  a4 a a
```

```
  \time 6/8
  \tempo 4. = 96
  a4. a
  \time 4/4
  \tempo  "Presto" 4 = 120
  a4 a a a
}
```
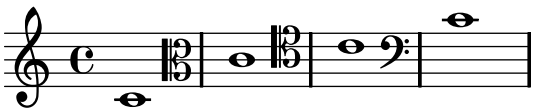


Clefs

Glossaire musicologique : Section “clef” dans *Glossaire*.

La *clef* peut être définie à l’aide de la commande `\clef` :
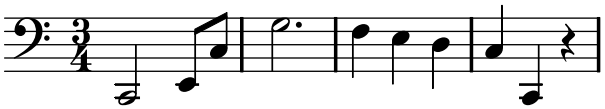
```
\relative c' {
  \clef "treble"
  c1
  \clef "alto"
  c1
  \clef "tenor"
  c1
  \clef "bass"
  c1
}
```



Tout ensemble

Voici un bref exemple qui rassemble tous les éléments que nous déjà vus :

```
\relative c, {
  \clef "bass"
  \time 3/4
  c2 e8 c'
  g'2.
  f4 e d
  c4 c, r
}
```



Voir aussi

Manuel de notation : Section “Écriture des hauteurs de note” dans *Manuel de notation*, Section “Écriture du rythme” dans *Manuel de notation*, Section “Écriture des silences” dans *Manuel de notation*, Section “Gravure du rythme” dans *Manuel de notation*, Section “Gravure des hauteurs” dans *Manuel de notation*.

1.2.2 Travail sur les fichiers d'entrée

Le traitement des fichiers source de LilyPond est semblable à celui du code de nombreux langages de programmation. La casse est prise en compte, et les caractères considérés comme espaces ont généralement peu d'importance. Les expressions sont délimitées par des accolades `{ }`, et les commentaires par `%` ou `%{ ... }` .

Si cette phrase vous paraît incompréhensible, ne vous en faites pas ! Expliquons tous ces termes :

- **La version** : Tout fichier LilyPond devrait porter mention de la version. Cette mention prend la forme d'une ligne décrivant le numéro de la version utilisée lors de la confection du fichier en question, comme ici :

  ```
  \version "2.14.2"
  ```

  Il est d'usage de porter cette mention au tout début du fichier LilyPond.

  Mentionner la version est important pour, au moins, les deux raisons suivantes : cela permet dans un premier temps de faciliter les opérations de mise à jour automatisée au fil de l'évolution de la syntaxe de LilyPond. Ensuite, cela met en évidence le minimum requis pour pouvoir compiler votre fichier.

  Si vous ne le mentionnez pas, Lilypond vous rappellera à l'ordre durant la compilation.

- **La casse** : LilyPond est sensible à la casse, c'est à dire qu'une lettre capitale n'a pas la même valeur qu'une lettre minuscule. Les notes, par exemple, doivent être entrées en minuscule : `{ c d e }` est un code valide, alors que `{ C D E }` produira un message d'erreur.

- **Les espaces multiples** : LilyPond ne tient pas compte du nombre d'espaces, de tabulations ou de retours à la ligne. `{ c d e }` a le même sens que `{ c  d  e }` ou que

  ```
  { c4                        d
                      e   }
  ```

  Bien sûr, ce dernier exemple est illisible. Une bonne habitude à prendre est d'indenter les blocs de code avec des doubles espaces :

  ```
  {
    c4 d e
  }
  ```

  L'espace est néanmoins **nécessaire** pour séparer nombre d'éléments syntaxiques les uns des autres. En d'autres termes, s'il est toujours possible d'*ajouter* des espaces, il faut prendre garde à ne pas trop en *supprimer*. En effet, l'absence d'une espace peut avoir des conséquences et entraîner une erreur ; aussi nous vous invitons à toujours insérer une espace avant et après chaque élément syntaxique, comme avant et après une accolade.

- **Expressions musicales** : Tout morceau saisi dans LilyPond doit être placé entre `{` **accolades** `}`. Ces caractères indiquent à LilyPond que ce bloc de texte représente une et une seule expression musicale, tout comme les parenthèses `()` en mathématiques. Pour éviter toute ambiguïté, il est préférable d'entourer ces accolades d'espaces ou de retours à la ligne.

  Un appel de fonction – `\relative c' { ... }` par exemple – compte également comme une seule expression musicale.

- **Les commentaires** : Un commentaire est une indication pour tout lecteur humain d'un fichier source de musique ; il est ignoré lors de la compilation et n'a donc aucun effet sur la partition imprimée. On distingue deux types de commentaires. Le commentaire de fin de ligne, introduit par le symbole `%` : tout ce qui suit ce symbole sur la même ligne sera ignoré. Par convention, un commentaire qui occupe une ligne entière se place juste *au-dessus* de la ligne à laquelle il fait référence.

  ```
  a4 a a a
  % ce commentaire fait référence aux si
  ```

```
b2 b
```

Le bloc de commentaire, qui peut occuper plusieurs lignes, voire toute une section : tout ce qui se trouve entre `%{` et `%}` est ignoré. Les blocs de commentaires ne peuvent s'imbriquer, ce qui signifie que vous ne pouvez pas placer un commentaire-bloc à l'intérieur d'un autre commentaire-bloc. Si jamais vous essayez, vous verrez que la première occurence de `%}` terminera « les *deux* commentaires-blocs ». Le fragment suivant met en évidence quelques usages possibles des commentaires :

```
% voici les notes de "ah vous dirai-je maman"
 c4 c g' g a a g2

%{
   Ces lignes et les notes qui suivent
   seront ignorées, car elles se trouvent
   dans un bloc de commentaire.

   f f e e d d c2
%}
```

1.3 Gestion des erreurs

Parfois, LilyPond ne produit pas le résultat escompté. Voici quelques pistes à suivre pour vous aider à éviter de telles déconvenues.

1.3.1 Quand ça ne fonctionne pas

Résoudre les problèmes rencontrés avec LilyPond est une gageure pour ceux qui ne connaissent que des interfaces graphiques puisque rien n'empêche de créer du code erroné. En pareil cas, il suffit souvent d'un peu de logique pour être en mesure d'identifier les causes du problème et le résoudre simplement. Le chapitre Section "Résolution de problèmes" dans *Utilisation des programmes* liste quelques directives à ce propos.

1.3.2 Erreurs courantes

Il peut arriver qu'un message d'erreur ne soit pas suffisamment explicite pour solutionner le problème. Quelques cas des plus courants sont répertoriés au chapitre Section "Quelques erreurs des plus courantes" dans *Utilisation des programmes*.

1.4 Bien lire le manuel

Nous allons voir ici comment consulter la documentation le plus efficacement possible. Nous en profiterons pour vous présenter quelques particularités de la domcumentation en ligne.

1.4.1 Matériel incomplet

Comme nous l'avons vu dans Section 1.2.2 [Travail sur les fichiers d'entrée], page 18, un code LilyPond doit être encadré par des accolades `{ }` ou bien par `\relative c'' { ... }`. Cependant, dans la suite de ce manuel, la plupart des exemples ne feront pas apparaître ces signes.

Pour reproduire les exemples, vous pouvez copier et coller le code affiché, mais **à condition** d'ajouter `\relative c'' { }` de la façon suivante :
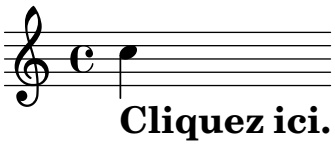
```
\relative c'' {
  ...collez ici votre exemple...
}
```

Pourquoi avoir omis les accolades ? La plupart des exemples de ce manuel peuvent être insérés au milieu d'un morceau de musique plus long. Il n'y a donc aucune raison d'ajouter

`\relative c'' { }` à ces exemples – en effet, il n'est pas possible d'insérer une expression `\relative` à l'intérieur d'une autre expression `\relative`. Si nous mettions tous nos exemples dans une expression `\relative`, vous ne pourriez plus copier un bref exemple de la documentation pour le coller dans vos pièces.

1.4.2 Exemples cliquables

Beaucoup de gens apprennent à utiliser les programmes en les essayant et en bidouillant avec. C'est également possible avec LilyPond. Si vous cliquez sur une image dans la version HTML de ce manuel, vous verrez exactement le code LilyPond utilisé pour générer cette image. Essayez sur cette image :



En copiant-collant le code à partir du commentaire « ly snippet » vers un fichier test, vous aurez un modèle de base pour faire vos expériences. Pour obtenir une gravure à l'identique, copiez tout le code à partir de « Start cut-&-pastable section ».

1.4.3 Vue d'ensemble des manuels

La documentation de LilyPond est relativement abondante. Ceci peut dérouter les nouveaux utilisateurs qui ne savent pas par quoi commencer ou bien sont tentés de faire l'impasse sur des passages importants.

> **Note :** Nous vous invitons à ne pas faire l'impasse sur les chapitres importants de notre documentation, au risque de vous retrouver complètement perdu lorsque vous aborderez d'autres parties.

- **Avant de vous lancer dans** *quoi que ce soit*, lisez le Chapitre 1 [Tutoriel], page 1 contenu dans le manuel d'initiation ainsi que les Chapitre 2 [Bases de notation musicale], page 21. Si vous y trouvez des termes que vous ne connaissez pas, n'hésitez pas à consulter le Section "Glossaire" dans *Glossaire*.
- **Avant de vous lancer dans la réalisation d'une partition complète**, lisez le chapitre Chapitre 3 [Concepts fondamentaux], page 41 du manuel d'initiation. Vous pourrez alors vous reporter aux parties qui vous intéresseront dans le Section "Manuel de notation" dans *Manuel de notation*.
- **Avant de modifier les réglages par défaut**, consultez le chapitre Chapitre 4 [Retouche de partition], page 88 du manuel d'initiation.
- **Avant de vous lancer dans un projet d'envergure**, lisez le chapitre Section "Suggestions pour la saisie de fichiers LilyPond" dans *Utilisation des programmes* du manuel d'utilisation.

2 Bases de notation musicale

Suite au premier contact avec le Chapitre 1 [Tutoriel], page 1, voyons comment créer de belles partitions utilisant une notation musicale courante
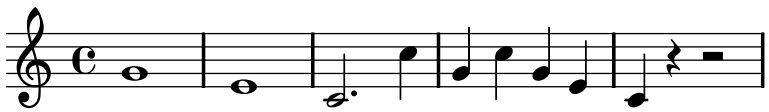
2.1 Notation sur une seule portée

Cette section présente la notation courante dont on a besoin pour écrire une voix sur une portée.

2.1.1 Contrôle de mesure

Les contrôles de barre de mesure – *bar checks* en anglais – ne sont à priori pas strictement nécessaires. Ils permettent cependant d'indiquer directement dans le code saisi – à l'aide d'une barre verticale `|` – l'emplacement des barres de mesure. Grâce à ces contrôleurs, LilyPond sera capable de déterminer si la durée que vous avez attribuée aux notes correspondent bien à longueur de la mesure qui les contient. Les contrôles de mesure apportent aussi clarté et organisation à votre code.

```
g1 | e1 | c2. c'4 | g4 c g e | c4 r r2 |
```



Voir aussi

Manuel de notation : Section "Vérification des limites et numéros de mesure" dans *Manuel de notation*.

2.1.2 Altérations et armure

> **Note :** Si, comme nombre de nouveaux utilisateurs, ce qui suit vous paraît déroutant, lisez cette partie jusqu'au bout, à plus forte raison si vous n'avez jamais fait de solfège !
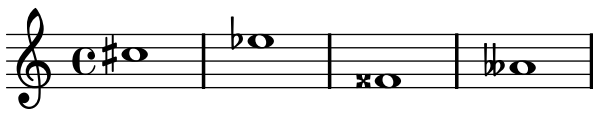
Altérations

Glossaire musicologique : Section "dièse" dans *Glossaire*, Section "bémol" dans *Glossaire*, Section "double dièse" dans *Glossaire*, Section "double bémol" dans *Glossaire*, Section "altération" dans *Glossaire*.

Dans la notation par défaut, on obtient un *dièse* en ajoutant `is` au nom de la note, et un *bémol* en ajoutant `es`. Comme vous pouvez vous y attendre, un double dièse ou double bémol s'obtiennent en ajoutant `isis` ou `eses`. Cette syntaxe est dérivée de la convention de dénomination des notes dans les langues nordiques et germaniques, comme l'allemand ou le hollandais.

Cependant, si vous utilisez la commande `\language "italiano"` pour entrer les noms de note français au lieu des noms hollandais, il faudra ajouter un `d` pour obtenir un dièse, et un `b` pour un bémol. Le double dièse et le double bémol s'obtiennent en ajoutant respectivement `dd` et `bb`. Pour en savoir plus sur les autres langues disponibles, consultez Section "Nom des notes dans d'autres langues" dans *Manuel de notation*.

```
cis1 ees fisis, aeses
```
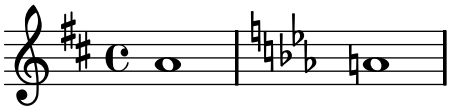


Armures

Glossaire musicologique : Section “armure” dans *Glossaire*, Section “majeur” dans *Glossaire*, Section “mineur” dans *Glossaire*.

L’armure est déterminée par la commande `\key`, suivie d’une hauteur puis de `\major` (majeur) ou `\minor` (mineur).

```
\key d \major
a1
\key c \minor
a
```



Attention aux armures et aux hauteurs
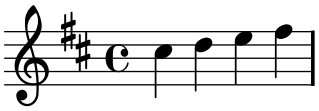
Glossaire musicologique : Section “altération” dans *Glossaire*, Section “armure” dans *Glossaire*, Section “hauteur” dans *Glossaire*, Section “bémol” dans *Glossaire*, Section “bécarre” dans *Glossaire*, Section “dièse” dans *Glossaire*, Section “transposition” dans *Glossaire*, Section “Noms de note” dans *Glossaire*.

La combinaison de l’*armure* et des hauteurs de note – y compris les altérations – permet à LilyPond de déterminer dans quel cas imprimer des *altérations accidentelles*. L’armure n’affecte que les altérations *imprimées*, et non les hauteurs réelles ! Cette fonctionnalité est souvent source de confusion pour les nouveaux utilisateurs, aussi expliquons-la en détail.

LilyPond fait une distinction nette entre le contenu musical et la mise en forme. L’altération d’une note – *bémol*, *bécarre* ou *dièse* – fait partie de sa hauteur, et relève donc du contenu musical. La gravure ou non d’une altération accidentelle – un *signe* bémol, bécarre ou dièse – devant la note correspondante est une question qui relève de la mise en forme. La gravure d’une partition suit des règles, en particulier des règles d’indication des altérations accidentelles. Les hauteurs de note, en revanche, relèvent de ce que vous voulez entendre ; et, dans la mesure où la musique que vous entrez est censée être celle que vous voulez entendre, LilyPond, qui n’est chargé que de la gravure, ne les choisira pas à votre place.

Dans cet exemple,
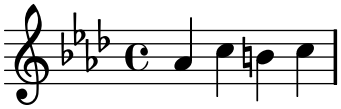
```
\key d \major
cis4 d e fis
```



aucune note n’a d’altération accidentelle, et pourtant vous devrez entrer le `is` pour les notes `cis` et `fis`.

Le code `b` ne veut pas dire « Imprimez-moi un point noir sur la troisième ligne de la portée. » Cela signifie plutôt : « Ici se trouve une note dont la hauteur est un si naturel. » Avec une armure de la bémol majeur, ce `si` est flanqué d’un bécarre accidentel :

```
\key aes \major
aes4 c b c
```



Prenons un autre exemple : imaginez-vous devant un piano ; dès lors que vous voulez enfoncer l'une des touches noires, il vous faudra **ajouter** un `-is` ou un `-es` au nom de la note.

Ajouter explicitement toutes les altérations demande un peu plus d'effort dans la phase de saisie, mais cela facilite grandement la *transposition*. De plus, les altérations accidentelles peuvent ainsi être imprimées suivant plusieurs conventions. Pour connaître les différentes manières dont les altérations accidentelles peuvent être imprimées, consultez Section "Altérations accidentelles automatiques" dans *Manuel de notation*.

Voir aussi

Manuel de notation : Section "Nom des notes dans d'autres langues" dans *Manuel de notation*, Section "Altérations" dans *Manuel de notation*, Section "Altérations accidentelles automatiques" dans *Manuel de notation*, Section "Armure" dans *Manuel de notation*.
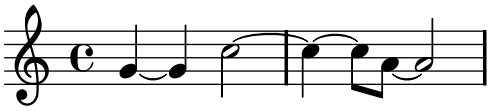
2.1.3 Liaisons

Liaisons de prolongation

Glossaire musicologique : Section "liaison de tenue" dans *Glossaire*.

Pour créer une liaison de prolongation – parfois aussi appelée liaison de tenue –, on ajoute un tilde `~` à la première note liée.

```
g4~ g c2~ | c4 ~ c8 a8 ~ a2 |
```
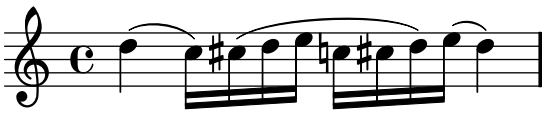


Liaisons d'articulation

Glossaire musicologique : Section "liaison" dans *Glossaire*, Section "phrasé" dans *Glossaire*.

Une liaison d'articulation ou *legato* peut englober plusieurs notes. Les notes de départ et d'arrivée sont suivies respectivement d'un signe `(` et `)`.
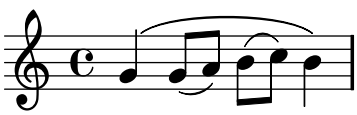
```
d4( c16) cis( d e c cis d) e( d4)
```



Liaisons de phrasé

De plus longues liaisons, dites de phrasé, sont délimitées par `\(` et `\)`. Il est possible d'avoir en même temps des legatos et des phrasés, mais pas plusieurs liaisons de phrasé ou de *legato* à la fois.

```
g4\( g8( a) b( c) b4\)
```
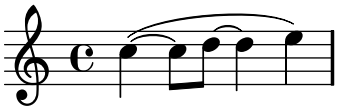


Attention aux types de liaison

Glossaire musicologique : Section "articulation" dans *Glossaire*, Section "liaison" dans *Glossaire*, Section "liaison de tenue" dans *Glossaire*.

Une liaison d'articulation ou de phrasé ressemble à une liaison de prolongation, mais n'a pas la même signification. Alors qu'une liaison de prolongation ne peut relier que deux notes de même hauteur, le *legato* indique une articulation de plusieurs notes, éventuellement en grand nombre. Les liaisons de tenue peuvent être enchâssées dans un *legato* ou un phrasé.

```
c4~( c8 d~ d4 e)
```



Voir aussi

Manuel de notation : Section "Liaisons de prolongation" dans *Manuel de notation*, Section "Liaisons d'articulation" dans *Manuel de notation*, Section "Liaisons de phrasé" dans *Manuel de notation*.
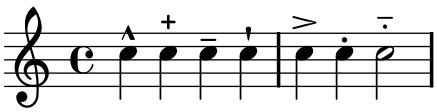
2.1.4 Articulations et nuances

Articulations

Glossaire musicologique : Section "articulation" dans *Glossaire*.

Des *articulations* peuvent être ajoutées à une note, au moyen d'un tiret - suivi d'un caractère :

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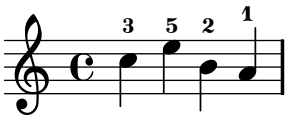


Doigtés

Glossaire musicologique : Section "doigté" dans *Glossaire*.
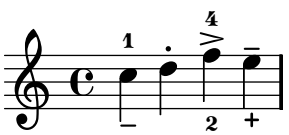
De même, des indications de doigté peuvent être ajoutées à une note en utilisant un tiret ('-') et le chiffre à écrire :

```
c4-3 e-5 b-2 a-1
```



Articulations et doigtés sont habituellement placés automatiquement, mais vous pouvez spécifier leur positionnement en utilisant ^ (en haut) ou _ (en bas). Vous pouvez aussi utiliser plusieurs articulations sur la même note. Dans la plupart des cas, cependant, il est bon de laisser LilyPond déterminer l'emplacement de l'articulation.

```
c4_-^1 d^. f^4_2-> e^-_+
```
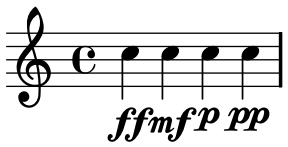


Nuances

Glossaire musicologique : Section “nuances” dans *Glossaire*, Section “crescendo” dans *Glossaire*, Section “decrescendo” dans *Glossaire*.
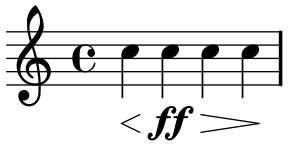
On obtient un signe de *nuance* en ajoutant à la note les lettres du signe, précédées d’un anti-slash `\` :

```
c4\ff c\mf c\p c\pp
```



Crescendos et *decrescendos* débutent avec les commandes `\<` et `\>`. Ils se terminent soit par une nuance d’arrivée, par exemple `\f`, soit par la commande `\!` :

```
c4\< c\ff\> c c\!
```



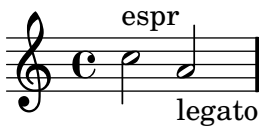
Voir aussi

Manuel de notation : Section “Articulations et ornements” dans *Manuel de notation*, Section “Doigtés” dans *Manuel de notation*, Section “Nuances” dans *Manuel de notation*.

2.1.5 Ajout de texte

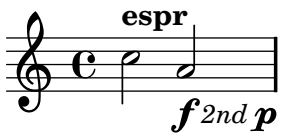
On peut ajouter du texte à une partition :

```
c2^"espr" a_"legato"
```



Pour mettre en forme du texte, on utilise la commande `\markup` :

```
c2^\markup{ \bold espr}
a2_\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
}
```



Voir aussi
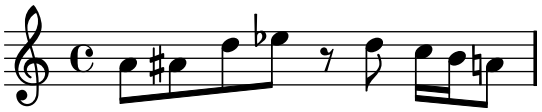
Manuel de notation : Section “Ajout de texte” dans *Manuel de notation*.

2.1.6 Barres de ligature automatiques et manuelles

Glossaire musicologique : Section “ligature” dans *Glossaire*.
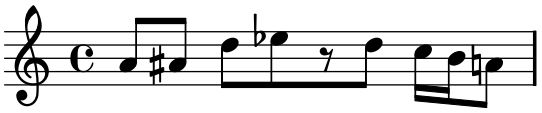
Toutes les barres de ligature sont dessinées automatiquement :

```
a8 ais d ees r d c16 b a8
```
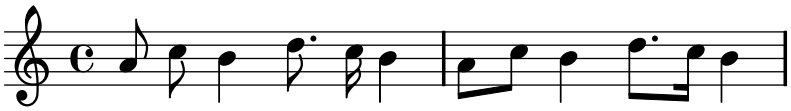


Lorsqu'on n'aime pas la manière dont les notes sont automatiquement groupées, il est possible de les ligaturer manuellement, en marquant la première note à attacher d'un crochet ouvrant `[` et la dernière d'un crochet fermant `]`.

```
a8[ ais] d[ ees r d] c16 b a8
```



Pour désactiver les barres de ligature automatiques pour des passages entiers, utilisez la commande `\autoBeamOff`, et utilisez `\autoBeamOn` pour les réactiver.

```
\autoBeamOff
a8 c b4 d8. c16 b4 |
\autoBeamOn
a8 c b4 d8. c16 b4 |
```



Voir aussi

Manuel de notation : Section “Barres de ligature automatiques” dans *Manuel de notation*, Section “Barres de ligature manuelles” dans *Manuel de notation*.
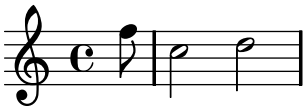
2.1.7 Commandes rythmiques avancées

Mesure incomplète

Glossaire musicologique : Section “anacrouse” dans *Glossaire*.

On crée une levée (ou anacrouse) avec la commande `\partial`, suivie d'une durée : `\partial 4` produit une levée d'une noire et `\partial 8` d'une croche.

```
\partial 8 f8 |
c2 d |
```

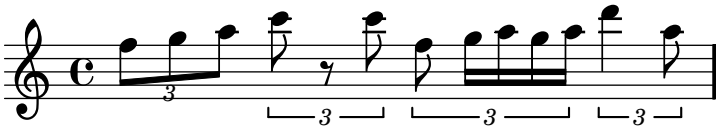


Nolets

Glossaire musicologique : Section “valeur d’une note” dans *Glossaire*, Section “triolet” dans *Glossaire*.

Les *nolets* sont créés avec la commande `\times`, qui prend deux arguments : une fraction et une expression musicale. La durée des notes de l’expression musicale est multipliée par la fraction. Par exemple les notes d’un *triolet* durent les deux tiers de la durée de leur notation réelle, cette fraction est donc de 2/3 pour les triolets :

```
\times 2/3 { f8 g a }
\times 2/3 { c8 r c }
\times 2/3 { f,8 g16[ a g a] }
\times 2/3 { d4 a8 }
```
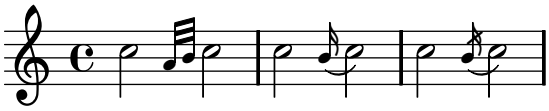


Notes d’ornement

Glossaire musicologique : Section “ornements” dans *Glossaire*, Section “acciacature” dans *Glossaire*, Section “appogiature” dans *Glossaire*.

Des *notes d’ornement* s’obtiennent en appliquant la commande `\grace`, `\appoggiatura` ou `\acciaccatura` à une expression musicale :

```
c2 \grace { a32[ b] } c2 |
c2 \appoggiatura b16 c2 |
c2 \acciaccatura b16 c2 |
```



Voir aussi

Manuel de notation : Section “Notes d’ornement” dans *Manuel de notation*, Section “Nolets” dans *Manuel de notation*, Section “Levées” dans *Manuel de notation*.

2.2 Notes simultanées

Cette section traite de situations où l’on a plus d’une note à la fois : plusieurs instruments, plusieurs voix ou portées pour un même instrument (le piano, par exemple), et les accords.

En théorie musicale, la polyphonie désigne une musique constituée de plusieurs voix ; dans LilyPond, ce terme désigne les situations où il y a plus d’une voix sur une même portée.

2.2.1 Les expressions musicales en clair
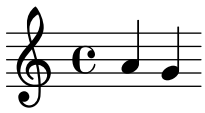
Dans les fichiers source LilyPond, la musique est représentée par ce qu’on appelle des *expressions musicales*. En soi, une seule note peut constituer une expression musicale :

```
a4
```
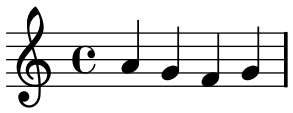


Mettre un groupe de notes entre accolades crée une nouvelle expression musicale, appelée *expression musicale composée*. En voici un exemple avec deux notes :

```
{ a4 g4 }
```



La mise entre accolades d'une séquence d'expressions musicales – des notes par exemple – signifie qu'elles doivent être jouées successivement, les unes après les autres. Le résultat est une expression, qui peut elle-même être regroupée séquentiellement avec d'autres expressions. Ici, l'expression de l'exemple précédent est combinée à deux notes :

```
{ { a4 g } f g }
```



Analogie avec les expressions mathématiques

Ce mécanisme est similaire aux formules mathématiques : une grosse formule est créée en assemblant plusieurs petites formules. De telles formules sont appelées expressions, elles ont une définition récursive, de telle sorte que vous pouvez fabriquer des expressions arbitrairement longues et complexes. Par exemple :

```
1

1 + 2

(1 + 2) * 3

((1 + 2) * 3) / (4 * 5)
```

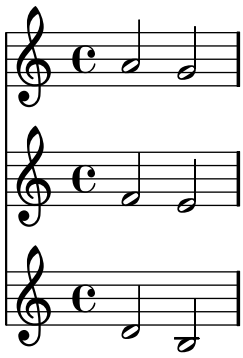
Ceci est une suite d'expressions, où chacune est contenue dans la suivante. Les expressions les plus simples sont les nombres, et de plus grandes expressions sont produites en combinant des expressions avec des opérateurs – comme `+`, `*` et `/` – et des parenthèses. Tout comme les expressions mathématiques, les expressions musicales peuvent être imbriquées avec une profondeur arbitraire, ce qui est nécessaire pour des partitions complexes comme de la musique polyphonique.

Expressions musicales simultanées – plusieurs portées

Glossaire musicologique : Section "polyphonie" dans *Glossaire*.

Cette technique est utile pour de la musique *polyphonique*. Pour entrer une musique avec plusieurs voix ou plusieurs portées, nous pouvons aussi combiner *en parallèle* les expressions : deux voix qui doivent être jouées en même temps, sont entrées comme une combinaison simultanée de deux expressions. Une expression musicale « simultanée » est formée en entourant les expressions entre `<<` et `>>`. Dans l'exemple suivant, trois expressions (contenant chacune deux notes distinctes) sont combinées simultanément.

```
\relative c'' {
  <<
    { a2 g }
    { f2 e }
    { d2 b }
  >>
}
```
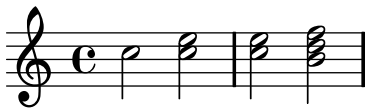


Notez que nous avons ici indenté chaque niveau du fichier d'entrée avec un nombre d'espaces différent. LilyPond se moque – ou presque – de l'espace qu'il peut y avoir ou non au début d'une ligne, mais un code bien indenté est bien plus lisible par des humains.

> **Note :** La hauteur de chaque note saisie est relative à la précédente, mais pas au `c''` de la commande `\relative` de départ.

Expressions musicales simultanées – une seule portée

Pour déterminer le nombre de portées, LilyPond regarde le début de la première expression. Si c'est une seule note, une seule portée est produite ; si c'est une expression simultanée, plusieurs portées sont produites. Nous avons dans l'exemple ci-dessous une expression complexe ; dans la mesure où elle débute par une note seule, elle sera produite sur une unique portée.

```
\relative c'' {
  c2 <<c e>> |
  << { e2 f } { c <<b d>> } >> |
}
```
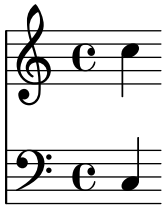


2.2.2 Plusieurs portées

Comme nous l'avons vu dans Section 2.2.1 [Les expressions musicales en clair], page 27, un fichier d'entrée LilyPond est fait d'expressions musicales. Si la partition commence par plusieurs expressions simultanées, LilyPond créera plusieurs portées. Cependant, il est plus facile de prévoir le nombre de portées si on les crée explicitement, ce que nous allons voir.

Pour créer plus d'une portée, on ajoute `\new Staff` au début de chaque partie de la musique constituant une portée. Ces éléments `Staff` sont ensuite combinés en parallèle avec `<<` et `>>`, comme ci-dessous.

```
\relative c'' {
  <<
    \new Staff { \clef "treble" c4 }
    \new Staff { \clef "bass" c,,4 }
  >>
}
```



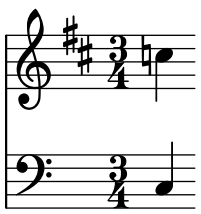
La commande `\new` introduit un « contexte de notation ». Un contexte de notation est un environnement dans lequel les événements musicaux – comme les notes ou les commandes `\clef` – sont interprétés. Pour des pièces simples, ces contextes sont créés automatiquement. Pour des pièces plus complexes, il est préférable de spécifier explicitement les contextes, afin de s'assurer que chaque fragment aura sa propre portée.

Il existe différents types de contextes. Les contextes `Score` (partition), `Staff` (portée) et `Voice` (voix) gèrent la notation de la mélodie, alors que `Lyrics` gère les paroles et `ChordNames` imprime des noms d'accord.

En terme de syntaxe, ajouter `\new` devant une expression musicale crée une plus grande expression musicale. En reprenant la comparaison précédente, cela ressemble au signe *moins* en mathématiques. La formule $(4+5)$ est une expression, donc $-(4+5)$ constitue une plus grande expression.

Les chiffres de métrique indiqués sur une portée affectent toutes les autres portées. En revanche l'armure d'une portée n'affecte *pas* les autres portées. Ces caractéristiques par défaut se justifient par le fait que l'utilisation d'instruments transpositeurs est bien plus fréquente que la musique polyrythmique.

```
\relative c'' {
  <<
    \new Staff { \clef "treble" \key d \major \time 3/4 c4 }
    \new Staff { \clef "bass" c,,4 }
  >>
}
```



2.2.3 Regroupements de portées
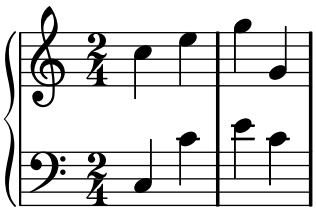
Glossaire musicologique : Section “accolade” dans *Glossaire*, Section “portée” dans *Glossaire*, Section “système” dans *Glossaire*.

La musique pour piano s'écrit sur deux portées reliées par une *accolade*. La gravure de ce type de portée est semblable à l'exemple de musique polyphonique de Section 2.2.2 [Plusieurs portées], page 29, mais maintenant cette expression entière est interprétée dans un contexte `PianoStaff` :

```
\new PianoStaff <<
  \new Staff ...
  \new Staff ...
>>
```

Voici un bref exemple :

```
\relative c'' {
  \new PianoStaff <<
    \new Staff { \time 2/4 c4 e | g g, | }
    \new Staff { \clef "bass" c,,4 c' | e c | }
  >>
}
```



Vous pouvez générer d'autres formes de regroupement, avec `\new GrandStaff` pour une partition d'orchestre, ou `\new ChoirStaff` qui sied particulièrement aux partitions chorales. Chacun de ces regroupements constitue un contexte à part entière, avec ses particularités, tant au niveau du signe qui regroupe les portées au sein d'un *système* qu'au niveau de l'étendue des barres de mesure.

Voir aussi
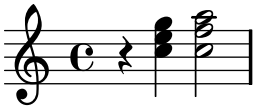
Manuel de notation : Section "Instruments utilisant des portées multiples" dans *Manuel de notation*, Section "Gravure des portées" dans *Manuel de notation*.

2.2.4 Combinaison de notes en accords

Glossaire musicologique : Section "accord" dans *Glossaire*.

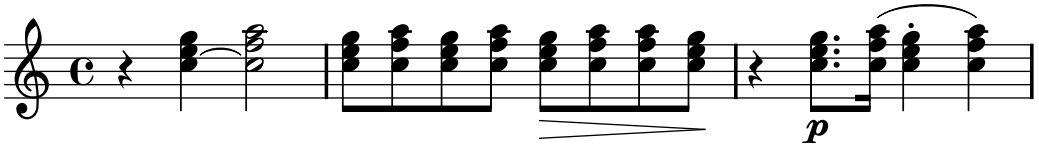
Nous avons vu précédemment comment combiner des notes simultanément, en les encadrant par des chevrons doubles `<<` et `>>`. Pour produire des accords simples, c'est-à-dire une superposition de notes de même durée, on encadre les hauteurs de notes par des chevrons simples `<` et `>`, et on écrit la durée juste après.

```
r4 <c e g> <c f a>2
```



Beaucoup d'éléments de notation que l'on peut attacher à une note simple, comme une liaison, un crochet indiquant un début ou une fin de lien, un signe d'articulation, peuvent être également attachés à un accord : il faut ajouter ces indications après les hauteurs et la durée, donc *à l'extérieur* des chevrons.

```
r4 <c e g>~ <c f a>2 |
<c e g>8[ <c f a> <c e g> <c f a>] <c e g>\>[ <c f a> <c   f a> <c e g>]\! |
r4 <c e g>8.\p <c f a>16( <c e g>4-. <c f a>) |
```



Voir aussi

Manuel de notation : Section "Notes en accords" dans *Manuel de notation*.

2.2.5 Polyphonie sur une portée

Bien que LilyPond gère la musique polyphonique sans difficulté, cela fait appel à des concepts que nous n'avons pas encore abordés. C'est la raison pour laquelle nous ne nous étendrons pas tout de suite sur ce sujet et préférons vous inciter à consulter les chapitres dédiés à l'étude de ces concepts.

Voir aussi

Manuel d'initiation : Section 3.2 [Les voix contiennent la musique], page 48.

Manuel de notation : Section "Notes simultanées" dans *Manuel de notation*.

2.3 Chansons
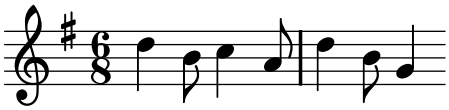
Cette section présente l'écriture vocale et les partitions de variété.

2.3.1 Écriture de chants simples

Glossaire musicologique : Section "lyrics" dans *Glossaire*.
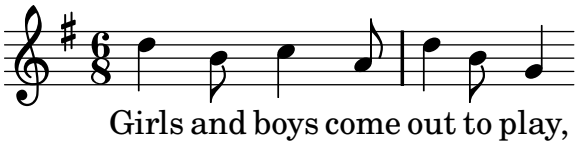
Prenons une mélodie toute simple, la comptine *Girls and boys come out to play.*

```
\relative c'' {
  \key g \major
  \time 6/8
  d4 b8 c4 a8 | d4 b8 g4
}
```



Des *paroles* peuvent être associées à ces notes, en les combinant avec la commande `\addlyrics`. On entre les paroles en séparant chaque syllable par une espace :

```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4
  }
  \addlyrics {
    Girls and boys come | out to play,
  }
>>
```



> **Note :** Il est primordial de séparer l'accolade fermant les paroles de la dernière syllabe – par une espace ou un saut de ligne – au risque de voir apparaître une Section "Erreur renvoyant à ../ly/init.ly" dans *Utilisation des programmes*.

Notez les doubles chevrons `<< ... >>` encadrant toute la pièce ; ils indiquent simplement que la musique et les paroles se produisent en même temps.

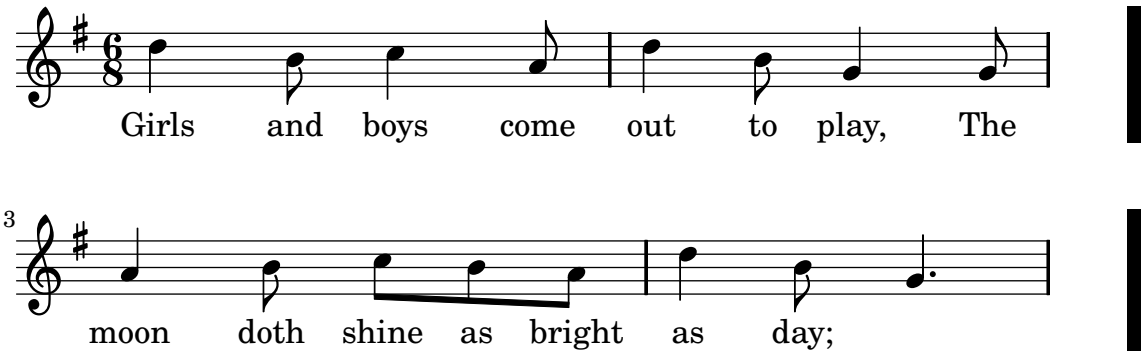
2.3.2 Alignement des paroles sur une mélodie

Glossaire musicologique : Section "mélisme" dans *Glossaire*, Section "ligne d'extension" dans *Glossaire*.

La deuxième ligne de la comptine précédente est *The moon doth shine as bright as day.* Ajoutons-la au code.

```
<<
  \relative c'' {
    \key g \major
```

```
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c b a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine as | bright as day; |
  }
>>
```
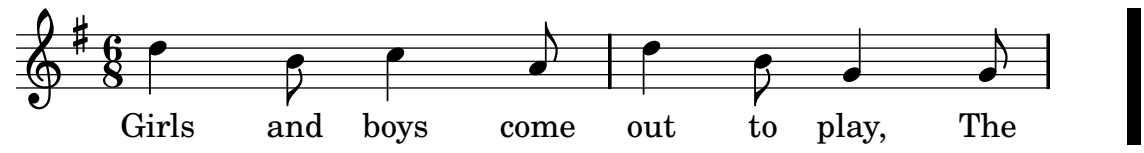


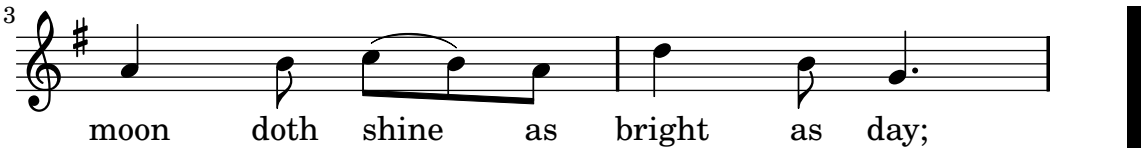
Si vous compilez ce code en l'état, vous verrez apparaître :

```
song.ly:12:29: Avertissement : échec du contrôle de mesure (barcheck) à : 5/8
    The | moon doth shine as
                              | bright as day; |
song.ly:12:46: Avertissement : échec du contrôle de mesure (barcheck) à : 3/8
    The | moon doth shine as | bright as day;
                                                |
```

Voici qui illustre bien l'utilité des contrôles de barre de mesure ! Remarquez que les paroles ajoutées ne s'alignent pas bien avec les notes. Le mot *shine* devrait être chanté sur deux notes au lieu d'une. On appelle ceci un *mélisme* : il s'agit d'une seule syllabe chantée sur plus d'une note. Il existe plusieurs façons d'étaler une sylabe sur plusieurs notes, la plus simple étant de lier les notes du mélisme. Pour les détails, consultez Section 2.1.3 [Liaisons], page 23.
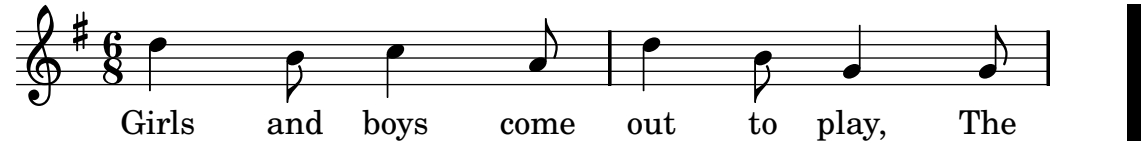
```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c( b) a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine as | bright as day; |
  }
>>
```

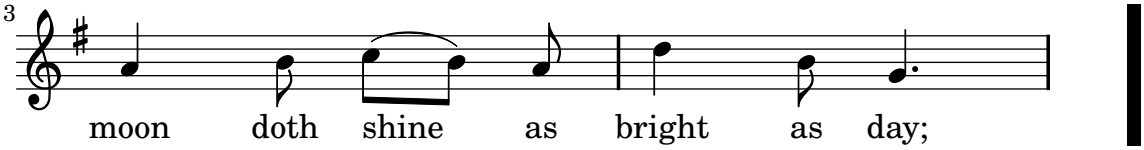




Les paroles sont maintenant correctement alignées, mais les liens de croche automatiques ne conviennent pas pour les notes au-dessus de *shine as*. On peut les corriger en ajoutant des liens de croche manuels ; pour ceci consultez Section 2.1.6 [Barres de ligature automatiques et manuelles], page 26.
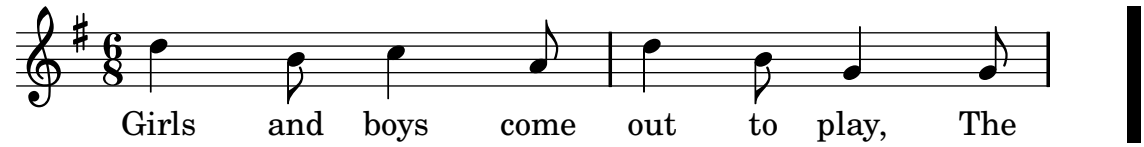
```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c([ b]) a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine as | bright as day; |
  }
>>
```

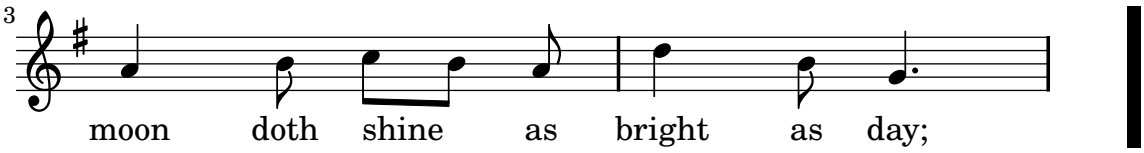




Au lieu d'utiliser une liaison, on peut indiquer le mélisme dans les paroles en insérant un caractère souligné _ pour chaque note du mélisme sauf la première.
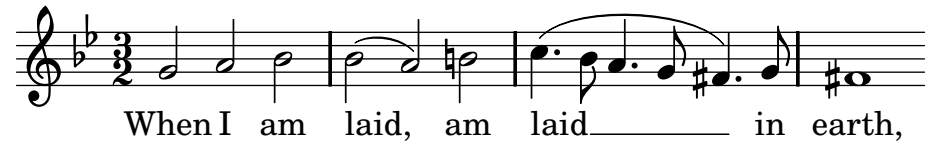
```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c[ b] a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine _ as | bright as day; |
  }
>>
```





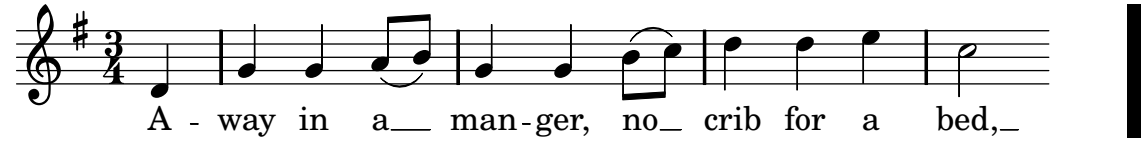
Si une syllabe s'étend sur un grand nombre de notes ou une note très longue, on représente souvent le mélisme par un *trait de prolongation*, qu'on entre avec double caractère souligné __. L'exemple suivant montre les trois premières mesures de la plainte de Didon, extraite de *Didon et Énée* de Purcell.

```
<<
  \relative c'' {
    \key g \minor
    \time 3/2
   g2 a bes | bes2( a) b2 |
    c4.( bes8 a4. g8 fis4.) g8 | fis1
  }
  \addlyrics {
    When I am | laid,
    am | laid __ in | earth,
  }
>>
```



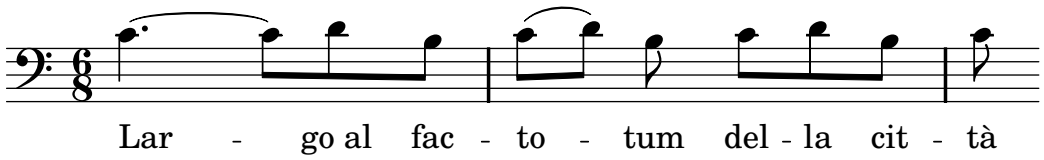
Aucun exemple jusqu'à présent n'a utilisé de mots de plus d'une syllabe. Dans des paroles, de tels mots sont écrits en syllabes séparées par des traits d'union. Avec LilyPond, on utilise deux tirets pour produire un trait d'union centré entre deux syllabes. L'exemple suivant montre tout ce que nous avons vu jusqu'à maintenant sur l'alignement de paroles à une mélodie.

```
<<
  \relative c' {
    \key g \major
    \time 3/4
    \partial 4
    d4 | g4 g a8( b) | g4 g b8( c) |
    d4 d e | c2
  }
  \addlyrics {
    A -- | way in a __ | man -- ger,
    no __ | crib for a | bed, __
  }
>>
```



Avec certaines paroles, en particulier en italien, il se produit la situation inverse : il peut y avoir plusieurs syllabes sur une seule note. On réalise ceci avec LilyPond grâce à un caractère souligné _ sans espace entre les syllabes, ou alors en groupant les syllabes avec des guillemets. L'exemple suivant est extrait de l'air de Figaro *Largo al factotum*, dans *Figaro* de Rossini, où la syllabe *al* est chantée sur la même note que *go*.

```
<<
  \relative c' {
    \clef bass
    \key c \major
    \time 6/8
    c4.~ c8 d b | c8([ d]) b c d b | c8
  }
  \addlyrics {
    Lar -- go_al fac -- | to -- tum del -- la cit -- | tà
  }
>>
```
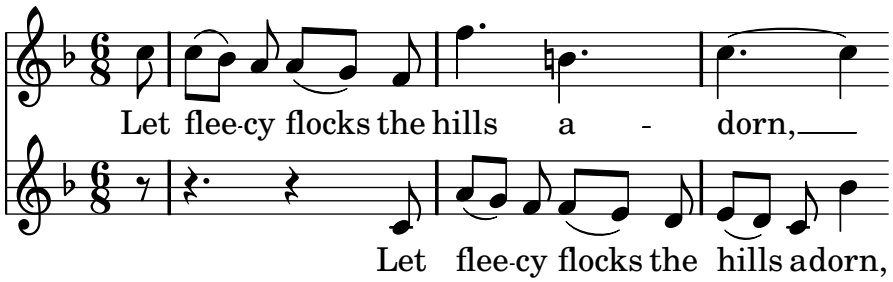


Voir aussi

Manuel de notation : Section “Musique vocale” dans *Manuel de notation*.

2.3.3 Paroles pour plusieurs portées

La méthode simple d’ajout de paroles avec `\addlyrics` peut être également utilisée pour placer des paroles sous plusieurs portées. L’exemple suivant est extrait de *Judas Macchabée* de Händel.

```
<<
  \relative c'' {
    \key f \major
    \time 6/8
    \partial 8
    c8 | c8([ bes]) a a([ g]) f | f'4. b, | c4.~ c4
  }
  \addlyrics {
    Let | flee -- cy flocks the | hills a -- | dorn, __
  }
  \relative c' {
    \key f \major
    \time 6/8
    \partial 8
    r8 | r4. r4 c8 | a'8([ g]) f f([ e]) d | e8([ d]) c bes'4
  }
  \addlyrics {
    Let | flee -- cy flocks the | hills a -- dorn,
  }
>>
```



Pour produire des partitions plus complexes ou plus longues que cet exemple simple, il est vivement conseillé de séparer la structure de la partition des notes et paroles, grâce à des variables. Ceci sera détaillé plus loin dans Section 2.4.1 [Organisation du code source avec des variables], page 37.

Voir aussi

Manuel de notation : Section “Musique vocale” dans *Manuel de notation*.

2.4 Dernières précisions

L'ultime section de ce tutoriel montre comment ajouter une touche finale à des morceaux simples, et constitue une introduction au reste du manuel.

2.4.1 Organisation du code source avec des variables

Lorsque l'on combine tous les éléments étudiés précédemment pour écrire des partitions plus longues, les expressions musicales prennent de l'ampleur et, dans le cas des pièces polyphoniques, deviennent profondément imbriquées, jusqu'au point où il devient difficile de se repérer dans le fichier source. Cet inconvénient peut être résolu par l'utilisation de *variables*.

En utilisant des variables, parfois appelées identificateurs ou macros, on peut scinder des expressions musicales complexes en des expressions plus simples. Une variable se définit comme suit :
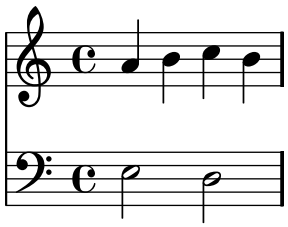
```
musiqueToto = { ... }
```

Le contenu de l'expression musicale `musiqueToto` pourra être utilisé plus loin en faisant précéder son nom d'un anti-slash, c'est-à-dire `\musiqueToto`, tout comme n'importe quelle commande LilyPond. Toute variable doit être définie *avant* son utilisation dans une autre expression musicale.

```
violin = \new Staff {
  \relative c'' {
    a4 b c b
  }
}

cello = \new Staff {
  \relative c {
    \clef bass
    e2 d
  }
}

{
  <<
    \violin
    \cello
  >>
}
```
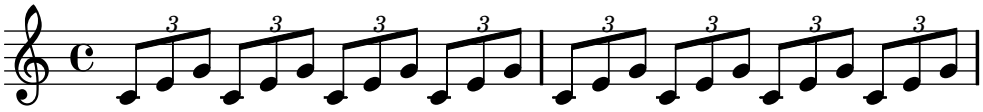


Le nom d'une variable ne doit comporter que des caractères alphabétiques non accentués, aucun nombre ni tiret ne sont autorisés.

On peut utiliser une variable déjà définie autant de fois que l'on veut, y compris dans la définition d'une nouvelle variable ; par exemple, cela peut servir à saisir un motif qu'une seule fois, même s'il se répète un grand nombre de fois dans la pièce.

```
tripletA = \times 2/3 { c,8 e g }
barA = { \tripletA \tripletA \tripletA \tripletA }

\relative c'' {
 \barA \barA
}
```



Il est possible d'utiliser des variables de types variés. Par exemple,

```
largeur = 4.5\cm
nom = "Wendy"
aFivePaper = \paper { paperheight = 21.0 \cm }
```

En fonction de son contenu, un identificateur peut être utilisé à différents endroits. L'exemple suivant utilise les variables définies ci-dessus.

```
\paper {
  \aFivePaper
  line-width = \largeur
}

{
  c4^\nom
}
```

2.4.2 Ajout de titres

On indique les informations bibliographiques – nom du morceau, du compositeur, numéro d'opus... – dans un bloc à part, le bloc d'en-tête `\header`, qui existe indépendamment de l'expression musicale principale. Le bloc `\header` est habituellement placé en début de fichier, après le numéro de version.

```
\version "2.14.2"

\header {
  title = "Symphonie"
  composer = "Moi"
  opus = "Op. 9"
}

{
 ... la musique ...
```
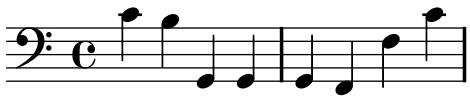
```
}
```

Quand LilyPond traite le fichier, le titre et le compositeur sont imprimés au début de la partition. Vous trouverez plus d'informations sur les titres à la section Section "Création de titres" dans *Manuel de notation*.

2.4.3 Noms de note absolus

Jusqu'ici nous n'avons utilisé que le mode `\relative` pour définir les hauteurs de notes. Si c'est souvent le moyen le plus simple de saisir la musique au clavier, il existe une autre façon de procéder : le mode de hauteurs absolues.

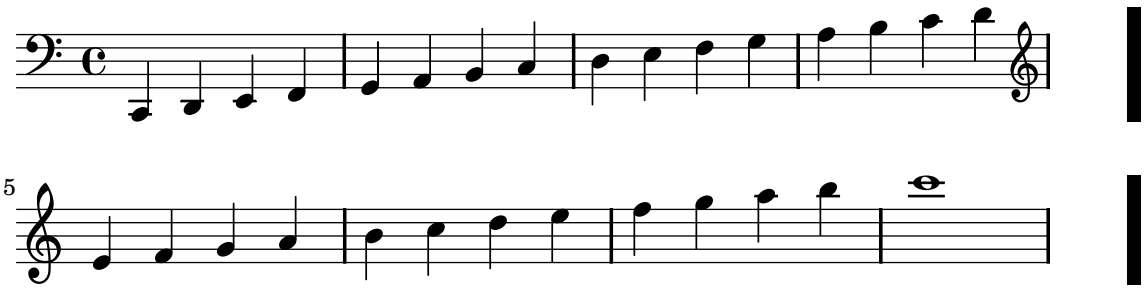
Si vous omettez la commande `\relative`, LilyPond considérera toutes les hauteurs comme des hauteurs absolues. Un `c'` désigne toujours le do central, un `b` se situe une seconde en dessous du do central, et un `g,` est situé sur la première ligne de la portée en clé de fa.

```
{
  \clef "bass"
  c'4 b g, g, |
  g,4 f, f c' |
}
```



Voici une gamme sur 4 octaves :
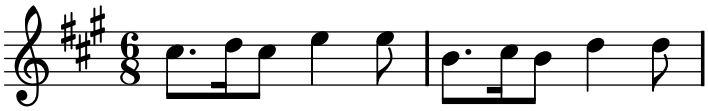
```
{
  \clef "bass"
  c,4 d, e, f, |
  g,4 a, b, c |
  d4 e f g |
  a4 b c' d' |
  \clef "treble"
  e'4 f' g' a' |
  b'4 c'' d'' e'' |
  f''4 g'' a'' b'' |
  c'''1 |
}
```



Comme vous pouvez le voir, il faut beaucoup d'apostrophes pour écrire de la musique dans un registre aigu, comme le montre cet extrait de Mozart.

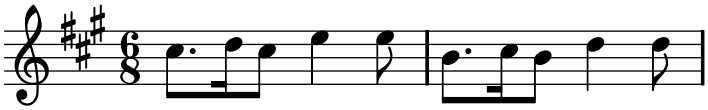
```
{
  \key a \major
  \time 6/8
  cis''8. d''16 cis''8 e''4 e''8 |
```

```
  b'8. cis''16 b'8 d''4 d''8 |
}
```



Toutes ces apostrophes rendent le fichier moins lisible, et surtout il est très probable d'oublier au moins une apostrophe au cours de la frappe. En mode `\relative`, le même exemple devient bien plus facile à lire et à saisir.

```
\relative c'' {
  \key a \major
  \time 6/8
  cis8. d16 cis8 e4 e8 |
  b8. cis16 b8 d4 d8 |
}
```



Si d'aventure vous faites une erreur d'octaviation, le mode `\relative` la mettra en évidence : toutes les notes suivantes seront placées à la mauvaise octave. En mode de hauteurs absolues, une erreur isolée ne serait pas autant visible, et serait donc plus difficile à dénicher.

Cependant, le mode de hauteurs absolues reste utile pour les musiques où les intervalles sont étendus, et plus encore pour les fichiers LilyPond créés par des programmes.

2.4.4 Après le tutoriel

Après avoir parcouru ce tutoriel, vous devriez essayer d'écrire un morceau ou deux. Commencez par copier l'un des modèles types et ajoutez-y des notes – consultez les Annexe A [Modèles], page 146. Si vous voulez employer une notation que vous n'avez pas trouvé dans le tutoriel, consultez le manuel de notation, en commençant par la Section "Notation musicale générale" dans *Manuel de notation*. Si vous désirez écrire pour un ensemble instrumental non couvert par les modèles, lisez la section Section 3.4 [Extension des modèles], page 70.

Après avoir écrit quelques pièces courtes, lisez les chapitres 3 à 5 du manuel d'initiation. Rien ne s'oppose à ce que vous consultiez dès à présent ces chapitres, bien sûr ! Néanmoins, le reste du manuel d'initiation part du principe que vous avez déjà bien assimilé la syntaxe de base de LilyPond. Vous pouvez toujours survoler ces chapitres 3 à 5, et y revenir plus tard après avoir acquis de l'expérience.

Dans ce tutoriel comme dans le reste de ce manuel, se trouve à chaque section un paragraphe **Voir aussi** contenant des références vers d'autres sections : il est conseillé de ne pas les suivre en première lecture ; lorsque vous aurez lu l'ensemble du manuel d'initiation, vous pourrez en relisant certaines sections suivre ces références pour approfondir certains aspects.

Si vous ne l'avez pas encore fait, lisez Section 1.4.3 [Vue d'ensemble des manuels], page 20. Les sources de documentation et d'information sur LilyPond sont vastes, il est normal pour un débutant de ne pas savoir où chercher ; si vous passez quelques minutes à lire attentivement cette section, vous vous épargnerez certainement la frustration causée par des heures de recherches infructueuses.

3 Concepts fondamentaux

Le tutoriel nous a montré comment obtenir une édition de toute beauté à partir d'un simple fichier texte. Nous nous intéresserons dans cette partie aux concepts et techniques qui permettent d'obtenir des partitions complexes de même qualité.

3.1 Organisation des fichiers LilyPond

La mise en forme des fichiers d'entrée de LilyPond est vraiment peu astreignante, afin d'offrir assez de souplesse aux utilisateurs expérimentés pour qu'ils puissent organiser leurs fichiers comme ils l'entendent. Cependant, les nouveaux utilisateurs peuvent parfois se perdre en raison de cette souplesse. Cette section présente sommairement l'organisation du code LilyPond, en privilégiant la simplicité au détriment de certains détails. Vous trouverez une description plus complète dans Section "Structure de fichier" dans *Manuel de notation*.

3.1.1 Introduction à la structure de fichier LilyPond

Un fichier d'entrée LilyPond ressemble à :

```
\version "2.14.2"
\header { }
\score {
  ...expression musicale composite...  % c'est là qu'est la musique !
  \layout { }
  \midi { }
}
```

Il existe de nombreuses variantes à ce schéma simpliste, mais cet exemple est un préambule à notre propos.

Jusqu'à présent, les exemples que nous avons pu voir ne faisaient pas appel à la commande `\score{}`. En fait, LilyPond ajoute automatiquement les commandes nécessaires au traitement d'un code simpliste. LilyPond considère

```
\relative c'' {
  c4 a d c
}
```

comme un raccourci de

```
\book {
  \score {
    \new Staff {
      \new Voice {
        \relative c'' {
          c4 a b c
        }
      }
    }
    \layout { }
  }
}
```

En d'autres termes, si le code n'est constitué que d'une expression musicale simple, LilyPond interprètera le fichier tout comme si cette expression était incluse dans les commandes de notre premier exemple.

Attention : de nombreux exemples, dans la documentation de LilyPond, ne font pas apparaître les commandes `\new Staff` ou `\new Voice`, qui sont créées implicitement. Ce qui n'est pas

primordial pour des exemples simples le devient dès que la situation se complexifie un tant soit peu. Le fait de ne pas déclarer explicitement un contexte peut alors amener à des résultats quelque peu surprenants, comme la création d'une portée supplémentaire et indésirable. La manière de créer explicitement des contextes est traitée plus en détails au chapitre Section 3.3 [Contextes et graveurs], page 59.

> **Note :** Dès lors que votre musique dépasse quelques lignes, nous vous engageons fortement à créer explicitement les voix et portées.

Mais revenons à notre premier exemple, et penchons-nous tout d'abord sur la commande `\score`.

Un bloc `\score` doit contenir une et une seule expression musicale, exprimée immédiatement à la suite de la commande `\score`. Rappelez-vous que cette expression peut être n'importe quoi, d'une note isolée à un gigantesque

```
{
  \new StaffGroup <<
    ...collez ici la partition complète d'un opéra de Wagner...
  >>
}
```

Dès lors que tout cela est entre accolades : `{ ... }`, LilyPond le considère comme une et une seule expression musicale.

Comme nous l'avons vu précédemment, un bloc `\score` peut contenir d'autres informations :

```
\score {
  { c'4 a b c' }
  \header { }
  \layout { }
  \midi { }
}
```

Gardez à l'esprit que ces trois commandes – `\header`, `\layout` et `\midi` – sont spécifiques : à l'inverse de toutes les commandes débutant par une oblique inversée `\` (*backslash* en anglais), **elles ne constituent pas** des expressions musicales et ne peuvent pas faire partie d'expressions musicales. Elles peuvent de ce fait être placées à l'intérieur du bloc `\score`, ou bien à l'extérieur. En réalité, ces commandes sont la plupart du temps indépendantes du bloc `\score` – par exemple, la commande `\header` intervient souvent avant le bloc `\score`, comme le montre l'exemple ci-dessus.

Les deux autres commandes – `\layout {}` et `\midi {}` – que nous n'avons pas détaillées pour l'instant, auront respectivement pour effet, lorsqu'elles interviennent, de produire une sortie imprimable et un fichier MIDI. Nous nous y intéressons plus particulièrement dans le manuel de notation, aux chapitres Section "Mise en forme de la partition" dans *Manuel de notation* et Section "Création de fichiers MIDI" dans *Manuel de notation*.

Vous pouvez tout à fait mentionner plusieurs blocs `\score`. Ils seront traités comme autant de partitions indépendantes qui seront regroupées dans un seul fichier résultant. La commande `\book` (*recueil* ou *ouvrage*) n'est pas obligatoire – elle sera créée implicitement. Néanmoins, le recours à la commande `\book` vous permettra d'obtenir des fichiers résultants distincts à partir d'un même fichier source '`.ly`' – par exemple un fichier par pupitre.

En résumé :

Dès que LilyPond rencontre un bloc `\book`, il crée un fichier distinct ('`.pdf`' par exemple). Dans le cas où il n'est pas mentionné explicitement, LilyPond regroupera l'intégralité du code dans un bloc `\book`.

Tout bloc `\score` inclus dans un bloc `\book` constitue un fragment de musique, par exemple un mouvement d'une symphonie.

Tout bloc `\layout` affecte le bloc `\score` ou `\book` au sein duquel il intervient : si c'est à l'intérieur d'un bloc `\score`, seul celui-ci en sera affecté. Dans le cas où le bloc `\layout` se trouve à l'extérieur du bloc `\score`, que le bloc `\book` soit explicite ou non, il affectera chacun des `\score` compris dans ce `\book`.

Pour plus de détail à ce sujet, consultez Section "Plusieurs partitions dans un même ouvrage" dans *Manuel de notation*.

Un autre raccourci pratique est la possibilité de définir des variables, également appelées « identificateurs » – voir Section 2.4.1 [Organisation du code source avec des variables], page 37 à ce sujet. Dans tous les modèles, vous trouverez :

```
melodie = \relative c' {
  c4 a b c
}

\score {
  { \melodie }
}
```

Lorsque LilyPond examinera ce fichier, il va prendre la valeur de la variable `melodie`, c'est-à-dire tout ce qui suit le signe `=`, et l'insérer partout où il rencontrera `\melodie`. Vous êtes libre de choisir comment dénommer vos variables[1] ; ce peut être `melodie`, `global`, `maindroitepiano`, ou `laTeteAToto`, tant qu'il ne s'agit pas de « mot réservé ». Pour plus de détails, voir Section 3.4.4 [Économie de saisie grâce aux identificateurs et fonctions], page 84.

Voir aussi

Pour une description complète du format des fichiers d'entrée, voir Section "Structure de fichier" dans *Manuel de notation*.

3.1.2 La partition est une (unique) expression musicale composée

Dans la section précédente, Section 3.1.1 [Introduction à la structure de fichier LilyPond], page 41, nous avons vu l'organisation générale des fichiers d'entrée de LilyPond. Mais c'est comme si nous avions éludé la question essentielle : comment diable peut-on savoir quoi mettre après `\score` ?

En fait, nous ne l'avons pas éludée du tout : le grand mystère est tout simplement qu'**il n'y a pas** de mystère. Allez, expliquons-le en une ligne :

> *Un bloc* `\score` *doit commencer par une et une seule expression musicale.*

Peut-être serait-il judicieux de relire la section Section 2.2.1 [Les expressions musicales en clair], page 27, dans laquelle vous avez appris à construire de grandes expressions musicales petit bout par petit bout – nous avons vu les notes, puis les accords, etc. Maintenant, nous allons partir d'une grande expression musicale, et remonter la pente. Pour rester simple, nous nous contenterons d'un chanteur accompagné au piano. On n'a pas besoin d'une partition d'orchestre – c.-à-d. des portées regroupées en `StaffGroup` – donc laissons cela de côté. Par contre, nous voulons bien une voix et une double portée de piano.

```
\score {
  {
    <<
      \new Staff = "chanteur" <<
```

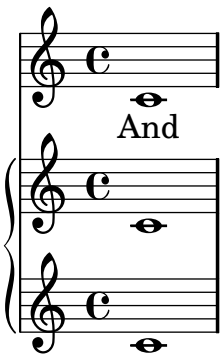
[1] Les noms de variables sont sensibles à la casse, et ne peuvent contenir ni chiffre, ni ponctuation, ni caractère accentué, ni espace.

```
        >>
        \new PianoStaff = "piano" <<
        >>
      >>
    }
    \layout { }
}
```

Nous avons ici attribué des noms aux portées – « chanteur » et « piano ». Bien que cela ne soit pas primordial, c'est une habitude qu'il est bon d'adopter dès le départ : vous saurez au premier coup d'œil à quoi correspond chaque portée.

Vous vous souvenez que nous avons recours à `<<` et `>>` en lieu et place de `{ ... }` pour gérer des musiques simultanées. Et, pour le coup, on aimerait *vraiment* que la partie vocale et l'accompagnement soient imprimés ensemble... Bien que faire appel à `<< ... >>` ne soit pas réellement nécessaire pour la portée du chanteur, dans la mesure où elle ne contient qu'une seule expression musicale, nous vous recommandons de prendre l'habitude de l'encadrer ainsi plutôt que par de simples accolades – une portée peut en effet contenir plusieurs voix, ou bien des notes **et** des paroles. Dans la mesure où nous y ajouterons des paroles, les chevrons sont donc obligatoires. Si vous avez oublié comment ajouter des paroles à l'aide de la commande `\addlyrics`, relisez le chapitre Section 2.3.1 [Écriture de chants simples], page 32.

```
\score {
  <<
    \new Staff = "singer" <<
      \new Voice = "vocal" { c'1 }
      \addlyrics { And }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "upper" { c'1 }
      \new Staff = "lower" { c'1 }
    >>
  >>
  \layout { }
}
```



On y voit nettement plus clair maintenant. Nous voici donc avec la partie du chanteur, qui contient un ensemble `Voice`, ce qui dans LilyPond correspond à une voix, au sens de voix d'une polyphonie plutôt que de voix chantée – ce pourrait être une partie de violon par exemple –, et des paroles. Nous avons également une partie de piano, qui contient deux portées : une pour la main droite, une autre pour la main gauche. Il nous faudra d'ailleurs ajouter une clef de fa à cette dernière.

À ce point, on pourrait commencer à ajouter les notes. Dans les accolades qui suivent `\new Voice = "chant"`, on pourrait commencer à écrire
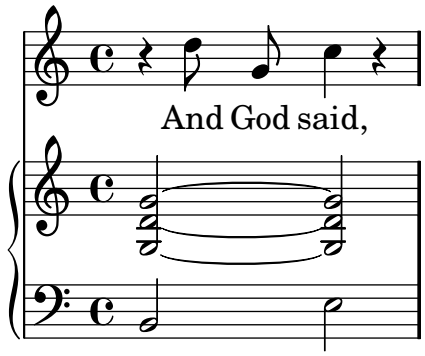
```
\relative c'' {
  r4 d8\noBeam g, c4 r
}
```

Mais si l'on procédait ainsi, la section `\score` deviendrait vite assez touffue, et très rapidement on ne s'y retrouverait plus. C'est pourquoi on utilisera plutôt des variables, ou identificateurs, comme nous l'avons vu plus haut. Pour s'assurer que le contenu de la variable `texte` soit bien interprété comme des paroles, nous le préfixons d'un `\lyricmode`. Sans cette précaution, LilyPond tenterait d'interpréter le contenu de cette variable comme des notes, ce qui déclencherait immanquablement des erreurs. LilyPond dispose de différents types de données – voir Section "Modes de saisie" dans *Manuel de notation* pour plus de détails.

Avec quelques notes de plus et une clef de fa, nous pourrions avoir :

```
melody = \relative c'' { r4 d8\noBeam g, c4 r }
text   = \lyricmode { And God said, }
upper  = \relative c'' { <g d g,>2~ <g d g,> }
lower  = \relative c { b2 e2 }

\score {
  <<
    \new Staff = "singer" <<
      \new Voice = "vocal" { \melody }
      \addlyrics { \text }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "upper" { \upper }
      \new Staff = "lower" {
        \clef "bass"
        \lower
      }
    >>
  >>
  \layout { }
}
```



Quand on écrit ou que l'on lit une section `\score`, mieux vaut y aller lentement et soigneusement. Commencez par le niveau le plus large, puis travaillez sur chaque niveau plus détaillé. À ce propos, une indentation stricte et propre est vraiment d'une aide précieuse : assurez-vous que chaque élément d'un même niveau a le même décalage horizontal dans votre éditeur de texte !
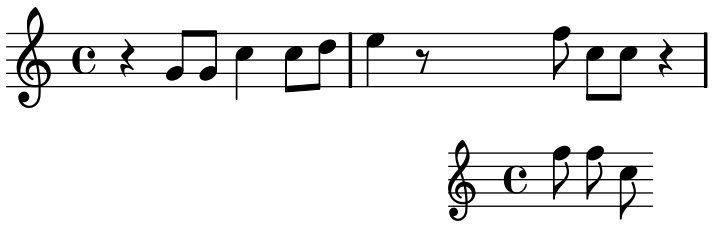
Voir aussi

Manuel de notation : Section "Structure d'une partition" dans *Manuel de notation*.

3.1.3 Expressions musicales imbriquées

Déclarer toutes les portées dès le départ n'est pas une obligation ; elles peuvent intervenir temporairement n'importe où dans la partition. Ceci est tout à fait indiqué pour créer des sections Section "ossia" dans *Glossaire*. L'exemple suivant illustre la manière de créer temporairement une nouvelle portée, l'espace de trois notes :
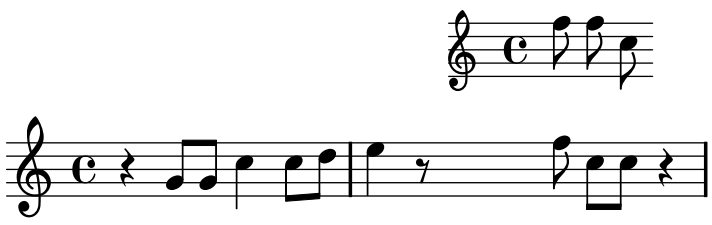
```
\new Staff {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f c c }
      \new Staff {
        f8 f c
      }
    >>
    r4 |
  }
}
```



Vous noterez la taille de la clef, identique à celle que l'on trouve lors d'un changement en cours de ligne – légèrement plus petite que celle imprimée en tête de ligne.

Une section ossia se placera au dessus de la portée en procédant ainsi :

```
\new Staff = "main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f c c }
      \new Staff \with {
        alignAboveContext = #"main" }
      { f8 f c }
    >>
    r4 |
  }
}
```



Cet exemple recourt à `\with`, que nous verrons en détail plus avant. C'est un moyen de modifier le comportement par défaut d'une portée individuelle. Nous indiquons ici que la nouvelle

portée doit se placer au-dessus de la portée « principal » plutôt qu'en dessous, ce qui est le comportement par défaut.

Voir aussi

Les ossia apparaissent souvent sans clef ni métrique, et dans une taille plus petite. Ceci requiert des commandes dont nous n'avons pas encore parlé. Voir Section 4.3.2 [Taille des objets], page 104 et Section "Portées d'ossia" dans *Manuel de notation*.

3.1.4 Non-imbrication des crochets et liaisons

Nous avons déjà rencontré plusieurs types de crochets au fil de nos fichiers LilyPond. Ils obéissent à des règles différentes qui peuvent paraître déroutantes de prime abord. Avant d'examiner ces règles, voici une liste des différents types de crochet :
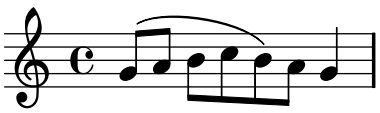
Type de crochet	**Fonction**
`{ .. }`	Délimite un segment de musique séquentielle
`< .. >`	Délimite les notes d'un accord
`<< .. >>`	Délimite des sections simultanées
`( .. )`	Marque le début et la fin d'une liaison
`\( .. \)`	Marque le début et la fin d'une liaison de phrasé
`[ .. ]`	Marque le début et la fin d'une ligature manuelle

D'autres constructions permettent d'obtenir des lignes regroupant ou en travers des notes : les liaisons de prolongation indiquées par un tilde (`~`), les marques de nolet avec `\times x/y {..}`, ou encore les notes d'ornement avec `\grace{..}`.

En dehors de LilyPond, l'imbrication correcte de différents types de crochets exige un strict respect des conventions, telles que `<< [ { ( .. ) } ] >>`, où les marques de fermeture interviennent obligatoirement dans l'ordre exactement inverse à celles d'ouverture. Ceci **doit** être rigoureusement respecté pour les trois types de crochets utilisés pour **délimiter** comme l'indique le tableau ci-dessus. Une telle rigueur dans l'imbrication n'est **pas** requise pour les types de crochets dont la fonction est de **marquer**, selon le tableau ci-dessus, lorsqu'il sont utilisés en combinaison avec des liaisons de prolongation ou des nolets. En effet, il ne s'agit pas de crochets ayant pour fonction de borner quelque chose ; ils agissent plutôt comme marquant le début de quelque chose et sa fin.

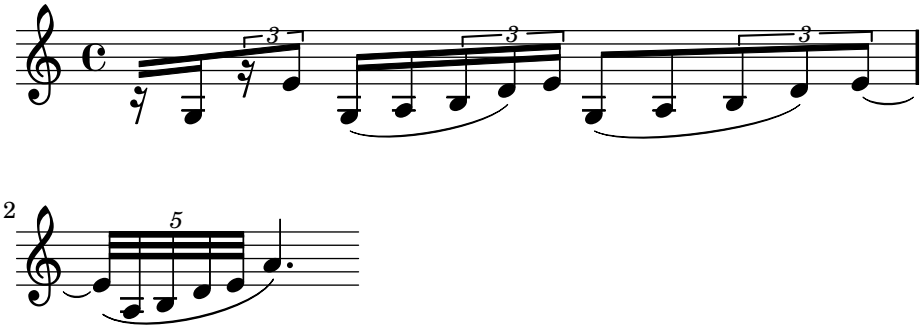
Ainsi, et bien que ce ne soit pas très musical, une liaison de phrasé peut débuter avant l'insertion d'une ligature manuelle et s'arrêter avant la fin de la ligature :

```
g8\( a b[ c b\) a] g4
```



De manière générale, différents types de crochets, notamment s'ils indiquent des nolets, liaisons de prolongation ou notes d'ornement, peuvent se mélanger entre eux. L'exemple suivant montre une ligature qui se prolonge sur un triolet (ligne 1), puis une liaison qui se prolonge sur un triolet (ligne 2) et enfin une ligature et une liaison qui s'étendent sur un triolet, lui-même lié à un quintolet agrémenté d'une liaison de phrasé se poursuivant (lignes 3 et 4).

```
r16[ g \times 2/3 { r16 e'8] }
g,16( a \times 2/3 { b16 d) e }
g,8[( a \times 2/3 { b8 d) e~] } |
\times 4/5 { e32\( a, b d e } a4.\)
```



3.2 Les voix contiennent la musique

Les chanteurs utilisent leur voix pour chanter ; il en va de même pour LilyPond. En fait, la musique de chacun des instruments d'une partition est contenue dans des voix (*Voices* en anglais), qui se trouvent être le concept fondamental de LilyPond.
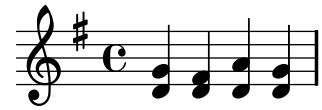
3.2.1 J'entends des Voix

Dans une partition gérée par LilyPond, le niveau le plus bas, ou bien élémentaire ou fondamental, est le « contexte de voix » – *Voice context* en anglais –. Pour d'autres logiciels, on fait tantôt référence à la notion de « couche » ou de « calque ».

En réalité, le contexte de voix est le seul à pouvoir contenir de la musique. S'il n'est pas déclaré explicitement, il sera créé automatiquement comme nous l'avons vu au début de ce chapitre. Certains instruments, le hautbois par exemple, ne peuvent jouer qu'une seule note à la fois. On dit en pareil cas qu'il s'agit de musique monophonique, et nous n'aurons alors besoin que d'une seule voix. Les intruments qui, comme le piano, peuvent émettre plusieurs sons en même temps, nécessitent de recourir à plusieurs voix pour gérer efficacement l'alignement des notes et rythmes différents.

Si une voix unique peut tout à fait contenir plusieurs notes dans un accord, à partir de quand aurons-nous vraiment besoin de plusieurs voix ? Considérons déjà ces quatre accords :

```
\key g \major
<d g>4 <d fis> <d a'> <d g>
```
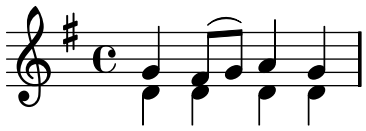


Nous exprimons ici chacun des accords par l'utilisation de chevrons gauche et droite simples, `< ... >`, puisque nous n'avons besoin que d'une seule voix. Supposons maintenant que le fa dièse soit une croche, suivie d'un sol croche – une note de passage vers le la ? Nous avons alors deux notes qui débutent au même moment, mais dont la durée est différente : un ré noire et un fa dièse croche. Comment coder cela ? Dans la mesure où toutes les notes d'un accord doivent avoir la même durée, nous ne pouvons pas écrire un accord. Nous ne pouvons pas non plus écrire deux notes séparées, puisqu'elles débutent en même temps. Nous avons alors besoin de deux voix.

Voyons comment cela se pratique selon la grammaire de LilyPond.

Le plus sûr moyen de saisir un fragment où plusieurs voix cohabitent sur la même portée, consiste à saisir chacune des voix séquentiellement (avec `{...}`), puis à les combiner en simultané à l'aide de doubles chevrons gauche/droite, `<<...>>`. Les fragments devront être séparés par une double oblique inversée, `\\`, pour les affecter à des voix séparées. Dans le cas contraire, les notes seraient toutes affectées à une même voix, ce qui pourait générer des erreurs. Cette manière de procéder est tout à fait indiquée dans le cas d'une pièce ne comportant que quelques courts passages de polyphonie.

Voici comment éclater les accords en deux voix, avec la note de passage et la liaison :
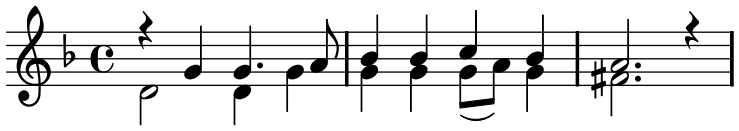
```
\key g \major
%    Voice "1"                Voice "2"
<< { g4 fis8( g) a4 g } \\ { d4 d d d }  >>
```



Notez que les hampes de la seconde voix sont dirigées vers le bas.
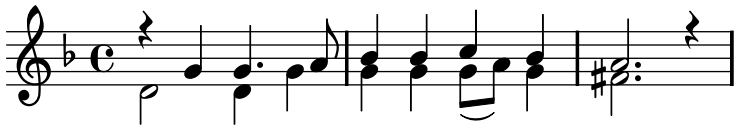
Autre exemple :

```
\key d \minor
%    Voice "1"             Voice "2"
<< { r4 g g4. a8 }    \\ { d,2 d4 g }       >> |
<< { bes4 bes c bes } \\ { g4 g g8( a) g4 } >> |
<< { a2. r4 }         \\ { fis2. s4 }       >> |
```



Le recours à une construction `<< \\ >>` particulière à chaque mesure n'est pas nécessaire. Bien qu'on y gagne en lisibilité si chaque mesure ne contient que quelques notes, il est plus judicieux de carrément séparer chaque voix :

```
\key d \minor
<< {
  % Voice "1"
  r4 g g4. a8 |
  bes4 bes c bes |
  a2. r4 |
} \\ {
  % Voice "2"
  d,2 d4 g |
  g4 g g8( a) g4 |
  fis2. s4 |
} >>
```

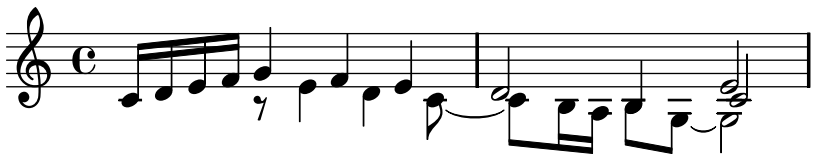


Cet exemple ne comporte que deux voix, mais il peut être étendu pour traiter trois voix ou plus en ajoutant autant de séparateurs `\\` que de besoin.

Les contextes `Voice` portent les noms `"1"`, `"2"`, etc. Pour chacun de ces contextes, le positionnement et l'orientation des liaisons, hampes, nuances, etc. est définie automatiquement.

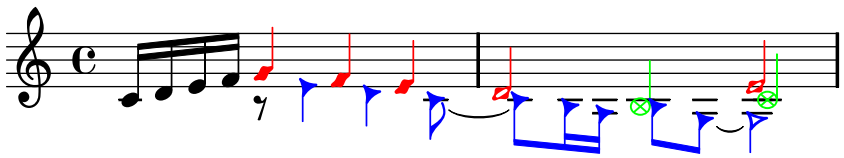
```
\new Staff \relative c' {
  % Main voice
  c16 d e f
  %    Voice "1"     Voice "2"                Voice "3"
  << { g4 f e } \\ { r8 e4 d c8~ } >> |
```

```
  << { d2 e }   \\ { c8 b16 a b8 g~ g2 } \\ { s4 b c2 } >> |
}
```



Ces voix sont séparées de la voix principale, laquelle contient les notes en dehors de la construction `<<...>>` – que nous appellerons *construction simultanée*. Les liaisons, de prolongation ou non, ne peuvent relier des notes que si elles appartiennent à la même voix ; elles ne peuvent ni pénétrer une contruction simultanée, ni en sortir. Inversement, les voix parallèles issues de constructions simultanées apparaissant sur une même portée appartiennent à la même voix. Les autres propriétés liées au contexte de voix s'appliquent tout au long des constructions simultanées. Reprenons notre exemple, en affectant une couleur et une allure différentes aux notes de chacune des voix. Vous noterez qu'un changement apporté à une voix ne se propage pas aux autres, et qu'il se reporte jusqu'au bout, et que la voix aux triangles bleus comporte une liaison de prolongation entre deux constructions.

```
\new Staff \relative c' {
  % Main voice
  c16 d e f
  <<  % Bar 1
    {
      \voiceOneStyle
      g4 f e
    }
  \\
    {
      \voiceTwoStyle
      r8 e4 d c8~
    }
  >> |
  <<  % Bar 2
     % Voice 1 continues
    { d2 e }
  \\
     % Voice 2 continues
    { c8 b16 a b8 g~ g2 }
  \\
    {
      \voiceThreeStyle
      s4 b c2
    }
  >> |
}
```



Les commandes `\voiceXXXStyle` sont principalement dédiées à une utilisation pédagogique, comme l'est ce document. Elles modifient la couleur des hampes et ligatures et le style de tête des note, pour permettre une meilleure distinction entre les différentes voix. La première voix comporte des têtes en losange rouge, la deuxième en triangle bleu, la troisième en cercle barré vert, la quatrième (non utilisée ici) en croix magenta ; `\voiceNeutralStyle` (non utilisé ici) revient au style par défaut. Nous verrons plus tard comment créer de telles commandes. Voir Section 4.3.1 [Visibilité et couleur des objets], page 99 et Section 4.6.2 [Utilisation de variables dans les retouches], page 137.

La polyphonie ne modifie en rien la relation entre les notes au sein d'un bloc `\relative`. Chaque note est calculée par rapport à celle qui la précède, ou bien par rapport à la première note de l'accord qui précède. Ainsi, dans
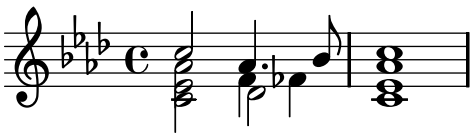
```
\relative c' { noteA << < noteB noteC > \\ noteD >> noteE }
```

`noteB` est relative à `noteA`
`noteC` est relative à `noteB`, pas à `noteA`
`noteD` est relative à `noteB`, pas à `noteA` ni `noteC`
`noteE` est relative à `noteD`, pas à `noteA`

Une méthode alternative, et qui peut simplifier les choses si les notes des différentes voix sont espacées, consiste à placer une commande `\relative` au début de chacune des voix :

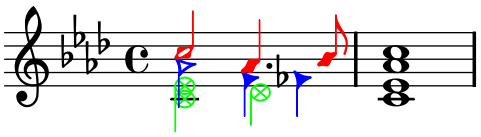
```
\relative c' { noteA ... }
<<
  \relative c'' { < noteB noteC > ... }
\\
  \relative g' { noteD ... }
>>
\relative c' { noteE ... }
```

Pour finir, analysons le principe d'utilisation des voix dans une pièce complexe. Nous allons nous concentrer sur les deux premières mesures du second des Deux nocturnes, opus 32 de Chopin. Cet exemple nous servira à plusieurs reprises, y compris dans le chapitre suivant, pour illustrer certaines techniques de notation. Aussi, ne prétez pas trop d'attention à ce qui pour l'instant pourrait vous paraître vraiment mystérieux dans le code, et intéressons-nous uniquement à ce qui concerne la musique et les voix – ce qui est plus compliqué sera décortiqué plus tard.
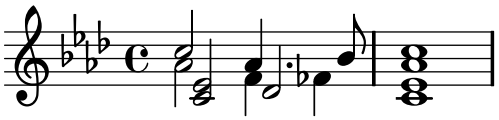


La direction des hampes sert souvent à indiquer dans la continuité deux lignes mélodiques simultanées. Ici, les hampes des notes les plus hautes vont vers le haut, et celles des notes plus basses vers le bas. C'est une première indication de ce que nous avons eu recours à plus d'une voix.

Mais le réel besoin de multiples voix se fait sentir dès lors que plusieurs notes qui débutent en même temps ont des durées différentes. C'est évident au troisième temps de la première mesure : le la bémol est une noire pointée, le fa une noire, et le ré bémol une blanche. On ne peut les grouper dans un accord, puisque toutes les notes composant un accord doivent être de même durée. On ne peut non plus les écrire séquentiellement, puisqu'elles débutent toutes au même instant. Ce fragment de mesure nécessite trois voix, et une bonne pratique voudrait que l'intégralité de la mesure soit sur trois voix, comme ci-dessous où nous avons une allure et une couleur différentes aux notes de chacune d'entre elles. Une fois de plus, nous reviendrons plus tard sur le code que vous ne comprendriez pas.
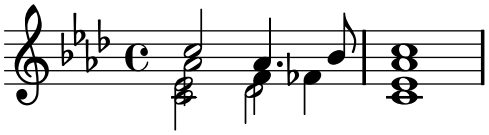


Essayons à présent de coder cette musique en partant de zéro. Comme nous le verrons, certaines difficultés vont se présenter. Partons de ce que nous avons appris : utilisons la construction `<< \\ >>` pour saisir la première mesure dans trois voix :

```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 } \\ { aes2 f4 fes } \\ { <ees c>2 des2 }
  >>
  <c ees aes c>1
}
```



La direction des hampes est attribuée automatiquement : les voix impaires portent des hampes vers le haut, les voix paires des hampes vers le bas. Les hampes des voix 1 et 2 sont orientées comme il faut mais celles de la voix 3 devraient, dans le cas qui nous occupe, aller vers le bas. Nous pouvons corriger cela en sautant la voix 3 et en plaçant la musique dans la voix 4 grâce à un `\\` supplémentaire :

```
\new Staff \relative c'' {
  \key aes \major
  << % Voice one
    { c2 aes4. bes8 }
  \\ % Voice two
    { aes2 f4 fes   }
  \\ % Omit Voice three
  \\ % Voice four
    { <ees c>2 des2 }
  >> |
  <c ees aes c>1 |
}
```



Cette manipulation nous permet de régler la direction des hampes, mais elle engendre un problème que l'on rencontre parfois avec de multiples voix, à savoir que les hampes d'une voix peuvent chevaucher les têtes de note des autres voix. En matière de mise en forme des notes, LilyPond tolère que des notes ou accords appartenant à deux voix se retrouvent dans le même empilement de notes (*note column* en anglais) si tant est que ces hampes vont dans des directions opposées ; néanmoins les notes des troisième et quatrième voix seront décalées si nécessaire pour éviter que les têtes ne se chevauchent. Cela marche plutôt bien mais, dans notre exemple, les notes de la voix la plus basse ne sont vraiment pas correctement placées. LilyPond met à notre disposition plusieurs moyens d'ajuster le positionnement horizontal des notes. Nous ne sommes pas encore tout à fait prêts pour voir comment corriger cela, aussi nous examinerons ce problème dans un autre chapitre (voir la propriété `force-hshift` dans Section 4.5.2 [Correction des collisions d'objets], page 121).

Voir aussi

Manuel de notation : Section “Plusieurs voix” dans *Manuel de notation*.

3.2.2 Instanciation explicite des voix

Les contextes Section “Voice” dans *Référence des propriétés internes* peuvent être déclarés manuellement dans un bloc `<< >>` pour créer de la musique polyphonique, en utilisant `\voiceOne`, ... jusqu'à `\voiceFour` pour assigner des directions de hampes et un déplacement horizontal pour chaque partie. Cette méthode apporte de la clarté pour des partitions plus importantes puisqu'elle permet de bien séparer les voix et de leur affecter un nom plus parlant.
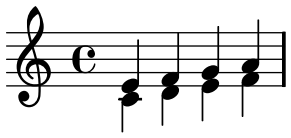
En particulier, la construction `<< \\ >>` que nous avons vue précédemment :

```
\new Staff {
  \relative c' {
    << { e4 f g a } \\ { c,4 d e f } >>
  }
}
```

équivaut à

```
\new Staff <<
  \new Voice = "1" { \voiceOne \relative c' { e4 f g a } }
  \new Voice = "2" { \voiceTwo \relative c' { c4 d e f } }
>>
```
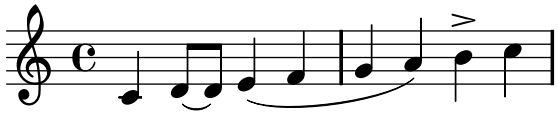
Toutes deux produiront
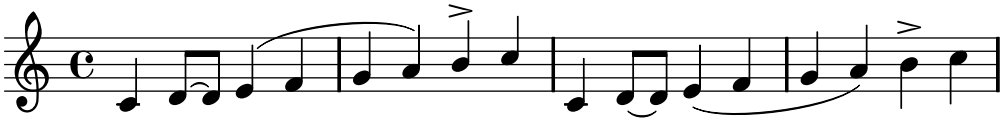


Les commandes `\voiceXXX` fixent la direction des hampes, des liaisons de prolongation et de phrasé, des articulations, des annotations, des points d'augmentation des notes pointées et des doigtés. `\voiceOne` et `\voiceThree` font pointer ces objets vers le haut, alors que `\voiceTwo` et `\voiceFour` les font pointer vers le bas. Ces commandes génèrent par ailleurs un décalage horizontal de chacune des voix pour éviter tout risque de chevauchement entre plusieurs notes. La commande `\oneVoice` les ramène aux critères normaux.

Voyons, à l'aide de ces exemples simples, les effets respectifs de `\oneVoice`, `\voiceOne` et `\voiceTwo` sur les annotations, liaisons de prolongation ou de phrasé, et sur les nuances.
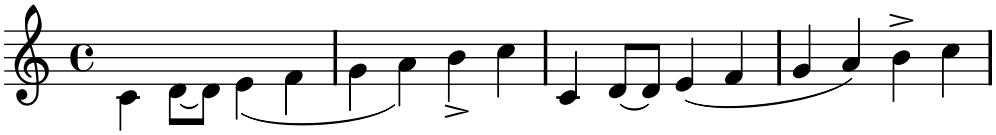
```
\relative c' {
  % Default behavior or behavior after \oneVoice
  c4 d8~ d e4( f | g4 a) b-> c |
}
```



```
\relative c' {
  \voiceOne
  c4 d8~ d e4( f | g4 a) b-> c |
  \oneVoice
  c,4 d8~ d e4( f | g4 a) b-> c |
}
```



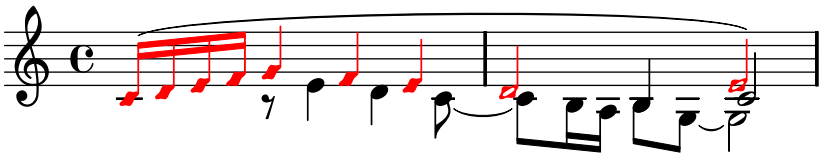
```
\relative c' {
  \voiceTwo
  c4 d8~ d e4( f | g4 a) b-> c |
  \oneVoice
  c,4 d8~ d e4( f | g4 a) b-> c |
}
```



Voyons à présent trois différentes façons d'exprimer un passage polyhonique, à partir d'un exemple de la section précédente. Chacune d'elles aura ses avantages selon les circonstances.

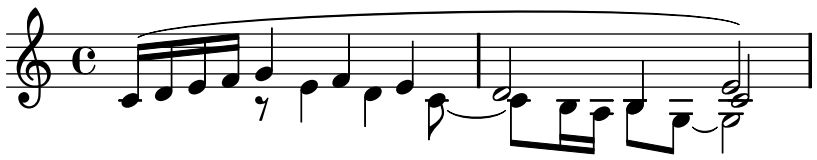
Une expression séquentielle qui apparaît en premier dans un `<< >>` – attention, **pas** dans une construction `<< \\ >>` – appartient à la voix principale. Ceci est utile lorsque des voix supplémentaires apparaissent pendant que la voix principale est jouée. Voici une meilleure réalisation de notre exemple. Les notes colorées et en croix mettent en évidence le fait que la mélodie principale est maintenant dans un seul contexte de voix, ce qui permet d'ajouter une liaison de phrasé à l'ensemble.

```
\new Staff \relative c' {
  \voiceOneStyle
  % The following notes are monophonic
  c16^( d e f
  % Start simultaneous section of three voices
  <<
    % Continue the main voice in parallel
    { g4 f e | d2 e) | }
    % Initiate second voice
    \new Voice {
      % Set stems, etc., down
      \voiceTwo
      r8 e4 d c8~ | c8 b16 a b8 g~ g2 |
    }
    % Initiate third voice
    \new Voice {
      % Set stems, etc, up
      \voiceThree
      s2. | s4 b c2 |
    }
  >>
}
```



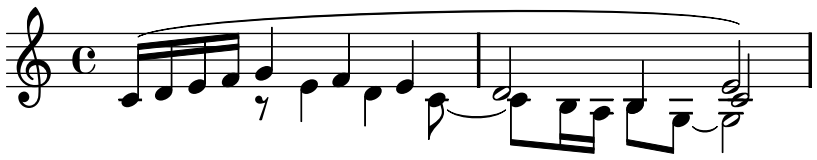
Dans certaines circonstances de polyphonie complexe, vous pourrez être amené à recourir à une voix temporaire, ce qui peut être une manière plus naturelle de saisir la musique :

```
\new Staff \relative c' {
  c16^( d e f
  <<
    { g4 f e | d2 e) | }
    \new Voice {
      \voiceTwo
      r8 e4 d c8~ |
      <<
        { c8 b16 a b8 g~ g2 | }
        \new Voice {
          \voiceThree
          s4 b c2 |
        }
      >>
    }
  >>
}
```



Cette manière de brièvement imbriquer des voix est bien utile pour de courts fragments de musique polyphonique. Mais lorsqu'une portée est très souvent polyphonique, on peut y gagner en clarté si l'on utilise plusieurs voix sur l'ensemble de cette portée et que l'on positionne des silences invisibles pour sauter les moments où il n'y a rien dans cette voix.

```
\new Staff \relative c' <<
  % Initiate first voice
  \new Voice {
    \voiceOne
    c16^( d e f g4 f e | d2 e2) |
  }
  % Initiate second voice
  \new Voice {
    % Set stems, etc, down
    \voiceTwo
    s4 r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2 |
  }
  % Initiate third voice
  \new Voice {
    % Set stems, etc, up
    \voiceThree
    s1 | s4 b4 c2 |
  }
>>
```



Empilement des notes

Les notes rapprochées d'un accord, ou des notes de différentes voix qui tombent ensemble, seront rangées sur deux colonnes, voire plus, pour palier d'éventuels chevauchements des têtes. On appelle cela des empilements de notes. Chaque voix dispose de plusieurs empilements, et l'attribution d'un décalage à une voix en particulier s'appliquera à l'empilement en question s'il y avait risque de collision. Nous en avons une illustration à la deuxième mesure de l'exemple ci-dessus : le do de la deuxième voix est décalé à droite du ré de la première voix et, dans l'accord final, le do de la troisième voix est lui aussi décalé à droite des autres notes.

Les commandes `\shiftOn`, `\shiftOnn`, `\shiftOnnn`, et `\shiftOff` spécifient le degré nécessaire de décalage qui sera appliqué aux notes ou accords de la voix en question afin d'éviter une collision. Par défaut, les voix extérieures – normalement les première et deuxième – se verront attribuer `\shiftOff`, alors que les voix intérieures – trois et quatre – se verront attribuer `\shiftOn`. Lorsqu'un décalage s'applique, les voix un et trois iront vers la droite, et les voix deux et quatre vers la gauche.

`\shiftOnn` et `\shiftOnnn` définissent des degrés augmentés de décalage auquel on peut devoir temporairement recourir dans des situations complexes – voir Section 4.5.3 [Exemple concret], page 126.

Un empilement peut ne contenir qu'une note ou un accord dans une voix aux hampes vers le haut, et une note ou un accord dans une voix aux hampes vers le bas. Dans le cas où des notes, issues de deux voix ayant toutes deux des hampes dans la même direction, se retrouvent au même moment et qu'aucun décalage n'a été spécifié ou qu'ils sont identiques, LilyPond vous le signalera par le message « Trop d'empilements en conflit ».

Voir aussi
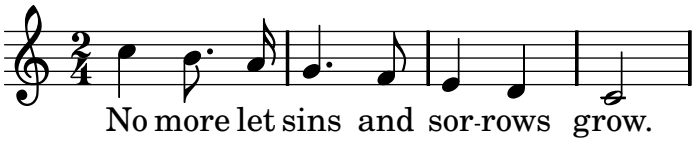
Manuel de notation : Section "Plusieurs voix" dans *Manuel de notation*.

3.2.3 Voix et paroles

La musique vocale est une gageure en soi : il nous faut combiner deux expressions différentes – des notes et des paroles.

Nous avons déjà abordé la commande `\addlyrics{}`, qui permet de gérer des partitions simples. Cette technique est cependant relativement limitée. Pour de la musique un peu plus compliquée, il vous faudra contenir les paroles dans un contexte `Lyrics`, créé par la commande `\new Lyrics` ; vous relierez ensuite ces paroles aux notes grâce à la commande `\lyricsto{}` et au nom assigné à la voix en question.

```
<<
  \new Voice = "one" {
    \relative c'' {
      \autoBeamOff
      \time 2/4
      c4 b8. a16 | g4. f8 | e4 d | c2 |
    }
  }
  \new Lyrics \lyricsto "one" {
    No more let | sins and | sor -- rows | grow. |
  }
>>
```



Notez bien que les paroles sont liées à un contexte de voix (`Voice`), **non** à un contexte de portée (`Staff`). Il est donc nécessaire de créer explicitement les contextes `Staff` et `Voice`.
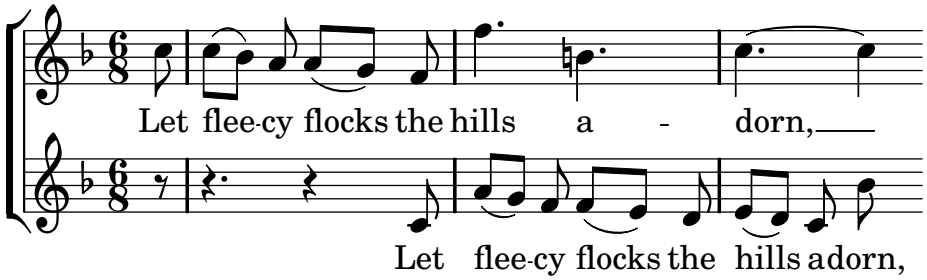
Si la ligature automatique que LilyPond applique par défaut est pleinement adaptée en matière de musique instrumentale, il n'en va pas de même dans le cas d'une musique associée à des paroles, et pour laquelle soit les ligatures sont carrément absentes, soit elles servent à indiquer un mélisme – plusieurs notes pour une même syllabe. Dans l'exemple qui suit, nous utilisons la commande `\autoBeamOff` afin de désactiver les ligatures automatiques.

Nous allons reprendre un extrait de Judas Maccabæus pour illustrer ce que cette technique apporte en flexibilité. Nous commençons par utiliser des variables – ou identificateurs – afin de séparer de la structure de la portée aussi bien la musique que les paroles. Nous ajoutons par la même occasion un crochet spécifique aux portées pour chœur (`ChoirStaff`). Quant aux blocs de paroles, nous les faisons précéder de la commande `\lyricmode` pour nous assurer qu'elles seront interprétées comme telles, et non comme de la musique.

```
global = { \key f \major \time 6/8 \partial 8 }

SopOneMusic = \relative c'' {
  c8 | c8([ bes)] a a([ g)] f | f'4. b, | c4.~ c4
}
SopOneLyrics = \lyricmode {
  Let | flee -- cy flocks the | hills a -- dorn, __
}
SopTwoMusic = \relative c' {
  r8 | r4. r4 c8 | a'8([ g)] f f([ e)] d | e8([ d)] c bes'
}
SopTwoLyrics = \lyricmode {
  Let | flee -- cy flocks the | hills a -- dorn,
}

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "SopOne" {
        \global
        \SopOneMusic
      }
      \new Lyrics \lyricsto "SopOne" {
        \SopOneLyrics
      }
    >>
    \new Staff <<
      \new Voice = "SopTwo" {
        \global
        \SopTwoMusic
      }
      \new Lyrics \lyricsto "SopTwo" {
        \SopTwoLyrics
      }
    >>
  >>
}
```



Voici donc la structure de base valable pour toute partition vocale. On peut y ajouter d'autres portées si besoin est, d'autres voix à chaque portée, plusieurs couplets aux paroles, et les variables contenant la musique peuvent même être stockées dans des fichiers indépendants dès lors que leur longueur devient conséquente.

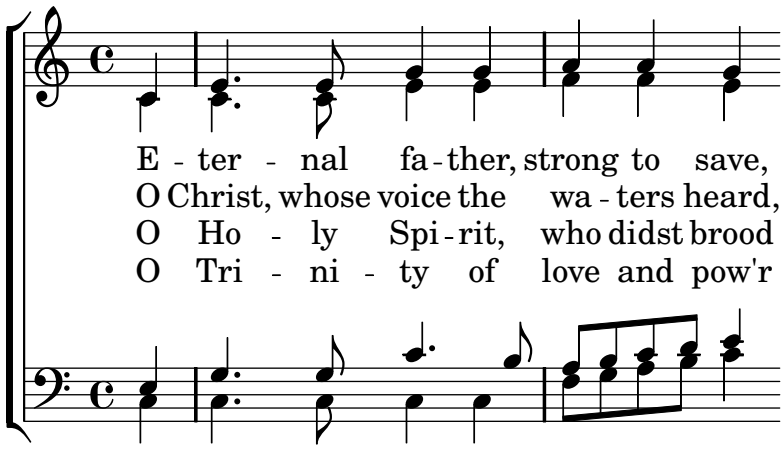
Voici maintenant la première ligne d'une hymne pour chœur à quatre voix mixtes, comportant quatre couplets. Les paroles sont ici identiques pour les quatre voix. Vous remarquerez le recours aux variables afin de séparer de la structure de portée aussi bien les notes que les paroles. Vous noterez aussi une variable particulière, que nous avons appelée « ArmureMetrique », et qui contient plusieurs commandes que nous utiliserons dans les deux portées. Dans de nombreux autres exemples, elle s'appelle « global ».

```
keyTime = { \key c \major \time 4/4 \partial 4 }

SopMusic   = \relative c' { c4 | e4. e8 g4  g    | a4   a   g  }
AltoMusic  = \relative c' { c4 | c4. c8 e4  e    | f4   f   e  }
TenorMusic = \relative c  { e4 | g4. g8 c4.   b8 | a8 b c d e4 }
BassMusic  = \relative c  { c4 | c4. c8 c4  c    | f8 g a b c4 }

VerseOne =
  \lyricmode { E -- | ter -- nal fa -- ther, | strong to save, }
VerseTwo   =
  \lyricmode { O | Christ, whose voice the | wa -- ters heard, }
VerseThree =
  \lyricmode { O | Ho -- ly Spi -- rit, | who didst brood }
VerseFour  =
  \lyricmode { O | Tri -- ni -- ty of | love and pow'r }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \clef "treble"
      \new Voice = "Sop"  { \voiceOne \keyTime \SopMusic }
      \new Voice = "Alto" { \voiceTwo \AltoMusic }
      \new Lyrics \lyricsto "Sop" { \VerseOne   }
      \new Lyrics \lyricsto "Sop" { \VerseTwo   }
      \new Lyrics \lyricsto "Sop" { \VerseThree }
      \new Lyrics \lyricsto "Sop" { \VerseFour  }
    >>
    \new Staff <<
      \clef "bass"
      \new Voice = "Tenor" { \voiceOne \keyTime \TenorMusic }
      \new Voice = "Bass"  { \voiceTwo \BassMusic }
    >>
  >>
}
```



Voir aussi

Manuel de notation : Section “Musique vocale” dans *Manuel de notation*.
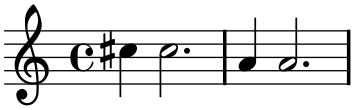
3.3 Contextes et graveurs

Nous avons évoqué rapidement les contextes et graveurs dans les chapitres précédents ; examinons en détail ces concepts essentiels à la maîtrise de LilyPond.

3.3.1 Tout savoir sur les contextes

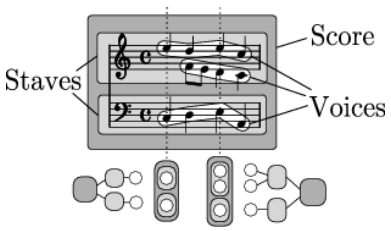
Imprimer de la musique impose d’ajouter un certain nombre d’éléments de notation. Par exemple, voici un fragment de partition, précédé du code qui l’engendre :

```
cis4 cis2. | a4 a2. |
```



Si le code est assez austère, dans la partition ont été ajoutés un chiffre de mesure, des barres de mesure, des altérations et une clé. Pour une bonne raison : LilyPond *interprète* le code. Il le compulse dans l’ordre chronologique, de même qu’on lit une partition de gauche à droite ; et pendant ce traitement, le logiciel garde en mémoire les limites des mesures, ou encore quelles hauteurs de note demandent des altérations accidentelles. Ces informations se présentent à plusieurs niveaux : ainsi, une altération n’a d’effet que sur une seule portée, tandis qu’une barre de mesure doit être synchronisée sur toute l’étendue verticale de la partition.

LilyPond regroupe ces règles et ces fragments d’information dans des *Contextes*. Certains contextes sont les voix (contexte `Voice`), les portées (contexte `Staff`), ou la partition dans son ensemble (contexte `Score`). Ils sont ordonnés hiérarchiquement : ainsi un contexte `Staff` peut contenir plusieurs contextes `Voice`, et un contexte `Score` peut contenir plusieurs contextes `Staff`.



Chaque contexte est chargé de faire appliquer certaines règles de gravure, de créer certains objets, et de prendre en compte les propriétés qui leur sont associées. Ainsi, le contexte `Voice` peut faire intervenir une altération accidentelle, puis le contexte `Staff` devra déterminer s’il faudra imprimer ou non cette dernière dans la suite de la mesure.

Les barres de mesure, quant à elles, sont alignées verticalement grâce au contexte `Score` par défaut. En revanche, dans une musique polymétrique, par exemple mêlant une portée à 3/4 et une autre à 4/4, les barres de mesures n'ont plus à être alignées : il faut alors modifier les comportement par défaut des contextes `Score` et `Staff`.

Dans une partition très simple, les contextes sont créés implicitement et peuvent être ignorés. Mais lorsqu'il s'agit de morceaux plus amples – entendons par là tout ce qui s'écrit sur plus d'une portée – il faut les créer explicitement pour être sûr d'obtenir toutes les portées nécessaires, et dans le bon ordre. Enfin, pour des morceaux impliquant une notation spéciale, modifier les contextes ou en créer de nouveaux devient extrêment utile.

En plus des contextes `Score`, `Staff` et `Voice` sont disponibles d'autres contextes intermédiaires entre les niveaux partition et portée, chargés de gérer certains regroupement, tels que `PianoStaff` ou `ChoirStaff`. Vous disposez aussi d'autres contextes de portée ou de voix alternatifs, ainsi que des contextes spécifiques pour les paroles, les percussions, les diagrammes pour instruments frettés, la basse chiffrée, etc.

Le nom de chacun des contextes est formé d'un ou plusieurs mots aux initiales en capitale et directement accolés les uns aux autres sans ponctuation, comme par exemple `GregorianTranscriptionStaff`.

Voir aussi

Manuel de notation : Section "Tout savoir sur les contextes" dans *Manuel de notation*.

3.3.2 Création d'un contexte

Il en va des contextes comme de toute hiérarchie : il faut un sommet – le contexte `Score` en l'occurence. La commande `\score` est chargée de le créer, mais pour des partitions simples, il le sera automatiquement. Le bloc `\score` contient donc une expression musicale unique ainsi que la définition des supports à produire – `\layout` pour du visuel ou `\midi` pour de l'acoustique.

Lorsqu'une partition ne comporte qu'une voix et une seule portée, vous pouvez laisser LilyPond créer automatiquement les contextes `Voice` et `Staff` ; mais leur présence explicite devient indispensable dès que la situation se complique. Le moyen le plus simple est d'utiliser la commande `\new`. Elle doit intervenir avant une expression musicale, ainsi :

```
\new type expression-musicale
```

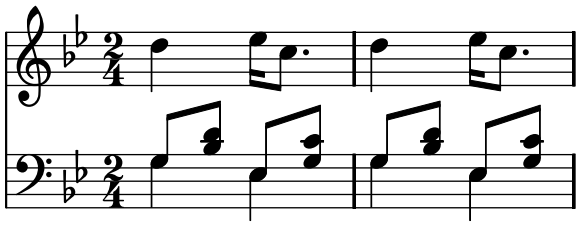
où *type* correspond au nom du contexte (tels `Staff` ou `Voice`). Cette commande crée un nouveau contexte, puis interprète l'*expression-musicale* contenue dans ledit contexte.

> **Note :** La commande `\new Score` ne devrait jamais servir en début de partition, puisque le contexte premier que constitue `Score` est créé automatiquement par l'interprétation de l'*expression-musicale* contenue dans le bloc `\score`. Les adaptations affectant les propriétés des différents contextes et qui s'appliqueront à l'ensemble de la partition trouvent leur place au sein d'un bloc `\layout`, en suivant les préceptes énoncés au chapitre Section 3.3.4 [Modification des propriétés d'un contexte], page 63.

Nous avons déjà vu au cours des chapitres précédents de nombreux exemples où des contextes `Staff` ou `Voice` étaient créés au besoin. Dans un but didactique, voici maintenant une application complète et abondamment commentée :

```
\score {  % start of single compound music expression
  <<  % start of simultaneous staves section
    \time 2/4
    \new Staff {  % create RH staff
```

```
      \key g \minor
      \clef "treble"
      \new Voice {  % create voice for RH notes
        \relative c'' {  % start of RH notes
          d4 ees16 c8. |
          d4 ees16 c8. |
        }  % end of RH notes
      }  % end of RH voice
    }  % end of RH staff
    \new Staff <<  % create LH staff; needs two simultaneous voices
      \key g \minor
      \clef "bass"
      \new Voice {  % create LH voice one
        \voiceOne
        \relative g {  % start of LH voice one notes
          g8 <bes d> ees, <g c> |
          g8 <bes d> ees, <g c> |
        }  % end of LH voice one notes
      }  % end of LH voice one
      \new Voice {  % create LH voice two
        \voiceTwo
        \relative g {  % start of LH voice two notes
          g4 ees |
          g4 ees |
        }  % end of LH voice two notes
      }  % end of LH voice two
    >>  % end of LH staff
  >>  % end of simultaneous staves section
}  % end of single compound music expression
```



Notez que toute déclaration qui ouvre un bloc par une accolade, `{`, ou un double chevron gauche, `<<`, est indentée de deux espaces supplémentaires, et de deux autres pour sa marque de fermeture. Bien que ceci ne soit pas obligatoire, nous vous invitons à adopter cette pratique qui vous évitera nombre d'erreurs « accolades non appariées ». La structure de la musique apparaît ainsi au premier coup d'œil, et les défauts de parité sont plus facilement repérables. Vous remarquerez que la portée MG est créée à l'aide d'un double chevron gauche – nécessaire pour gérer ses deux voix – alors que la portée MD ne contient qu'une seule expression musicale – il n'y a qu'une voix – bornée par des accolades simples.

La commande `\new` peut aussi permettre de nommer le contexte créé, et ainsi le distinguer des autres contextes déjà existants :

`\new` *`type = "UnNom" expression-musicale`*

Vous noterez la distinction entre le nom du type de contexte, `Staff`, `Voice`, etc. et le nom – une simple suite de lettres au bon gré de l'utilisateur – permettant d'identifier une instance particulière du type en question. Vous pouvez utiliser des chiffres et espaces, à la

stricte condition d'englober le tout dans des guillemets ; l'identificateur suivant est tout à fait valide : `\new Staff = "MaPortee 1" expression-musicale`. Comme nous l'avons déjà vu dans le chapitre consacré aux paroles (Section 3.2.3 [Voix et paroles], page 56), cet identifiant permettra ensuite de se référer à ce contexte particulier.

Voir aussi

Manuel de notation : Section "Création d'un contexte" dans *Manuel de notation*.

3.3.3 Tout savoir sur les graveurs

Tout point qui compose une partition générée par LilyPond est produit par un graveur (*engraver* en anglais). Ainsi, il y en a un qui imprime les portées, un autre les têtes de note, un autre les hampes, un autre encore pour les ligatures, etc. LilyPond dispose de plus de 120 graveurs ! La plupart des partitions ne requièrent de s'intéresser qu'à quelques-uns seulement, et pour des partitions simples, vous n'aurez même pas à vous en préoccuper.

Les graveurs résident et opèrent au sein des contextes. Les graveurs tels que le `Metronome_mark_engraver`, dont les effets s'appliquent à la partition dans son intégralité, opèrent au sein du contexte de plus haut niveau – le contexte `Score`.

Les graveurs `Clef_engraver` et `Key_engraver` seront logés dans chacun des contextes `Staff` ; deux portées peuvent requérir des clefs et des armures différentes.

Les graveurs `Note_heads_engraver` et `Stem_engraver` résident dans chacun des contextes `Voice`, contexte du plus bas niveau.

Chaque graveur confectionne les objets spécifiquement associés à sa fonction et traite les propriétés attachées à cette fonction. Ces propriétés, tout comme celles relatives aux contextes, peuvent être modifiées afin d'influencer le comportement du graveur et par voie de conséquence le rendu des éléments dont il a la charge.

Les graveurs ont tous un nom composé, formé des différents mots décrivant leur fonction. Seule l'initiale du premier mot est en majuscule, et les mots qui le composent sont joints par un caractère souligné. Ainsi, le `Staff_symbol_engraver` est chargé de créer les lignes de la portée, et le `Clef_engraver` détermine la hauteur de référence de la portée en dessinant le symbole de la clef.

Voici quelques-uns des graveurs les plus courants, ainsi que leur fonction. Vous noterez qu'il est facile d'en connaître la fonction à partir du nom, et vice versa.

Graveur	**Fonction**
Accidental_engraver	Crée les altérations, y compris de précaution, accidentelles ou suggérées
Beam_engraver	Grave les ligatures (*beams*)
Clef_engraver	Grave les clefs
Completion_heads_engraver	Divise les notes qui dépassent de la mesure
New_dynamic_engraver	Crée les soufflets et textes de nuance
Forbid_line_break_engraver	Empêche un saut de ligne si un élément musical est toujours actif
Key_engraver	Crée l'armure
Metronome_mark_engraver	Grave les indications métronomiques
Note_heads_engraver	Grave les têtes de note
Rest_engraver	Grave les silences
Staff_symbol_engraver	Grave les cinq lignes (par défaut) de la portée
Stem_engraver	Crée les hampes et les trémolos sur une hampe unique
Time_signature_engraver	Crée les métriques

Nous verrons plus avant comment le résultat de LilyPond peut changer lorsqu'on modifie l'action des graveurs.

Voir aussi

Référence des propriétés internes : Section "Engravers and Performers" dans *Référence des propriétés internes*.

3.3.4 Modification des propriétés d'un contexte

Les contextes gèrent les différentes valeurs des nombreuses *propriétés* qui leur sont attachées. Beaucoup d'entre elles sont susceptibles d'être modifiées afin d'influer sur l'interprétation de l'input et ainsi changer l'apparence du résultat. On les modifie grâce à la commande `\set`, qui s'utilise ainsi :

```
\set ContexteNommé.propriétéNommée = #valeur
```

où *ContexteNommé* est habituellement `Score`, `Staff` ou `Voice`. S'il n'est pas mentionné, il sera considéré comme étant `Voice`.

Les noms des propriétés de contexte sont composés de mots accolés sans trait d'union ni caractère souligné, et dont seul le premier n'aura pas d'initiale en majuscule. Voici quelques exemples de celles les plus communément utilisées.

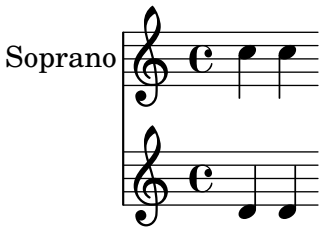
propriétéNommée	**Type**	**Fonction**	**Exemple de valeur**
extraNatural	Booléen	Si vrai, ajoute un bécarre avant une altération accidentelle	`#t`, `#f`
currentBarNumber	Entier	Détermine le numéro de la mesure en cours	`50`
doubleSlurs	Booléen	Si vrai, imprime les liaisons au-dessous **et** au-dessus des notes	`#t`, `#f`
instrumentName	Texte	Détermine le nom à afficher en début de portée	`"Cello I"`
fontSize	Réel	Augmente ou diminue la taille de la fonte	`2.4`
stanza	Texte	Détermine le texte à imprimer avant le début d'un couplet	`"2"`

où un boléen correspond soit à vrai (`#t` pour *True* en anglais) ou faux (`#f` pour *False* en anglais), un entier est un nombre entier positif, un réel est un nombre décimal positif ou négatif, et texte correspond à une suite de caractères encadrée par des apostrophes doubles. Attention à la présence des signes dièse (`#`) dans deux cas particuliers : ils sont partie intégrante des valeurs boléennes et précèdent les `t` ou `f`, mais doivent aussi précéder *valeur* dans le libellé de la commande `\set`. Il faudra donc, dans le cas d'une valeur boléenne, ne pas oublier de saisir deux signes dièse – par exemple `##t`.

Avant de déterminer l'une de ces propriétés, nous devons savoir dans quel contexte elle intervient. Si cela est bien souvent évident, il peut arriver que cela tourne au cauchemar. Lorsque vous ne spécifiez pas le bon contexte, aucun message d'erreur ne s'affiche et l'effet attendu n'est pas au rendez-vous. Par exemple, le `instrumentName` est de manière incontestable membre du contexte `Staff`, puisque c'est bien la portée que l'on va nommer. Dans l'exemple suivant, la première portée affiche effectivement un nom, alors que ce n'est pas le cas pour la deuxième dans la mesure où le contexte n'a pas été spécifié.

```
<<
  \new Staff \relative c'' {
    \set Staff.instrumentName = #"Soprano"
    c4 c
```

```
 }
  \new Staff \relative c' {
  \set instrumentName = #"Alto"  % Wrong!
  d4 d
 }
>>
```



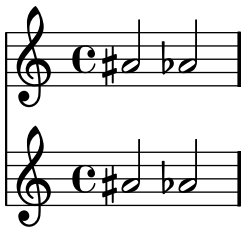
Dans la mesure où le nom de contexte par défaut est `Voice`, la deuxième commande `\set` a défini « Alto » comme propriété `instrumentName` du contexte de voix. Puisque LilyPond n'ira pas chercher une telle propriété dans le contexte `Voice`, celle-ci ne sera pas interprétée. Il ne s'agit pas d'une erreur, aucun message d'erreur ne sera ni émis ni enregistré.

De la même manière, une faute d'orthographe dans le nom de la propriété ne génèrera aucun message d'erreur et l'action escomptée ne se produira pas. Vous pourriez déterminer par la commande `\set` n'importe quelle « propriété », même fictive, à partir de n'importe quel nom et dans n'importe lequel des contextes disponibles. Mais tant que ce nom est inconnu de LilyPond, rien ne se passera. Certains éditeurs de texte disposent d'une prise en charge spécifique aux fichiers source LilyPond, à l'instar de LilyPondTool couplé à JEdit et qui documente les noms des propriétés dans une infobulle lorsque vous les survolez à la souris, ou les souligne différemment s'ils sont inconnus, comme ConTEXT. Dans le cas où votre éditeur ne dispose pas de ces fonctionnalités, nous vous recommandons de vérifier le nom des propriétés que vous manipulez dans la Référence des propriétés internes – voir Section "Tunable context properties" dans *Référence des propriétés internes*, ou Section "Contexts" dans *Référence des propriétés internes*.

La propriété `instrumentName` ne sera prise en compte que si elle est définie dans un contexte `Staff` ; d'autres propriétés peuvent par contre être définies dans plusieurs contextes différents. C'est le cas de la propriété `extraNatural` qui est définie par défaut à `##t` (vrai) pour toutes les portées. Si vous lui attribuez la valeur `##f` (faux) dans un contexte `Staff` particulier, elle ne s'appliquera qu'aux altérations de la portée en question ; si vous lui attribuez la valeur « faux » au niveau du contexte `Score`, cela s'appliquera alors à toutes les portées.

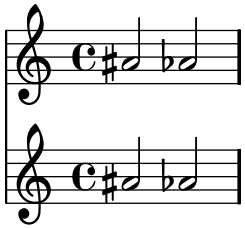
Voici comment supprimer les bécarres supplémentaires pour une portée :

```
<<
  \new Staff \relative c'' {
    ais2 aes
  }
  \new Staff \relative c'' {
    \set Staff.extraNatural = ##f
    ais2 aes
  }
>>
```



et pour toutes les portées :

```
<<
  \new Staff \relative c'' {
    ais2 aes
  }
  \new Staff \relative c'' {
    \set Score.extraNatural = ##f
    ais2 aes
  }
>>
```
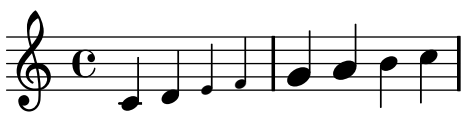


Autre exemple, si la propriété `clefOctavation` est déterminée au niveau du contexte `Score`, elle modifiera la valeur de l'octave en cours pour toutes les portées actives ; cette valeur sera considérée comme étant la nouvelle valeur par défaut pour toutes les portées à venir.

La commande opposée, `\unset`, efface la propriété du contexte ; la plupart des propriétés reviennent de ce fait à leur valeur par défaut. En règle générale, la commande `\unset` n'est pas nécessaire dès lors que vous faites appel à une nouvelle commande `\set` pour modifier le réglage.

Les commandes `\set` et `\unset` peuvent intervenir n'importe où dans votre fichier source. Elles seront effectives dès leur apparition et jusqu'à la fin de la partition, à moins d'être affectées par un `\unset` ou un nouveau `\set`. À titre d'exemple, nous allons jouer avec la taille des fontes, ce qui affecte entre autres la grosseur des têtes de note. Les modifications s'appliquent toujours par rapport à la valeur par défaut, non par rapport à la dernière valeur définie.

```
c4 d
% make note heads smaller
\set fontSize = #-4
e4 f |
% make note heads larger
\set fontSize = #2.5
g4 a
% return to default size
\unset fontSize
b4 c |
```



Nous venons de voir comment déterminer la valeur de différents types de propriété. N'oubliez pas que les nombres, entiers ou réels, doivent être précédés d'un signe dièse (`#`) et les valeurs

vrai ou faux de deux signes dièse – respectivement `##t` et `##f` –. Une valeur textuelle doit être encadrée de guillemets anglais, `"... "`, bien que, comme nous le constaterons plus tard, la commande `\markup` permet aussi de spécifier du texte.
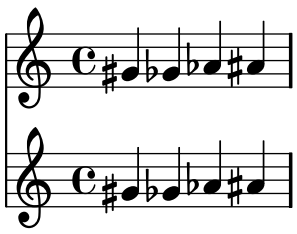
Définition des propriétés de contexte avec `\with`

Les propriétés d'un contexte peuvent aussi être réglées lors de la création de ce contexte. Ceci constitue parfois une façon plus claire de spécifier les valeurs d'une propriété pour la durée de vie du contexte. Lorsque vous créez un contexte à l'aide de la commande `\new`, vous pouvez la faire suivre immédiatement d'un bloc `\with { .. }` qui contiendra les réglages des différentes propriétés. Ainsi, si nous voulions par exemple annuler l'impression des bécarres supplémentaires sur la durée d'une portée, nous écririons :

```
\new Staff \with { extraNatural = ##f }
```

ce qui donnerait :

```
<<
  \new Staff
    \relative c'' {
      gis4 ges aes ais
    }
  \new Staff \with { extraNatural = ##f } {
    \relative c'' {
      gis4 ges aes ais
    }
  }
>>
```



Les propriétés réglées de cette manière peuvent néanmoins être modifiées de façon dynamique grâce à `\set` ; un `\unset` les ramènera à leur valeur initialisée par `\with`.
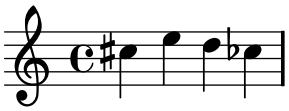
La propriété `fontSize` constitue une exception : lorsqu'elle est déterminée au sein d'un bloc `\with`, cela redéfinit la valeur par défaut de la taille de fonte. Une modification est possible par la commande `\set`, mais la commande `\unset fontSize` fera revenir à la nouvelle valeur par défaut.

Définition des propriétés de contexte avec `\context`

Vous pouvez régler les valeurs des propriétés de contexte en une seule fois pour tous les contextes d'un même type, par exemple tous les contextes `Staff`. Le type du contexte doit être donné explicitement d'après son nom, par exemple `Staff`, précédé d'une oblique inverse, donc nous saisirons `\Staff`. La manière de régler la valeur des propriétés est la même que ce que nous avons vu avec la commande `\with`, puisqu'on se place dans un bloc `\context` inclus dans un bloc `\layout`. Chaque bloc `\context` affectera tous les contextes concernés par le bloc `\score` ou `\book` au sein duquel apparaît ce bloc `\layout`. Voici comment le mettre en place :
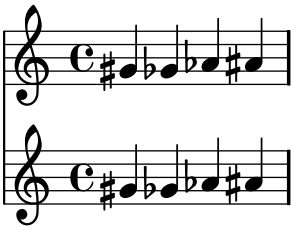
```
\score {
  \new Staff {
    \relative c'' {
```

```
      cis4 e d ces
    }
  }
  \layout {
    \context {
      \Staff
      extraNatural = ##t
    }
  }
}
```



Dans le cas où ces ajustements de propriété doivent affecter toutes les portées de la partition, nous utiliserons alors :

```
\score {
  <<
    \new Staff {
      \relative c'' {
        gis4 ges aes ais
      }
    }
    \new Staff {
      \relative c'' {
        gis4 ges aes ais
      }
    }
  >>
  \layout {
    \context {
      \Score extraNatural = ##f
    }
  }
}
```



Les propriétés de contexte ainsi définies peuvent être adaptées pour chacun des contextes en particulier grâce à un bloc `\with` ou bien une commande `\set` au fil des notes.

Voir aussi

Manuel de notation : Section “Modification des réglages par défaut d’un contexte” dans *Manuel de notation*, Section “La commande de fixation (set)” dans *Manuel de notation*.

Référence des propriétés internes : Section “Contexts” dans *Référence des propriétés internes*, Section “Tunable context properties” dans *Référence des propriétés internes*.

3.3.5 Ajout et suppression de graveurs

Nous avons vu que chaque contexte met en œuvre plusieurs graveurs, et que chacun de ces graveurs est chargé de générer une composante particulière du fichier de sortie, qui les barres de mesure, qui la portée, qui les têtes de note, les hampes, etc. Le fait de supprimer un graveur d'un contexte éliminera sa contribution au fichier de sortie. Bien que ce soit là un moyen radical de modifier le résultat, cette pratique est dans quelques cas fort utile.

Modification d'un seul contexte

Nous utilisons, pour supprimer un graveur d'un contexte, la commande `\with` dès la création dudit contexte, comme nous l'avons vu dans la section précédente.

Illustrons notre propos en reprenant un exemple du chapitre précédent, duquel nous supprimerons les lignes de la portée. Pour mémoire, les lignes d'une portée sont générées par le `Staff_symbol_engraver`.

```
\new Staff \with {
  \remove Staff_symbol_engraver
}
\relative c' {
  c4 d
  \set fontSize = #-4  % make note heads smaller
  e4 f |
  \set fontSize = #2.5  % make note heads larger
  g4 a
  \unset fontSize  % return to default size
  b4 c |
}
```
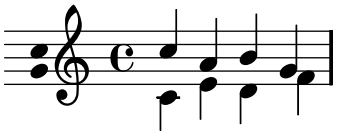


Vous pouvez aussi ajouter individuellement un graveur à un contexte. La commande se formule ainsi :

`\consists` *`Nom_du_graveur`*

et se place dans un bloc `\with`. Certaines partitions vocales font apparaître un Section "ambitus" dans *Glossaire* au début de la portée, afin d'indiquer ses notes extrêmes. L'ambitus est généré par l'`Ambitus_engraver`, que l'on peut adjoindre à n'importe quel contexte. Si nous l'ajoutons au contexte `Voice`, seule la tessiture de cette voix sera calculée :
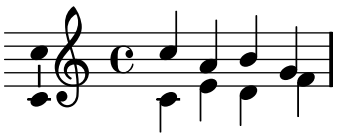
```
\new Staff <<
  \new Voice \with {
    \consists Ambitus_engraver
  } {
    \relative c'' {
      \voiceOne
      c4 a b g
    }
  }
  \new Voice {
    \relative c' {
      \voiceTwo
      c4 e d f
```

```
    }
  }
>>
```



alors que si nous l'ajoutons au contexte `Staff`, l'`Ambitus_engraver` calculera l'écart maximal à partir de toutes les notes de toutes les voix de la portée :

```
\new Staff \with {
  \consists Ambitus_engraver
}
<<
  \new Voice {
    \relative c'' {
      \voiceOne
      c4 a b g
    }
  }
  \new Voice {
    \relative c' {
      \voiceTwo
      c4 e d f
    }
  }
>>
```

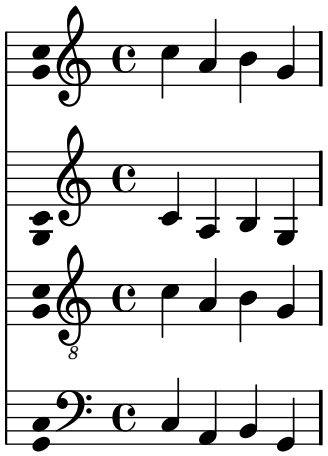


Modification de tous les contextes d'un même type

Les exemples ci-dessus nous ont montré comment ajouter ou retirer des graveurs à des contextes individuels. Nous pourrions aussi ajouter ou supprimer des graveurs à tous les contextes d'un même type en insérant les commandes pour le contexte approprié au sein d'un bloc `\layout`. Si nous voulions afficher un ambitus pour chacune des portées d'un système à quatre portées, il nous suffirait d'écrire :

```
\score {
  <<
    \new Staff {
      \relative c'' {
        c4 a b g
      }
    }
    \new Staff {
      \relative c' {
        c4 a b g
      }
    }
```

```
    \new Staff {
      \clef "G_8"
      \relative c' {
        c4 a b g
      }
    }
    \new Staff {
      \clef "bass"
      \relative c {
        c4 a b g
      }
    }
  >>
  \layout {
    \context {
      \Staff
      \consists Ambitus_engraver
    }
  }
}
```



Vous réglerez de la même manière les propriétés de tous les contextes d'un type particulier si vous insérez les commandes `\set` dans un bloc `\context`.

Voir aussi

Manuel de notation : Section "Modification des greffons de contexte" dans *Manuel de notation*, Section "Modification des réglages par défaut d'un contexte" dans *Manuel de notation*.

Problèmes connus et avertissements

Dans la mesure où les `Stem_engraver` et `Beam_engraver` rattachent à des têtes de note les objets qu'ils créent, désactiver le `Note_heads_engraver` entraîne l'absence de hampe et de ligature.

3.4 Extension des modèles

Bon, vous avez lu le tutoriel, vous savez écrire de la musique. Mais comment obtenir les portées que vous voulez ? Les Annexe A [Modèles], page 146, c'est bien beau, mais que faire quand ils ne traitent pas ce que l'on veut précisément ?

Les exemples qui suivent vous donneront des méthodes générales pour adapter des modèles.

3.4.1 Soprano et violoncelle

Commencez par le modèle qui vous semblera le plus proche de ce à quoi vous voulez aboutir. Disons par exemple que vous voulez écrire une pièce pour soprano et violoncelle : dans ce cas, on pourrait commencer par les « notes et paroles », pour la partie de soprano.

```
\version "2.14.2"

melodie = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

texte = \lyricmode {
  Aaa Bee Cee Dee
}

\score{
  <<
    \new Voice = "un" {
      \autoBeamOff
      \melodie
    }
    \new Lyrics \lyricsto "un" \texte
  >>
  \layout { }
  \midi { }
}
```

Maintenant, on veut ajouter une partie de violoncelle. Jetons un coup d'œil sur l'exemple avec les notes seules :

```
\version "2.14.2"

melodie = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

\score {
  \new Staff \melodie
  \layout { }
  \midi { }
}
```

On n'a pas besoin de deux commandes `\version`. Ce dont on a besoin, c'est de la section `melodie`. De même, on n'a pas besoin de deux sections `\score` – si nous les gardions toutes les deux, on obtiendrait deux parties séparées ; mais nous voulons un vrai duo, avec les deux parties ensemble. Dans la section `\score`, on n'a pas besoin non plus de deux `\layout` ni de deux `\midi`.

Si on se contente de couper et coller les sections `melodie`, on se retrouvera avec deux sections de ce nom ; il nous faut donc les renommer. Appelons la section pour la soprano `sopranoMusique` et celle pour le violoncelle `violoncelleMusique`. Tant qu'on y est, renommons `texte` en `sopranoParoles`. Attention à bien renommer les deux occurrences de chacune de ces dénominations : c'est-à-dire la définition de départ, où l'on trouve `melodie = \relative c' {` , et l'endroit où cette dénomination est utilisée, dans la section `\score`.

Et puis, toujours tant qu'on y est, mettons le violoncelle en clé de fa, comme le veut l'usage, et donnons-lui d'autres notes.

```
\version "2.14.2"

sopranoMusique = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

sopranoParoles = \lyricmode {
  Laaa Siii Dooo Rééé
}

violoncelleMusique = \relative c {
  \clef "bass"
  \key c \major
  \time 4/4
  d4 g fis8 e d4
}

\score{
  <<
    \new Voice = "un" {
      \autoBeamOff
      \sopranoMusique
    }
    \new Lyrics \lyricsto "un" \sopranoParoles
  >>
  \layout { }
  \midi { }
}
```

Voilà qui est mieux, mais la partie de violoncelle n'apparaît pas sur la partition – en effet, nous n'y avons pas fait appel dans la section `\score`. Si l'on veut que la partie de violoncelle s'imprime sous la partie de soprano, on va devoir ajouter :

```
\new Staff \musiqueVioloncelle
```

en dessous de tout ce qui concerne la soprano. Il nous faut également encadrer la musique par des `<<` et `>>`, qui feront comprendre à LilyPond que plusieurs événements – ici, des objets `Staff` – se déroulent en même temps. Le bloc `\score` ressemble maintenant à

```
\score {
  <<
  <<
    \new Voice = "un" {
```

```
      \autoBeamOff
      \sopranoMusique
    }
    \new Lyrics \lyricsto "un" \sopranoParoles
  >>
  \new Staff \violoncelleMusique
  >>
  \layout { }
  \midi { }
}
```

C'est un peu le bazar dans tout ça ; mais il vous sera facile de mettre un peu d'ordre dans l'indentation. Voici le modèle pour soprano et violoncelle au complet :

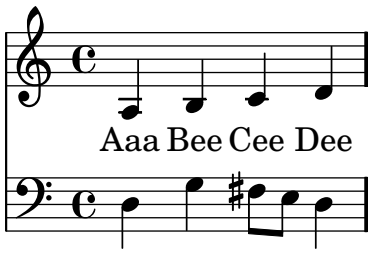
```
\version "2.14.2"


sopranoMusic = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

sopranoLyrics = \lyricmode {
  Aaa Bee Cee Dee
}

celloMusic = \relative c {
  \clef "bass"
  \key c \major
  \time 4/4
  d4 g fis8 e d4
}

\score {
  <<
    <<
      \new Voice = "one" {
        \autoBeamOff
        \sopranoMusic
      }
      \new Lyrics \lyricsto "one" \sopranoLyrics
    >>
    \new Staff \celloMusic
  >>
  \layout { }
  \midi { }
}
```

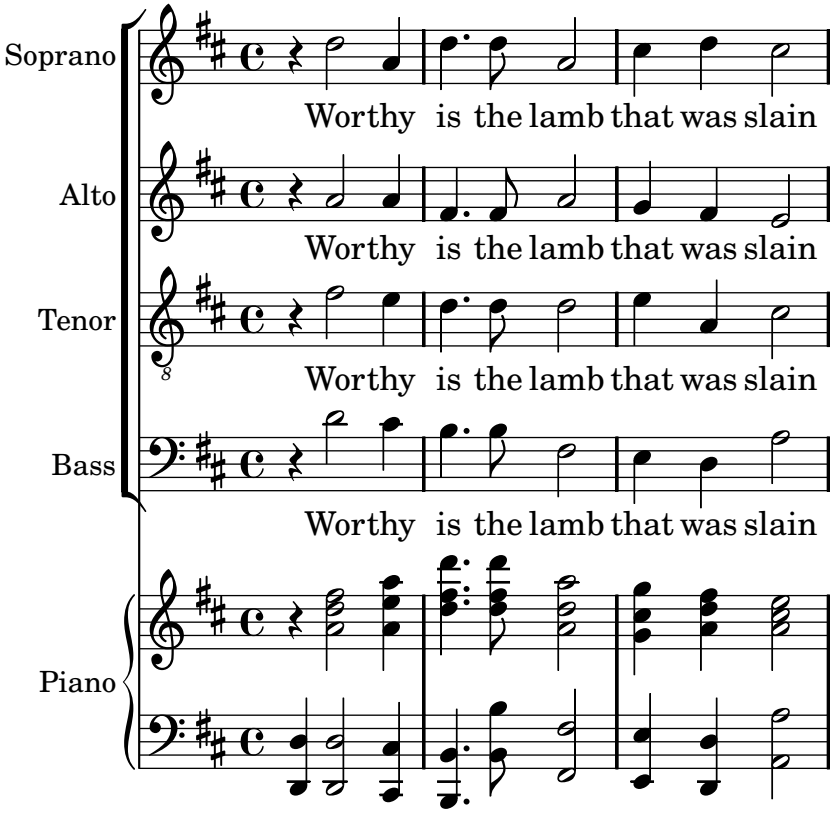


Voir aussi

Les patrons originaux sont disponibles à l'annexe « Modèles », voir Section A.1 [Portée unique], page 146.

3.4.2 Partition pour chœur à quatre voix mixtes

La plupart des œuvres écrites pour chœur à quatre voix mixtes et orchestre, comme Elias de Mendelssohn ou le Messie de Haendel, disposent la musique et les paroles du chœur sur quatre portées – soprano, alto, ténor et basse – surmontant une réduction pour piano de l'accompagnement orchestral. En voici un exemple, tiré du Messie de Haendel :



Aucun des modèles ne permet d'arriver exactement à cette mise en forme. Celui qui s'en rapprocherait le plus est « SATB vocal score and automatic piano reduction » – voir Section A.4 [Ensemble vocal], page 156 – mais encore faudrait-il en modifier la mise en forme et refaire la partie de piano qui n'est plus une simple reprise des parties vocales. Les variables qui gèrent la musique et les paroles du chœur ne nécessitent pas de modification, mais il nous faut d'autres variables pour la réduction de piano.

L'ordre dans lequel apparaissent les contextes dans le `ChoirStaff` du modèle ne correspond pas à ce que nous voyons ci-dessus. Il nous faudra y revenir pour obtenir quatre portées avec des paroles en dessous de chacune d'elles. Toutes les voix devraient être `\voiceOne`, ce qui est la position par défaut ; il nous faudra donc éliminer toutes les commandes `\voiceXXX`. Les ténors auront besoin d'une clé spécifique. Enfin, nous n'avons pas encore abordé la façon dont les

paroles sont présentées dans le modèle ; nous procèderons donc comme nous en avons l'habitude. Il faudra aussi ajouter un nom à chaque portée.

Une fois tout ceci accompli, voici notre `ChoirStaff` :

```
\new ChoirStaff <<
  \new Staff = "sopranos" <<
    \set Staff.instrumentName = #"Soprano"
    \new Voice = "sopranos" {
      \global
      \sopranoMusique
    }
  >>
  \new Lyrics \lyricsto "sopranos" {
    \sopranoParoles
  }
  \new Staff = "altos" <<
    \set Staff.instrumentName = #"Alto"
    \new Voice = "altos" {
      \global
      \altoMusique
    }
  >>
  \new Lyrics \lyricsto "altos" {
    \altoParoles
  }
  \new Staff = "tenors" <<
    \set Staff.instrumentName = #"Tenor"
    \new Voice = "tenors" {
      \global
      \tenorMusique
    }
  >>
  \new Lyrics \lyricsto "tenors" {
    \tenorParoles
  }
  \new Staff = "basses" <<
    \set Staff.instrumentName = #"Bass"
    \new Voice = "basses" {
      \global
      \bassMusique
    }
  >>
  \new Lyrics \lyricsto "basses" {
    \bassParoles
  }
>>  % fin du ChoirStaff
```

Il nous faut maintenant nous occuper de la partie de piano. Nous allons nous contenter de récupérer la partie de piano du modèle « Solo piano » :

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano  "
  \new Staff = "upper" \superieur
  \new Staff = "lower" \inferieur
```

```
>>
```

puis d'ajouter les définitions de variable pour `superieur` et `inferieur`.

Les systèmes pour chœur et pour piano doivent être combinés à l'aide de doubles chevrons gauche/droite puisqu'ils doivent s'empiler :

```
<<  % combine ChoirStaff et PianoStaff l'un au-dessus de l'autre
  \new ChoirStaff <<
    \new Staff = "sopranos" <<
      \new Voice = "sopranos" {
        \global
        \sopranoMusique
      }
    >>
    \new Lyrics \lyricsto "sopranos" {
      \sopranoParoles
     }
    \new Staff = "altos" <<
      \new Voice = "altos" {
        \global
        \altoMusique
      }
    >>
    \new Lyrics \lyricsto "altos" {
      \altoParoles
    }
    \new Staff = "tenors" <<
      \clef "G_8"  % tenor clef
      \new Voice = "tenors" {
        \global
        \tenorMusique
      }
    >>
    \new Lyrics \lyricsto "tenors" {
      \tenorParoles
    }
    \new Staff = "basses" <<
      \clef "bass"
      \new Voice = "basses" {
        \global
        \bassMusique
      }
    >>
    \new Lyrics \lyricsto "basses" {
      \bassParoles
    }
  >>  % fin du ChoirStaff

  \new PianoStaff <<
    \set PianoStaff.instrumentName = #"Piano"
    \new Staff = "upper" \superieur
    \new Staff = "lower" \inferieur
  >>
```

```
>>
```

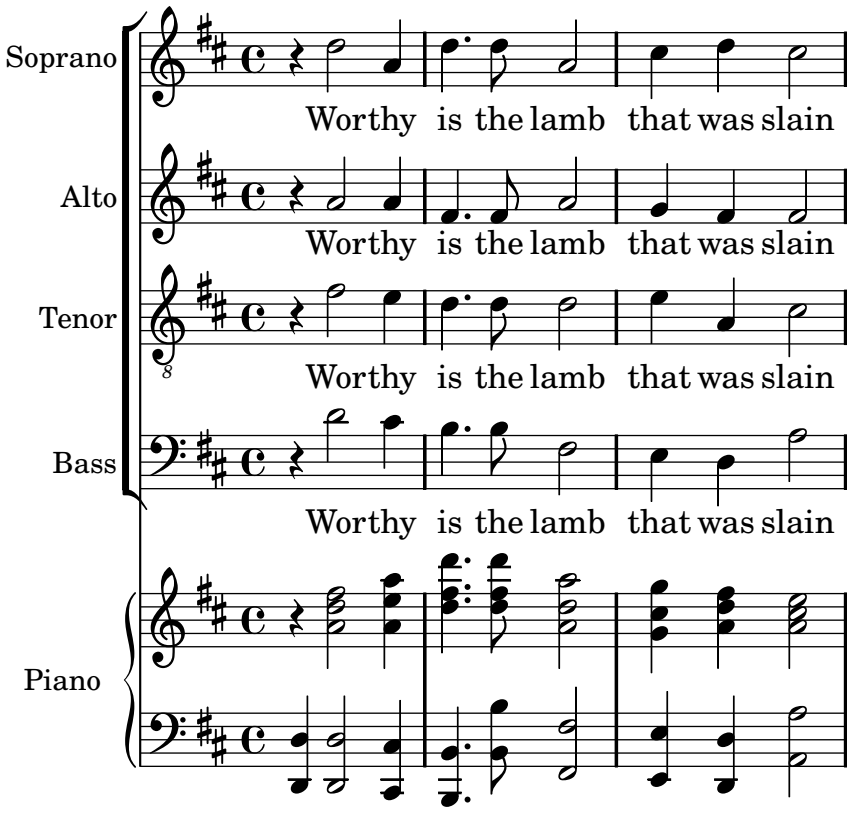
Une fois tout cela mis en place, et après avoir ajouté les notes et les paroles de ces trois mesures du Messie, nous obtenons :

```
\version "2.14.2"
global = { \key d \major \time 4/4 }
sopranoMusic = \relative c'' {
  \clef "treble"
  r4 d2 a4 | d4. d8 a2 | cis4 d cis2 |
}
sopranoWords = \lyricmode {
  Wor -- thy | is the lamb | that was slain |
}
altoMusic = \relative a' {
  \clef "treble"
  r4 a2 a4 | fis4. fis8 a2 | g4 fis fis2 |
}
altoWords = \sopranoWords
tenorMusic = \relative c' {
  \clef "G_8"
  r4 fis2 e4 | d4. d8 d2 | e4 a, cis2 |
}
tenorWords = \sopranoWords
bassMusic = \relative c' {
  \clef "bass"
  r4 d2 cis4 | b4. b8 fis2 | e4 d a'2 |
}
bassWords = \sopranoWords
upper = \relative a' {
  \clef "treble"
  \global
  r4 <a d fis>2 <a e' a>4 |
  <d fis d'>4. <d fis d'>8 <a d a'>2 |
  <g cis g'>4 <a d fis> <a cis e>2 |
}
lower = \relative c, {
  \clef "bass"
  \global
  <d d'>4 <d d'>2 <cis cis'>4 |
  <b b'>4. <b' b'>8 <fis fis'>2 |
  <e e'>4 <d d'> <a' a'>2 |
}

\score {
  <<  % combine ChoirStaff and PianoStaff in parallel
    \new ChoirStaff <<
      \new Staff = "sopranos" <<
        \set Staff.instrumentName = #"Soprano"
        \new Voice = "sopranos" {
          \global
          \sopranoMusic
        }
```

```
        >>
        \new Lyrics \lyricsto "sopranos" {
          \sopranoWords
        }
        \new Staff = "altos" <<
          \set Staff.instrumentName = #"Alto"
          \new Voice = "altos" {
            \global
            \altoMusic
          }
        >>
        \new Lyrics \lyricsto "altos" {
          \altoWords
        }
        \new Staff = "tenors" <<
          \set Staff.instrumentName = #"Tenor"
          \new Voice = "tenors" {
            \global
            \tenorMusic
          }
        >>
        \new Lyrics \lyricsto "tenors" {
          \tenorWords
        }
        \new Staff = "basses" <<
          \set Staff.instrumentName = #"Bass"
          \new Voice = "basses" {
            \global
            \bassMusic
          }
        >>
        \new Lyrics \lyricsto "basses" {
          \bassWords
        }
      >>  % end ChoirStaff

      \new PianoStaff <<
        \set PianoStaff.instrumentName = #"Piano  "
        \new Staff = "upper" \upper
        \new Staff = "lower" \lower
      >>
    >>
  }
```



3.4.3 Écriture d'une partition à partir de zéro

Après avoir acquis une certaine dextérité dans l'écriture de code LilyPond, vous devez vous sentir suffisament prêt à vous lancer dans la création d'une partition à partir de zéro, autrement dit en ne partant pas d'un exemple. Vous pourrez ainsi vous construire vos propres patrons selon le type de musique que vous affectionnez plus particulièrement. Pour voir comment procéder, nous allons monter la partition d'un prélude pour orgue.

Nous débutons par une section d'en-tête ; nous y mettrons entre autres le titre et le nom du compositeur. Puis viennent toutes les définitions de toutes les variables. Nous terminons par le bloc `\score`. Attelons-nous pour cette aventure, en gardant bien à l'esprit ce que nous venons de dire ; nous nous occuperons des détails en temps voulu.

Nous nous appuyons sur les deux premières mesures du prélude sur *Jesu, meine Freude*, écrit pour orgue avec pédalier. Vous pouvez voir ces deux mesures au bas de cette page. La main droite comporte deux voix, la main gauche et le pédalier une seule. Il nous faut donc quatre définitions de musique, plus une qui contiendra la métrique et l'armure :

```
\version "2.14.2"
\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
ArmureMetrique = { \key c \minor \time 4/4 }
ManuelUnVoixUnMusique = { s1 }
ManuelUnVoixDeuxMusique = { s1 }
ManuelDeuxMusique = { s1 }
PedalierOrgueMusique = { s1 }

\score {
}
```

Pour l'instant, nous utilisons des silences invisibles, `s1`, en lieu et place des notes réelles. On verra plus tard.

Passons maintenant au bloc `\score` et à ce qu'il devrait contenir. Nous y recopions simplement la structure des portées que nous voulons. La musique pour orgue se présente généralement sous la forme de trois portées, une pour chaque main et une pour le pédalier. Les portées du manuel sont regroupées, nous utiliserons donc un `PianoStaff`. La première partie du manuel requiert deux voix et la seconde une seule.

```
\new PianoStaff <<
  \new Staff = "ManuelUn" <<
    \new Voice {
      \ManuelUnVoixUnMusique
    }
    \new Voice {
      \ManuelUnVoixDeuxMusique
    }
  >>  % fin du contexte de portée ManuelUn
  \new Staff = "ManuelDeux" <<
    \new Voice {
      \ManuelDeuxMusique
    }
  >>  % fin du contexte de portée ManuelDeux
>>  % fin du contexte PianoStaff
```

Il nous faut ajouter à cela une portée pour le pédalier. Elle se place sous le système de piano, mais puisqu'elle doit rester synchrone avec lui, nous utilisons des doubles chevrons pour les regrouper. Négliger ceci nous renverrait une erreur, et personne n'est à l'abri de cette faute ! Pour preuve, il vous suffit de copier l'exemple complet en fin de chapitre, de supprimer ces `<<` et `>>`, et de le compiler, pour voir de quoi il retourne.

```
<<  % Système pianistique et portée de pédalier sont synchrones
  \new PianoStaff <<
    \new Staff = "ManuelUn" <<
      \new Voice {
        \ManuelUnVoixUnMusique
      }
      \new Voice {
        \ManuelUnVoixDeuxMusique
      }
    >>  % fin du contexte de portée ManuelUn
    \new Staff = "ManualDeux" <<
      \new Voice {
        \ManuelDeuxMusique
      }
    >>  % fin du contexte de portée ManuelDeux
  >>  % fin du contexte PianoStaff
  \new Staff = "PedalierOrgue" <<
    \new Voice {
      \PedalierOrgueMusique
     }
  >>
>>
```

La construction en simultané – `<<...>>` – n'est pas strictement obligatoire pour les portées manuel deux et pédalier, qui ne contiennent chacune qu'une seule expression musicale ; mais cela ne mange pas de pain, et c'est une bonne habitude que de toujours encadrer par des doubles chevrons gauche/droite ce qui suit une commande `\new Staff` au cas où il y aurait plusieurs voix.

Il en va autrement pour les contextes `Voice` : ils doivent être toujours suivis d'accolades – `{...}` – au cas où vous auriez employé plusieurs variables qui doivent intervenir consécutivement.

Ajoutons donc cette structure au bloc `\score`, tout en fignolant l'indentation. Nous en profitons pour ajouter les clés appropriées, effectuer les réglages concernant les hampes et liaisons de la portée supérieure grâce à `\voiceOne` et `\voiceTwo`, et mettre en place la métrique et l'armure de chaque portée grâce à notre variable `\MetriqueArmure`.

```
\score {
  <<  % Système pianistique et portée de pédalier sont synchrones
    \new PianoStaff <<
      \new Staff = "ManuelUn" <<
        \ArmureMetrique % définition de l'armure et de la métrique
        \clef "treble"
        \new Voice {
          \voiceOne
          \ManuelUnVoixUnMusique
        }
        \new Voice {
          \voiceTwo
          \ManuelUnVoixDeuxMusique
        }
      >>  % fin du contexte de la portée ManuelUn
      \new Staff = "ManuelDeux" <<
        \ArmureMetrique
        \clef "bass"
        \new Voice {
          \ManuelDeuxMusique
        }
      >>  % fin du contexte de la portée ManuelDeux
    >>  % fin du contexte PianoStaff
    \new Staff = "PedalierOrgue" <<
      \ArmureMetrique
      \clef "bass"
      \new Voice {
        \PedalierOrgueMusique
      }
    >>  % fin du contexte de la portée PedalOrgan
  >>
}  % fin du contexte Score
```

Cette partition pour orgue est préesque parfaite. Rest juste ce petit défaut qui ne se remarque pas lorsque l'on considère un seul système : la distance qui sépare la portée de pédalier de celle de la main gauche devrait être plus ou moins égale à celle qui sépare les deux mains. En fait, la distance entre les deux portées d'un `PianoStaff` ne saurait trop se dilater ; le pédalier devrait adopter le même comportement.

La propension des portées à se dilater se contrôle à l'aide de la propriété `staff-staff-spacing`, attachée à « l'objet graphique » `VerticalAxisGroup` – la documentation de LilyPond utilise souvent l'abréviation *grob* pour *graphical object*. Pas de panique ! Tout ceci sera expliqué plus tard – pour les curieux, jetez un œil au chapitre Section "Vue d'ensemble de la modification des propriétés" dans *Manuel de notation*. Revenons à notre propos : nous voulons modifier uniquement la sous-propriété `stretchability`. Les impatients trouveront les valeurs par défaut de la propriété `staff-staff-spacing` dans le fichier '`scm/define-grobs.scm`', en examinant la définition du *grob* `VerticalAxisGroup`. La

valeur que nous affecterons à `stretchability` est celle que contient la définition du contexte `PianoStaff` telle qu'elle apparaît dans le fichier '`ly/engraver-init.ly`').

```
\score {
  <<  % Système pianistique et portée de pédalier sont synchrones
    \new PianoStaff <<
      \new Staff = "ManuelUn" <<
        \ArmureMetrique % définition de l'armure et de la métrique
        \clef "treble"
        \new Voice {
          \voiceOne
          \ManuelUnVoixUnMusique
        }
        \new Voice {
          \voiceTwo
          \ManuelUnVoixDeuxMusique
        }
      >>  % fin du contexte de la portée ManuelUn
      \new Staff = "ManuelDeux" \with {
        \override VerticalAxisGroup
          #'staff-staff-spacing  #'stretchability = 5
      } <<
        \ArmureMetrique
        \clef "bass"
        \new Voice {
          \ManuelDeuxMusique
        }
      >>  % fin du contexte de la portée ManuelDeux
    >>  % fin du contexte PianoStaff
    \new Staff = "PedalierOrgue" <<
      \ArmureMetrique
      \clef "bass"
      \new Voice {
        \PedalierOrgueMusique
      }
    >>  % fin du contexte de la portée PedalOrgan
  >>
}  % fin du contexte Score
```

Nous en avons fini avec la structure. Toutes les partitions pour orgue auront cette structure, même si le nombre de voix peut changer. Tout ce qui nous reste à faire maintenant consiste à saisir la musique et à regrouper toutes les parties.
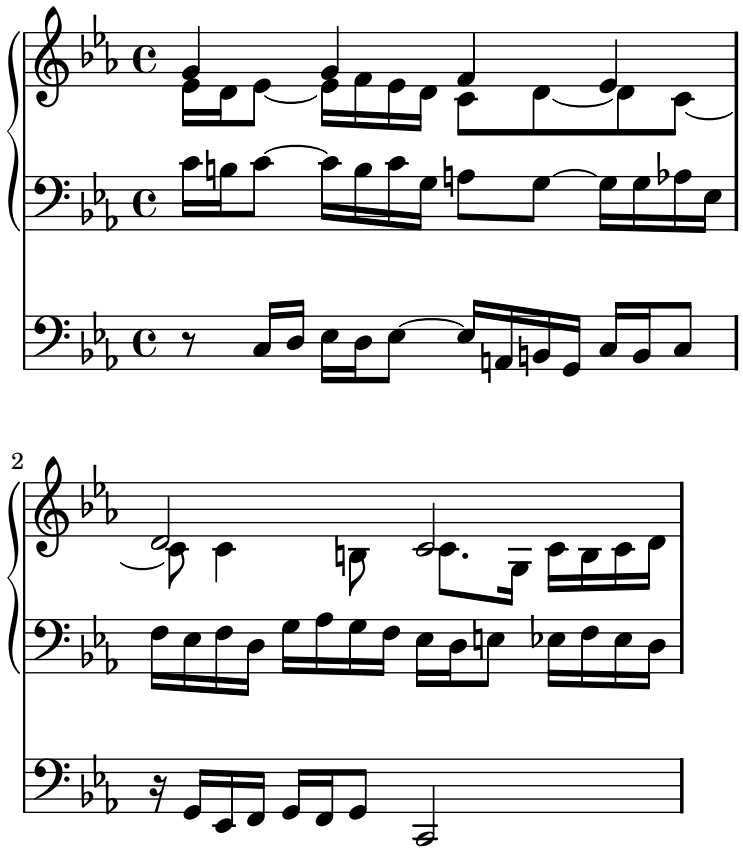
```
\version "2.14.2"
\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
keyTime = { \key c \minor \time 4/4 }
ManualOneVoiceOneMusic = \relative g' {
  g4 g f ees |
  d2 c |
}
ManualOneVoiceTwoMusic = \relative c' {
```

```
  ees16 d ees8~ ees16 f ees d c8 d~ d c~ |
  c8 c4 b8 c8. g16 c b c d |
}
ManualTwoMusic = \relative c' {
  c16 b c8~ c16 b c g a8 g~ g16 g aes ees |
  f16 ees f d g aes g f ees d e8~ ees16 f ees d |
}
PedalOrganMusic = \relative c {
  r8 c16 d ees d ees8~ ees16 a, b g c b c8 |
  r16 g ees f g f g8 c,2 |
}

\score {
  <<  % PianoStaff and Pedal Staff must be simultaneous
    \new PianoStaff <<
      \new Staff = "ManualOne" <<
        \keyTime  % set key and time signature
        \clef "treble"
        \new Voice {
          \voiceOne
          \ManualOneVoiceOneMusic
        }
        \new Voice {
          \voiceTwo
          \ManualOneVoiceTwoMusic
        }
      >>  % end ManualOne Staff context
      \new Staff = "ManualTwo" \with {
        \override VerticalAxisGroup
          #'staff-staff-spacing #'stretchability = 5
      } <<
        \keyTime
        \clef "bass"
        \new Voice {
          \ManualTwoMusic
        }
      >>  % end ManualTwo Staff context
    >>  % end PianoStaff context
    \new Staff = "PedalOrgan" <<
      \keyTime
      \clef "bass"
      \new Voice {
        \PedalOrganMusic
      }
    >>  % end PedalOrgan Staff context
  >>
}  % end Score context
```

Jesu, meine Freude

J S Bach
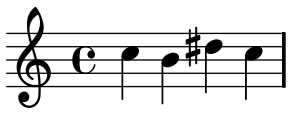


Voir aussi

Glossaire musicologique : Section “système” dans *Glossaire*.

3.4.4 Économie de saisie grâce aux identificateurs et fonctions

Jusqu’à maintenant, vous avez vu ce type de code :

```
hornNotes = \relative c'' { c4 b dis c }
\score {
  {
    \hornNotes
  }
}
```
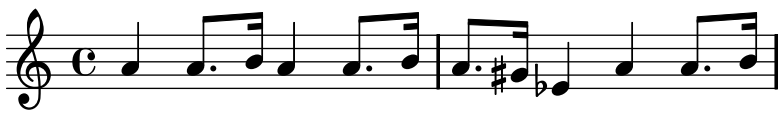


Vous comprendrez combien cela peut être utile pour écrire de la musique minimaliste :

```
fragmentA = \relative c'' { a4 a8. b16 }
fragmentB = \relative c'' { a8. gis16 ees4 }

violin = \new Staff { \fragmentA \fragmentA \fragmentB \fragmentA }

\score {
  {
    \violin
  }
}
```
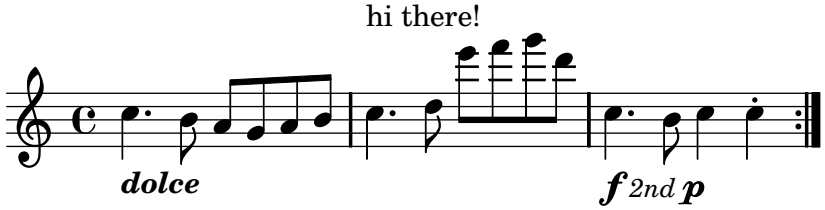


Néanmoins vous pouvez aussi utiliser ces identificateurs – aussi connus sous le nom de variables, macros, ou commandes (définies par l'utilisateur) – pour des retouches :

```
dolce = \markup { \italic \bold dolce }

padText = { \once \override TextScript #'padding = #5.0 }
fthenp =_\markup {
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
}

violin = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \padText
    c4.^"hi there!" d8 e' f g d |
    c,4.\fthenp b8 c4 c-. |
  }
}

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



Ces identificateurs sont évidemment utiles pour économiser de la frappe. Mais ils peuvent l'être même si vous ne les utilisez qu'une seule fois : ils réduisent la complexité. Regardons l'exemple précédent sans aucun identificateur. C'est beaucoup plus laborieux à lire, et particulièrement la dernière ligne.
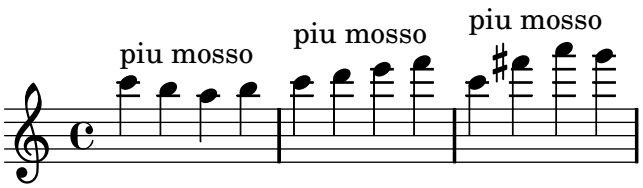
```
violin = \relative c'' {
  \repeat volta 2 {
    c4._\markup { \italic \bold dolce } b8 a8 g a b |
    \once \override TextScript #'padding = #5.0
    c4.^"hi there!" d8 e' f g d |
    c,4.\markup {
      \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
    }
    b8 c4 c-. |
  }
}
```

Jusqu'ici nous avons vu des substitutions statiques : quand LilyPond rencontre `\padText`, il le remplace par le contenu que nous lui avons défini – c'est-à-dire le contenu à droite de `padText=`.

LilyPond gère également des substitutions non-statiques – vous pouvez les voir comme des fonctions.

```
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 |
}
```



Utiliser des identificateurs est aussi un bon moyen pour vous épargner du travail si la syntaxe de LilyPond change un jour – voir Section “Mise à jour avec convert-ly” dans *Utilisation des programmes*. Si vous avez une seule définition, par exemple `\dolce`, pour tous vos fichiers (voir Section 4.6.3 [Feuilles de style], page 138) et que la syntaxe change, alors vous n’aurez qu’à mettre à jour votre seule définition `\dolce`, au lieu de devoir modifier chaque fichier ‘`.ly`’.

3.4.5 Conducteurs et parties

Dans la musique d’orchestre, toutes les notes sont imprimées deux fois. D’abord dans les parties séparées destinées aux musiciens, et ensuite dans le conducteur destiné au chef. Les variables sont là pour vous éviter un double travail. La musique n’est entrée qu’une seule fois, et stockée dans une variable, dont le contenu servira à imprimer à la fois la partie séparée et la partition d’orchestre.

Il est judicieux de définir les notes dans un fichier séparé. Par exemple, supposons que le fichier ‘`musique-Cor.ly`’ contienne la partie suivante pour un duo cor/basson.

```
notesCor = \relative c {
  \time 2/4
  r4 f8 a | cis4 f | e d |
}
```

On établira alors une partie séparée en constituant un nouveau fichier :

```
\include "musique-Cor.ly"

\header {
  instrument = "Cor en Fa"
}
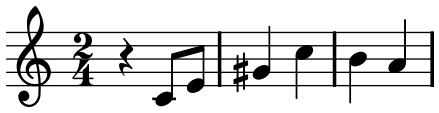
{
 \transpose f c' \notesCor
```

```
}
```

À la ligne

```
\include "musique-Cor.ly"
```

sera substitué le contenu du fichier ‘`musique-Cor.ly`’, et de ce fait la variable `notesCor` se trouvera définie. La commande `\transpose f c'` indique que son argument `\notesCor` sera transposé à la quinte supérieure : le son réel `f` s’écrit `c'`, ce qui est la caractéristique d’un Cor en fa. La transposition est visible comme suit :
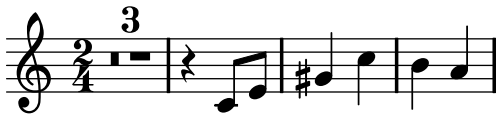


Dans les pièces d’ensemble, il arrive souvent qu’une voix ne joue pas pendant plusieurs mesures. Un silence spécial, appelé silence multimesures, l’indique alors. On l’obtient par un `R` majuscule, suivi d’une durée : `1` pour une pause, `2` pour une demi-pause, etc. Cette durée peut être multipliée pour établir de plus longs silences. Par exemple, le silence suivant dure 3 mesures à 2/4.

```
R2*3
```

Dans une partie séparée, les silences multimesure sont compressés. Il faut pour cela définir la propriété `skipBars` à « vrai » :

```
\set Score.skipBars = ##t
```

Cette commande assigne la valeur « vrai » – *true* en anglais, et `#t` dans le langage Scheme – à cette propriété dans le contexte `Score`. Si l’on ajoute dans la musique ci-dessus le silence multimesure et cette option, on obtient le résultat suivant :

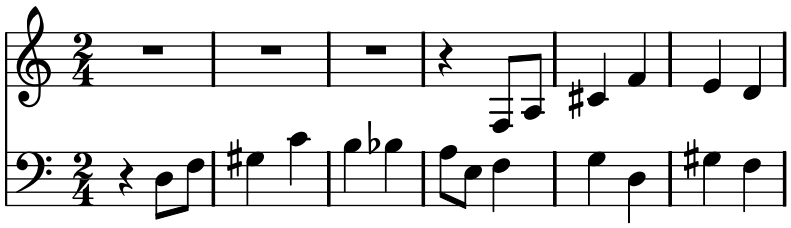


Le conducteur rassemble toute la musique. Si l’on suppose que l’autre voix de notre duo se trouve dans le fichier ‘`musique-Basson.ly`’ en tant que variable `notesBasson`, on établira un conducteur avec

```
\include "musique-Basson.ly"
\include "musique-Cor.ly"

<<
  \new Staff \notesCor
  \new Staff \notesBasson
>>
```

ce qui équivaut à



4 Retouche de partition

Ce chapitre indique comment modifier le résultat obtenu. LilyPond offre de nombreuses possibilités de réglages, permettant théoriquement de modifier chaque élément de votre partition.

4.1 Retouches élémentaires

4.1.1 Introduction aux retouches

LilyPond regroupe sous le terme de « retouches » (*tweaks* en anglais) les différents moyens dont dispose l'utilisateur pour intervenir sur l'interprétation du fichier d'entrée et pour modifier l'apparence du fichier de sortie. Certaines retouches sont très simples à mettre en œuvre ; d'autres sont plus complexes. Mais à elles toutes, elles permettent d'obtenir tout ce qu'on veut en matière de musique imprimée.

Dans ce chapitre, nous traitons des concepts de base nécessaires pour comprendre l'art de la retouche. Puis nous présentons de nombreuses commandes déjà prêtes, qu'il suffit de recopier pour obtenir un résultat identique dans vos partitions ; nous en profitons pour expliquer comment ces commandes ont été construites, si bien que vous pouvez apprendre par la même occasion à développer vos propres retouches.

Avant de vous lancer dans ce chapitre, il peut être utile de revoir la section Section 3.3 [Contextes et graveurs], page 59, dans la mesure où les contextes, graveurs et autres propriétés qui y sont décrits, sont indispensables pour comprendre et construire les retouches.

4.1.2 Objets et interfaces

Toute retouche implique que l'on modifie les opérations internes et les structures du programme LilyPond. Nous allons donc, pour commencer, présenter certains termes qui servent à décrire ces opérations internes et ces structures.

Le terme « Objet » est un terme générique qui fait référence à une multitude de structures internes mises en place par LilyPond durant la compilation d'un fichier d'entrée. Ainsi, quand une commande du type `\new Staff` apparaît, un nouvel objet du type `Staff` est créé. Cet objet `Staff` contient alors toutes les propriétés associées à cette portée, par exemple son nom et son armure, ainsi que le détail des graveurs qui ont été désignés pour fonctionner dans ce contexte de portée. Certains objets peuvent contenir les propriétés de tous les autres contextes, comme les objets `Voice`, les objets `Score`, les objets `Lyrics` ; d'autres se rapportent à tous les éléments de notation, comme les barres de mesure, les liaisons, les nuances, etc. Chaque objet dispose de son propre échantillon de valeurs pour le réglage des propriétés.

Certains types d'objet portent des noms spécifiques. Les objets qui se rapportent à des éléments de notation sur le fichier de sortie, comme les notes, les hampes, les liaisons de phrasé ou de prolongation, les doigtés, les clefs, etc. sont appelés « Objets de rendu » ; ils sont aussi connus sous le nom d'« Objets graphiques » (en anglais : *Graphical objects* ou *Grobs* pour faire court). Ce sont bien des objets au sens générique ci-dessus, et, en tant que tels, ils reçoivent des propriétés qui leur sont associées, comme leur position, leur taille, leur couleur, etc.

Certains objets de rendu, comme les liaisons de phrasé, les soufflets de crescendo, les marques d'octaviation et beaucoup d'autres *grobs*, ont pour particularité de ne pas se situer à un seul et unique endroit – ils ont un point de départ, un point d'arrivée, et éventuellement d'autres propriétés relatives à leur forme. Ces objets avec une forme étendue sont appelés des bandeaux (*Spanners* en anglais).

Il reste à expliquer ce que sont les « interfaces ». De nombreux objets, qui peuvent être très différents les uns des autres, ont pour point commun de devoir être compilés simultanément. Par exemple, tous les *grobs* ont une couleur, une taille, une position, etc., et toutes ces propriétés sont compilées simultanément durant l'interprétation du fichier d'entrée par LilyPond. Pour alléger

ces opérations internes, ces actions et propriétés communes sont regroupées en un objet appelé `grob-interface`. Il existe beaucoup d'autres regroupements de propriétés communes dans le genre de celui-ci, chacun portant un nom qui se termine par `interface`. En tout, on en compte plus d'une centaine. Nous verrons plus loin en quoi c'est intéressant et utile pour l'utilisateur.

Ainsi s'achève le tour des principaux termes relatifs aux objets et que nous serons amenés à utiliser dans ce chapitre.

4.1.3 Conventions de nommage des objets et propriétés

Nous avons eu un aperçu, dans Section 3.3 [Contextes et graveurs], page 59, de la façon de nommer les objets. Voici maintenant une liste de référence des types d'objets et de propriétés les plus courants, avec leurs conventions de nommage et quelques exemples de cas concrets. La lettre *A* représente n'importe quel caractère alphabétique majuscule, et les lettres *aaa* un nombre indéterminé de caractères alphabétiques minuscules. Les autres caractères sont à prendre tels qu'ils se présentent.

Type d'objet/propriété	**Convention de nommage**	**Exemples**
Contextes	Aaaa ou AaaaAaaaAaaa	Staff, GrandStaff
Objets de rendu	Aaaa ou AaaaAaaaAaaa	Slur, NoteHead
Graveurs	Aaaa_aaa_engraver	Clef_engraver, Note_heads_engraver
Interfaces	aaa-aaa-interface	grob-interface, break-aligned-interface
Propriétés de contexte	aaa ou aaaAaaaAaaa	alignAboveContext, skipBars
Propriétés d'objet de rendu	aaa ou aaa-aaa-aaa	direction, beam-thickness

Comme nous le verrons bientôt, les propriétés des différents types d'objet sont modifiées par des commandes différentes, si bien qu'il est bon de savoir reconnaître le type d'un objet en fonction du nom de ses propriétés.

4.1.4 Méthodes de retouche

La commande \override

Dans Section 3.3.4 [Modification des propriétés d'un contexte], page 63 et dans Section 3.3.5 [Ajout et suppression de graveurs], page 68, nous avons déjà rencontré les commandes `\set` et `\with`, qui servent à changer les propriétés des **contextes** et à supprimer ou ajouter des **graveurs**. Voici maintenant d'autres commandes plus importantes.

La commande pour changer les propriétés des **objets de rendu** est `\override`. Du fait que cette commande modifie en profondeur les propriétés internes de LilyPond, sa syntaxe n'est pas aussi simple que pour les commandes vues précédemment. Elle a besoin de savoir avec précision quelle est la propriété à modifier, pour quel objet et dans quel contexte, et quelle doit être sa nouvelle valeur. Voyons de quoi il retourne.

La syntaxe générale de cette commande est :

```
\override Contexte.ObjetDeRendu #'propriété-rendu = #valeur
```

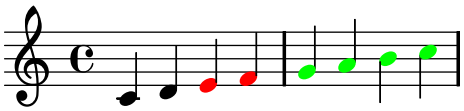
Elle attribue à la propriété appelée *propriété-rendu*, associée à l'objet *ObjetDeRendu*, appartenant lui-même au contexte *Contexte*, une valeur *valeur*.

Le contexte *Contexte* peut être omis (c'est généralement le cas) quand il n'y a pas d'ambiguïté et qu'il s'agit d'un contexte de très bas niveau, comme `Voice`, `ChordNames` ou `Lyrics`. Dans les exemples qui suivent, le contexte sera très souvent omis. Nous verrons plus tard dans quelles circonstances il doit impérativement être indiqué.

Les sections ci-dessous traitent largement des propriétés et de leurs valeurs mais, pour illustrer la mise en forme et l'utilisation de ces commandes, nous nous limiterons à n'employer que quelques propriétés et valeurs simples, facilement compréhensibles.

Nous ne parlerons dans l'immédiat ni du `#'`, qui précède toujours la propriété, ni du `#`, qui précède toujours la valeur. Ces deux éléments doivent obligatoirement être présents sous cette forme. Voici la commande la plus fréquente pour faire des retouches, et pratiquement tout le reste de ce chapitre aura pour but montrer, à travers des exemples, comment l'utiliser. L'exemple ci-dessous change la couleur des têtes de notes :

```
c4 d
\override NoteHead #'color = #red
e4 f |
\override NoteHead #'color = #green
g4 a b c |
```



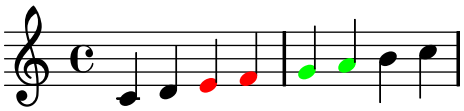
La commande \revert

Une fois qu'elle a été modifiée, la propriété conserve sa nouvelle valeur jusqu'à ce qu'elle soit à nouveau modifiée ou qu'elle rencontre la commande `\revert`. La commande `\revert` obéit à la syntaxe ci-dessous et ramène la valeur de la propriété à sa valeur d'origine. Attention : dans le cas où plusieurs `\override` ont été employés, il ne s'agit pas de la valeur précédente mais bien de la valeur par défaut.

`\revert` *`Contexte.ObjetDeRendu #'propriété-de-rendu`*

Tout comme pour la commande `\override`, la mention du *Contexte* est souvent facultative. Elle sera omise dans de nombreux exemples ci-dessous. Voici un exemple qui ramène la couleur des deux dernières notes à leur valeur par défaut :

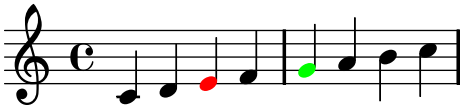
```
c4 d
\override NoteHead #'color = #red
e4 f |
\override NoteHead #'color = #green
g4 a
\revert NoteHead #'color
b4 c |
```



Le préfixe \once

Les commandes `\override` et `\set` peuvent supporter toutes les deux le préfixe `\once`. Celui-ci a pour fonction de n'appliquer la commande `\override` ou `\set` qu'à l'instant musical en cours, avant que la propriété ne reprenne sa valeur par défaut. Toujours à partir du même exemple, il est possible de ne changer la couleur que d'une seule note :

```
c4 d
\once \override NoteHead #'color = #red
e4 f |
\once \override NoteHead #'color = #green
g4 a b c |
```



La commande \overrideProperty
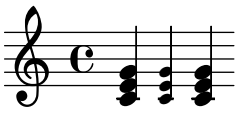
Il existe une autre forme de commande `\override`, `\overrideProperty`, qui est parfois utile. Nous la mentionnons ici par souci d'exhaustivité ; pour le détail, voir Section "Retouches complexes" dans *Extension de LilyPond*.

La commande \tweak

La dernière commande disponible pour les retouches est `\tweak`. Elle sert à changer les propriétés d'objets qui surviennent simultanément dans la musique, comme par exemple les notes d'un accord. La commande `\override` modifierait toutes les notes de l'accord, tandis que `\tweak` permet de ne modifier que l'élément suivant dans la chaîne de saisie.

Voici un exemple. Supposons que nous voulions changer la taille de la tête de note du milieu (le mi) dans un accord de do majeur. Voyons d'abord ce que donnerait `\once \override` :

```
<c e g>4
\once \override NoteHead #'font-size = #-3
<c e g>4
<c e g>4
```
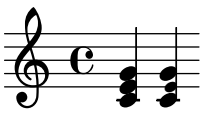


Nous voyons que `\override` modifie *toutes* les têtes de notes de l'accord, car toutes les notes de l'accord surviennent au même *instant musical* et que la fonction de `\once` est de faire porter la modification sur tous les objets du type spécifié qui surviennent au même instant musical, comme le fait la commande `\override` elle-même.

La commande `\tweak` opère différemment. Elle agit sur l'élément immédiatement suivant dans la chaîne de saisie. Elle ne fonctionne toutefois que sur des objets créés directement à partir de la chaîne de saisie, c'est-à-dire essentiellement des têtes de notes et des articulations ; des objets comme les hampes ou les altérations accidentelles sont créés ultérieurement et ne peuvent être retouchés de cette manière. En outre, quand la retouche porte sur une tête de note, celle-ci *doit* appartenir à un accord, c'est-à-dire être comprise à l'intérieur de chevrons gauche/droite. Pour retoucher une note isolée, il faut donc placer la commande `\tweak` avec la note à l'intérieur des chevrons gauche/droite.

Pour reprendre notre exemple, la taille de la note du milieu d'un accord peut être modifiée de cette manière :

```
<c e g>4
<c \tweak #'font-size #-3 e g>4
```
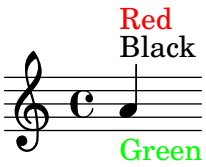


Vous noterez que la syntaxe de `\tweak` est différente de celle de la commande `\override`. Ni le contexte, ni l'objet n'ont besoin d'être spécifiés ; au contraire, cela produirait une erreur si on le faisait. Tous deux sont sous-entendus par l'élément suivant dans la chaîne de saisie. La syntaxe générale de la commande `\tweak` est donc, tout simplement :

```
\tweak #'propriété-de-rendu = #valeur
```

La commande `\tweak` est aussi utilisée quand on veut, dans une série d'articulations, n'en modifier qu'une seule. Ainsi :
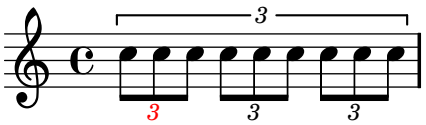
```
a4^"Black"
  -\tweak #'color #red ^"Red"
  -\tweak #'color #green _"Green"
```



Attention : la commande `\tweak` doit être précédée d'une marque d'articulation, comme si elle-même était une articulation.
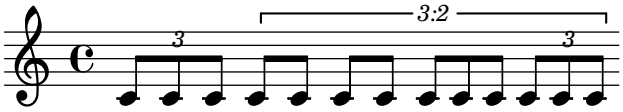
Quand plusieurs nolets sont imbriqués et commencent au même instant musical, c'est encore la commande `\tweak` qui est utilisée pour changer l'apparence de l'un d'entre eux. Dans l'exemple suivant, le long crochet de nolet et le premier des trois crochets courts commencent au même instant musical ; une commande `\override` s'appliquerait donc à la fois aux deux. En revanche, `\tweak` permet de les dissocier. La première commande `\tweak` indique que le long crochet doit être placé au-dessus des notes, et la seconde indique que le coefficient de nolet doit être imprimé en rouge sur le premier crochet de triolet court.

```
\tweak #'direction #up
\times 4/3 {
  \tweak #'color #red
  \times 2/3 { c8[ c c] }
  \times 2/3 { c8[ c c] }
  \times 2/3 { c8[ c c] }
}
```



Si les nolets imbriqués ne commencent pas au même moment, leur apparence peut alors être modifiée de la façon habituelle, avec la commande `\override` :

```
\times 2/3 { c8[ c c] }
\once \override TupletNumber
  #'text = #tuplet-number::calc-fraction-text
\times 2/3 {
  c8[ c]
  c8[ c]
  \once \override TupletNumber #'transparent = ##t
  \times 2/3 { c8[ c c] }
  \times 2/3 { c8[ c c] }
}
```



Voir aussi

Manuel de notation : Section "La commande d'affinage (*tweak*)" dans *Manuel de notation*.

4.2 Le manuel des références internes

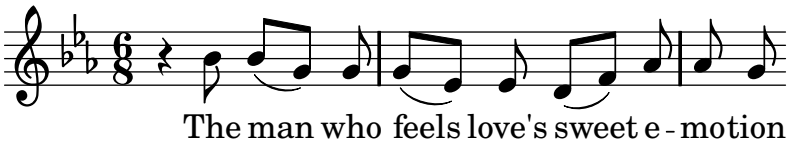
4.2.1 Propriétés des objets de rendu

Imaginons que votre partition contienne une liaison trop fine à votre goût et que vous vouliez la rendre plus épaisse. Comment vous y prendre ? Vous êtes convaincu, avec tout ce qui a été dit sur la souplesse de LilyPond, qu'une telle retouche est réalisable et vous vous dites qu'elle fera sans doute intervenir la commande `\override`. Mais existe-t-il une propriété lourde qui s'applique à une liaison et, dans l'affirmative, comment faire pour la modifier ? C'est là qu'intervient la Référence des propriétés internes. Elle contient toutes les informations dont vous avez besoin pour construire n'importe quelle commande `\override`.

Avant de nous plonger dans la Référence des propriétés internes, un mot d'avertissement. Il s'agit d'un document de **références**, de sorte qu'il ne contient pas ou peu d'explications : son but est de présenter les informations de façon précise et concise. Cela peut paraître décourageant à première vue. Pas d'inquiétude ! Les conseils et les explications fournis ici vous permettent de retrouver par vous-même les informations dans la Référence des propriétés internes. Il suffit d'un peu de pratique.

Prenons un exemple concret tiré d'un morceau de musique connu :

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8 bes[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



Admettons que nous voulions rendre les traits de liaison plus épais. Est-ce possible ? Une liaison est assurément un objet de rendu, si bien que la question est « Existe-t-il une propriété attachée aux liaisons et qui en contrôle l'épaisseur ? » Pour y répondre, nous consultons la Référence des propriétés internes (ou RPI pour faire court).

Vous trouverez sur le site de LilyPond `http://lilypond.org` la RPI correspondant à votre version du programme. Allez sur la page Documentation et cliquez sur Référence des propriétés internes. Pour l'apprentissage, mieux vaut utiliser la version HTML standard, et non la « page unique en anglais » ou le PDF. Durant la lecture des prochains paragraphes, il vous est conseillé de vous y reporter réellement afin que les explications prennent tout leur sens.

En dessous du bandeau d'en-tête figurent cinq liens. Cliquez sur le lien vers le *Backend*, où se trouvent les informations sur les objets de rendu. En dessous du titre **Backend**, choisissez alors le lien vers *Tous les objets de rendu*. La page qui s'ouvre énumère, dans l'ordre alphabétique, tous les objets utilisés dans votre version de LilyPond. Cliquez sur Liaisons (*Slurs* en anglais), et les propriétés des liaisons apparaîtront.

Il existe un autre moyen de trouver cette page, à partir du Manuel de notation. Une des pages qui traitent des liaisons contient un lien vers la Référence des propriétés internes, qui

mène directement à cette page. Mais lorsque vous connaissez le nom de l'objet à retoucher, le plus simple est de consulter la RPI.

La page de la RPI sur les liaisons commence par préciser que les objets Liaison sont créés par le graveur Slur_engraver. Vient ensuite la liste des réglages standard. Attention : ceux-ci **ne suivent pas** l'ordre alphabétique. Il faut donc les parcourir en entier pour trouver la propriété susceptible de contrôler l'épaisseur des traits de liaison.

```
thickness (nombre)
     1.2
     Épaisseur de ligne, généralement mesurée en line-thickness
```

Voilà qui semble approprié pour changer l'épaisseur (*thickness* en anglais). On apprend que la valeur de `thickness` est un simple nombre (*number*), qu'elle est par défaut définie à 1,2 et que l'unité de mesure est fixée par une autre propriété appelée `line-thickness`.

Comme il a été indiqué, on ne trouve que peu, voire pas du tout d'explication dans la RPI, mais nous en savons assez pour essayer de changer l'épaisseur de la liaison. Comme nous l'avons vu, le nom de l'objet est `Slur`, le nom de la propriété à changer est `thickness` et la nouvelle valeur sera un nombre supérieur à `1.2` si l'on veut augmenter l'épaisseur du trait.

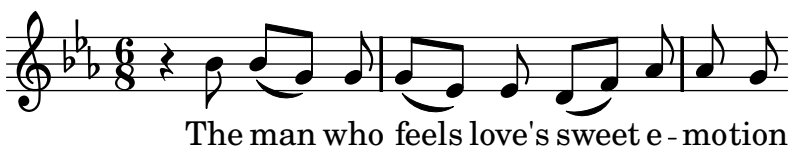
Pour construire la commande `\override`, il suffit donc de remplacer les valeurs que nous avons trouvées en guise de noms, en laissant de côté le contexte. Commençons par une valeur très élevée dans un premier temps, pour nous assurer que la commande fonctionne. Nous obtenons :

```
\override Slur #'thickness = #5.0
```

N'oublions pas le `#'` qui doit précéder le nom de la propriété et le `#` qui doit précéder la nouvelle valeur.

La dernière question est : « Où placer cette commande ? » Tant qu'on n'est pas sûr de soi, la meilleure réponse est « À l'intérieur de l'expression musicale, avant la première liaison et proche d'elle. » Essayons :

```
{
  \key es \major
  \time 6/8
  {
    % Increase thickness of all following slurs from 1.2 to 5.0
    \override Slur #'thickness = #5.0
    r4 bes8 bes[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



et nous constatons que le trait de liaison est beaucoup plus épais.

Telle est la façon normale de construire les commandes `\override`. Comme nous aurons l'occasion de le voir par la suite, le problème est parfois plus complexe. Dans l'immédiat, nous en savons assez pour construire nos propres commandes – mais il faut encore s'exercer. Les exemples suivants sont là dans cette intention.

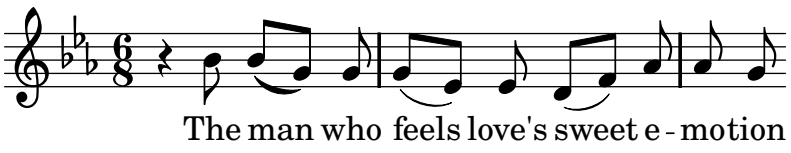
Détermination du contexte adéquat

Tout d'abord, de quoi avions-nous besoin pour préciser le contexte ? À quoi devait-il ressembler ? Gageons que les liaisons appartiennent au contexte Voix, dans la mesure où elles sont étroitement liées à une ligne mélodique, mais comment en être sûr ? Pour répondre à cette question, revenons en haut de la page de la RPI consacrée aux liaisons ; il est écrit : « Les objets Liaison sont créés par le graveur Slur_engraver ». Ainsi les liaisons seront créées dans n'importe quel contexte où se trouve le `Slur_engraver`. Suivons le lien vers la page `Slur_engraver`. Tout en bas, on lit que le `Slur_engraver` est un élément appartenant à cinq contextes Voix, dont le contexte de voix standard, `Voice`. Notre hypothèse était donc juste. Et parce que `Voice` est un contexte de très bas niveau, qu'il est activé sans ambiguïté par le fait que l'on est en train de saisir des notes, on peut ici ne pas le mentionner.

Redéfinition pour une seule occurence

Dans le dernier exemple ci-dessus, *toutes* les liaisons étaient plus épaisses. Et si on veut épaissir uniquement la première liaison ? On recourt alors à la commande `\once`. Placée juste avant la commande `\override`, elle lui indique de ne changer que la liaison commençant avec la note **juste après**. Si la note juste après n'ouvre pas une liaison, la commande sera sans aucun effet – elle ne reste pas en mémoire jusqu'à la prochaine liaison, elle est purement et simplement ignorée. Il faut donc que la commande introduite par `\once` soit insérée comme suit :

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8
    % Increase thickness of immediately following slur only
    \once \override Slur #'thickness = #5.0
    bes8[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



Alors seule la première liaison est rendue plus épaisse.
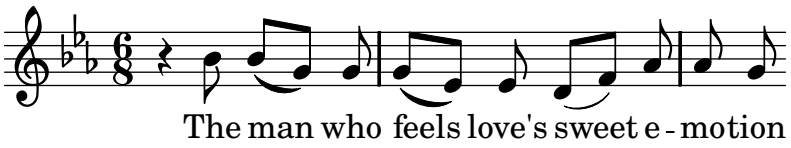
La commande `\once` peut aussi être utilisée devant la commande `\set`.

Rétablissement

Et si l'on voulait que les deux premières liaisons soient plus épaisses ? On pourrait bien sûr utiliser deux commandes, chacune précédée de `\once`, et placées juste avant la note par laquelle débute la liaison :

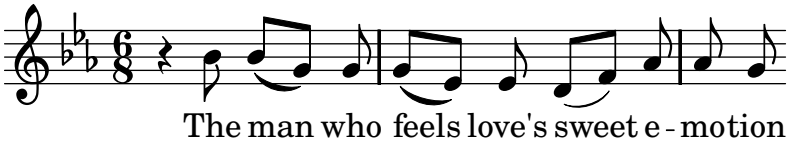
```
{
  \key es \major
  \time 6/8
  {
```

```
    r4 bes8
    % Increase thickness of immediately following slur only
    \once \override Slur #'thickness = #5.0
    bes[( g]) g |
    % Increase thickness of immediately following slur only
    \once \override Slur #'thickness = #5.0
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



mais on peut aussi, au lieu de la commande `\once`, utiliser après la seconde liaison la commande `\revert`, qui ramène la propriété `thickness` à sa valeur par défaut :

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8
    % Increase thickness of all following slurs from 1.2 to 5.0
    \override Slur #'thickness = #5.0
    bes[( g]) g |
    g8[( es]) es
    % Revert thickness of all following slurs to default of 1.2
    \revert Slur #'thickness
    d8[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



N'importe quelle propriété modifiée par `\override` peut ainsi être ramenée, grâce à la commande `\revert`, à sa valeur par défaut.

Ici s'achève notre introduction à la RPI et aux retouches simples. Vous trouverez d'autres exemples dans les prochaines sections de ce chapitre ; ils vous permettront, d'une part, d'apprendre à connaître un peu mieux la RPI et, d'autre part, de vous entraîner un peu plus à y chercher les informations. Ces exemples seront progressivement accompagnés d'explications et introduiront des termes nouveaux.

4.2.2 Propriétés listées par interface

Supposons maintenant que nous voulions imprimer des paroles en italique. Quelle formulation de la commande `\override` allons-nous utiliser ? Nous consultons en premier lieu, comme précédemment, la page de la RPI qui contient la liste « Tous les objets de rendu », et recherchons un objet qui contrôle les paroles. Nous trouvons `LyricText`, qui semble approprié. Nous cliquons dessus et nous voyons apparaître les différentes propriétés des paroles, parmi lesquelles `font-series` et `font-size`. Mais aucune ne propose l'italique. Car la mise en forme des caractères est une propriété commune à tous les objets d'écriture, si bien que, au lieu de figurer dans tous les objets de rendu, elle est regroupée avec d'autres propriétés semblables et placée dans une **Interface**, la `font-interface`.

Il nous faut donc apprendre à trouver les propriétés des interfaces et découvrir les objets qui utilisent les propriétés de ces interfaces.

Retournons à la page de la RPI qui traite des paroles (*LyricText*). En bas de la page est dressée sous forme de liens la liste des interfaces qui concernent LyricText. Cette liste comporte plusieurs entrées, dont `font-interface`. En cliquant dessus, nous voyons apparaître les différentes propriétés associées à cette interface, qui sont en même temps les propriétés de tous les objets qui s'y rapportent, parmi lesquels LyricText.

Nous avons alors sous les yeux tous les réglages des propriétés qui contrôlent les polices de caractères, et notamment `font-shape(symbole)`, où `symbole` peut prendre la valeur `upright`, `italics` ou `caps`.

Vous remarquerez que `font-series` et `font-size` figurent aussi dans la liste. La question qui ne manque pas de se poser est : « Comment se fait-il que les propriétés `font-series` et `font-size` se retrouvent à la fois dans `LyricText` et dans l'interface `font-interface` alors que ce n'est pas le cas pour `font-shape` ? » La réponse est que lorsqu'un objet `LyricText` est créé, les valeurs globales par défaut de `font-series` et `font-size` sont modifiées, mais pas celles de `font-shape`. Les possibilités de modification dans `LyricText` ne concernent donc que les valeurs à appliquer à `LyricText`. Les autres objets qui dépendent de `font-interface` fixeront leurs propriétés différemment lorsqu'ils seront créés.

Voyons maintenant si nous sommes en mesure de formuler la commande `\override` pour mettre les paroles en italique. L'objet est `LyricText`, la propriété est `font-shape` et la valeur est `italic`. Comme auparavant, nous laissons de côté le contexte.

Signalons rapidement – même si cette remarque est importante – que, puisque les valeurs de `font-shape` se présentent sous forme de symboles, elles doivent être précédées d'une simple apostrophe, `'`. C'est pour cette raison qu'il fallait une apostrophe devant `thickness` dans l'exemple précédent, et qu'il en faut une devant `font-shape`. Ce sont à chaque fois des symboles, qui sont interprétés comme tels par LilyPond. Certains symboles peuvent être des noms de propriété, comme `thickness` ou `font-shape`, d'autres sont des valeurs à attribuer aux propriétés, comme `italic`. À ne pas confondre avec les chaînes de caractères libres, qui se présentent comme `"un texte libre"` ; pour plus de détails sur les symboles et les chaînes de caractères, voir le Section "Tutoriel Scheme" dans *Extension de LilyPond*.
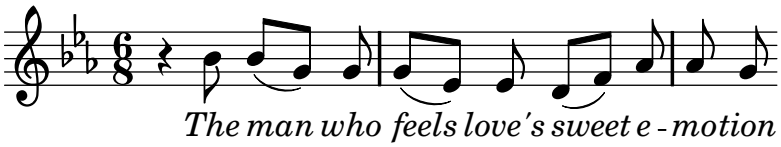
Ainsi, la commande `\override` pour mettre les paroles en italique est :

```
\override LyricText #'font-shape = #'italic
```

et doit être placée juste devant et tout près des paroles à modifier, comme ceci :

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8 bes[( g]) g |
    g8[( es]) es d[( f]) as |
```

```
    as8 g
  }
  \addlyrics {
    \override LyricText #'font-shape = #'italic
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



et voilà les paroles en italiques.

Spécification du contexte en mode lyrique

Lorsqu'il s'agit de paroles et qu'on cherche à préciser le contexte sur le modèle de ce qui a été fait précédemment, la commande échoue. Car une syllabe saisie en mode Paroles (lyricmode) se termine obligatoirement par une espace, un saut de ligne ou un nombre. Tout autre caractère compte comme un élément de la syllabe. C'est pourquoi il faut une espace ou un saut de ligne avant le `}` final, pour éviter qu'il ne soit assimilé à la dernière syllabe. De même, il faut insérer des espaces avant et après le point, « . », qui sépare le nom de contexte du nom de l'objet, faute de quoi les deux noms seront joints et l'interpréteur ne pourra pas les reconnaître. La formulation correcte est donc :

```
\override Lyrics . LyricText #'font-shape = #'italic
```

> **Note :** Dans la saisie des paroles, pensez à toujours laisser une espace entre la dernière syllabe et l'accolade fermante.

> **Note :** Lorsqu'on retouche des paroles, toujours placer des espaces autour du point qui sépare le nom de contexte du nom d'objet.

Voir aussi

Manuel d'extension : Section "Tutoriel Scheme" dans *Extension de LilyPond*.

4.2.3 Types de propriétés

Nous avons vu jusqu'à maintenant deux types de propriétés : `nombre` et `symbole`. Pour pouvoir fonctionner, la valeur associée à une propriété doit correspondre au type attendu et suivre les règles liées à ce type. Le type de propriété est toujours donné entre parenthèses après le nom de propriété dans la RPI. Voici une liste des différents types de propriétés, avec les règles qui les régissent et quelques exemples d'utilisation. Il faut, bien sûr, toujours ajouter un symbole *hash*, `#`, devant ces valeurs lors de la saisie de la commande `\override`.

Type de propriété	**Règles**	**Exemples**
Booléenne (*Boolean* en anglais)	Vrai (*true* en anglais) ou Faux (*false* en anglais), sous la forme `#t` ou `#f`	`#t`, `#f`
Dimension (en lignes de portée)	Un nombre positif décimal (en unités de lignes de portée)	`2.5`, `0.34`

Direction	Une direction valide ou son équivalent numérique (valeur décimale comprise entre -1 et 1 seulement)	`LEFT`, `CENTER`, `UP`, `1`, `-1`
Entier (*Integer* en anglais)	Un nombre entier positif	`3`, `1`
Liste	Plusieurs valeurs séparées par une espace, encadrées par des parenthèses et prédédées par une apostrophe	`'(left-edge staff-bar)`, `'(1)`, `'(1.0 0.25 0.5)`
Markup (ou étiquette)	Toute commande `\markup` valide	`\markup { \italic "cresc." }`
Durée (*Moment* en anglais)	Une durée de note construite avec la fonction make-moment	`(ly:make-moment 1 4)`, `(ly:make-moment 3 8)`
Nombre	Une valeur décimale positive ou négative	`3.5`, `-2.45`
Paire (de nombres)	Deux nombres séparées par « espace point espace », encadrés par des parenthèses et précédés par une apostrophe	`'(2 . 3.5)`, `'(0.1 . -3.2)`
Symbole	L'un des symboles autorisés pour cette propriété, précédé par une apostrophe	`'italic`, `'inside`
Inconnu (*Unknown* en anglais)	Un processus, ou `#f` pour empêcher toute action	`bend::print`, `ly:text-interface::print`, `#f`
Vecteur	Une liste de trois éléments encadrés par des parenthèses et précédés par apostrophe-hash, `'#`	`'#(#t #t #f)`

Voir aussi

Manuel d'extension : Section "Tutoriel Scheme" dans *Extension de LilyPond*.

4.3 Apparence des objets

Il est temps de mettre en pratique les notions apprises précédemment pour modifier l'allure de la partition ; les exemples qui suivent montrent l'utilisation des différentes méthodes de retouche.

4.3.1 Visibilité et couleur des objets

Dans un but pédagogique, on peut être amené à masquer certains éléments d'une partition, que les élèves doivent ensuite compléter. Imaginons, par exemple, un exercice dans lequel il faudrait rétablir les barres de mesure dans un morceau de musique. En temps normal, les barres de mesure s'insèrent automatiquement. Comment faire pour les effacer de la partition ?

Avant de nous y attaquer, souvenons-nous que les propriétés d'objets sont parfois groupées dans ce qu'on appelle des *interfaces* – voir Section 4.2.2 [Propriétés listées par interface], page 97. Cela permet de rapprocher toutes les propriétés susceptibles d'être utilisées ensemble pour modifier un objet graphique – si l'une d'elles est choisie pour un objet, elle s'appliquera à tous les autres. Certains objets tirent alors leurs propriétés de telle ou telle interface, d'autres objets de telle ou telle autre interface. La liste des interfaces qui contiennent les propriétés liées à un objet graphique (*grob*) figure dans la RPI, en bas de la page de description du *grob* ; pour voir ces propriétés, il faut aller voir ces interfaces.
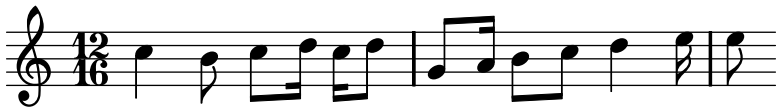
Nous avons vu, dans Section 4.2.1 [Propriétés des objets de rendu], page 93, comment trouver les informations sur les *grobs*. Nous procédons de la même manière et consultons la RPI pour connaître l'objet chargé d'imprimer les barres de mesure. En cliquant sur *Backend* puis sur *Tous les objets de rendu*, nous trouvons un objet appelé `BarLine`. Parmi ses propriétés, deux d'entre elles déterminent son aspect visuel : `break-visibility` et `stencil`. L'objet `BarLine`

est également lié à plusieurs interfaces, dont la `grob-interface` où figurent les propriétés `transparent` et `color`. Toutes peuvent modifier l'aspect visuel des barres de mesure – et de beaucoup d'autres objets, bien sûr. Examinons chacune d'elles tour à tour.

stencil
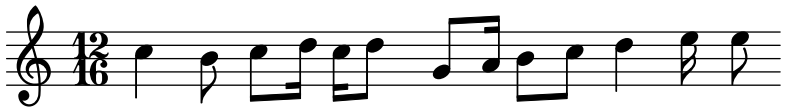
Cette propriété contrôle l'apparence des barres de mesure en précisant le type de symbole (glyphe) à imprimer. Comme pour de nombreuses autres propriétés, on peut lui indiquer de ne rien imprimer en lui attribuant la valeur `#f`. Essayons en laissant de côté, une fois encore, le contexte concerné (`Voice` en l'occurrence) :

```
{
  \time 12/16
  \override BarLine #'stencil = ##f
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



Les barres de mesure sont encore là ! Pourquoi ? Retournons à la RPI et regardons de nouveau la page qui traite des propriétés de BarLine. En haut de la page, il est précisé que « Les objets BarLine sont créés par le graveur Bar_engraver ». Allons à la page de `Bar_engraver`. Tout en bas se trouve la liste des contextes dans lesquels fonctionne ce graveur. Tous sont du type `Staff`, de sorte que, si la commande `\override` n'a pas fonctionné comme prévu, c'est parce que `Barline` n'appartient pas au contexte par défaut, `Voice`. Si le contexte spécifié est erroné, la commande ne fonctionne pas. Cela n'entraîne pas de message d'erreur, et rien n'apparaît dans le fichier log. Essayons de corriger en mentionnant le bon contexte :

```
{
  \time 12/16
  \override Staff.BarLine #'stencil = ##f
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
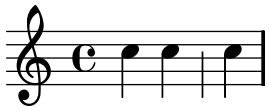


Cette fois, les barres de mesure ont disparu.

Vous remarquerez que l'affectation de la valeur `#f` à la propriété `stencil` déclechera une erreur dès lors que l'objet en question se doit d'avoir des dimensions pour les nécessités du traitement. Ce sera le cas, par exemple, si vous effacez le `stencil` d'un objet `NoteHead`. Il vaut mieux, en pareil caas, utiliser la fonction `point-stencil` qui, quant à elle, attribue à l'objet une taille à zero :

```
{
  c4 c
  \once \override NoteHead #'stencil = #point-stencil
  c4 c
```
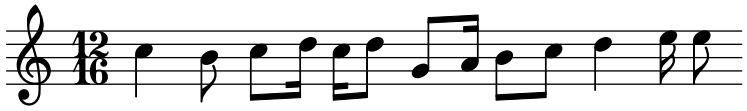
```
}
```



visibilité des barres (break-visibility)

La RPI mentionne, à la page sur `BarLine`, que la propriété `break-visibility` attend comme argument un vecteur de trois boléens. Ceux-ci indiquent respectivement si les barres de mesure doivent être imprimées ou non à la fin de la ligne, à l'intérieur de la ligne et au début de la ligne. Dans notre cas, nous voulons que toutes les barres soient supprimées, si bien que la valeur dont nous avons besoin est `'#(#f #f #f)`. Essayons, sans oublier d'ajouter le contexte `Staff`. Vous remarquerez que, en plus de cette valeur, nous ajoutons `#'#` devant la parenthèse ouvrante. Le `'#` est nécessaire car il fait partie intégrante de la valeur contenant un vecteur, et le premier `#` est là, comme toujours avec la commande `\override`, pour introduire la valeur elle-même.

```
{
  \time 12/16
  \override Staff.BarLine #'break-visibility = #'#(#f #f #f)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
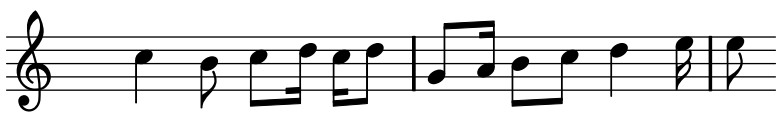


Comme on peut le constater, cette solution-là aussi supprime les barres de mesure.

transparence

La RPI mentionne, à la page sur `grob-interface`, que la propriété `transparent` attend comme argument un boléen. Il faudrait donc mettre `#t` pour rendre l'objet transparent. Dans cet exemple, essayons de rendre transparente la métrique (*time signature* en anglais) plutôt que les barres de mesure. Pour cela, il nous faut trouver le nom du *grob* chargé de l'indication de mesure. De retour sur la page « Tous les objets de rendu » de la RPI, nous cherchons les propriétés de l'objet `TimeSignature`. Celui-ci est géré par le graveur `Time_signature_engraver` qui, comme vous pouvez le constater, appartient au contexte `Staff` et peut se rattacher à la `grob-interface`. Dans ces conditions, la commande pour rendre la métrique transparente est :
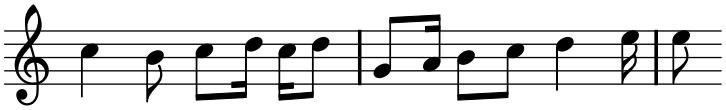
```
{
  \time 12/16
  \override Staff.TimeSignature #'transparent = ##t
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



La métrique a bien disparu mais la commande a laissé un blanc en lieu et place du chiffrage. Ce peut être souhaitable dans le cadre d'un exercice, afin que les élèves aient la place pour

compléter, mais dans d'autres circonstances, ce peut être gênant. Pour y remédier, attribuons plutôt au stencil des métriques la valeur `#f` :

```
{
  \time 12/16
  \override Staff.TimeSignature #'stencil = ##f
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
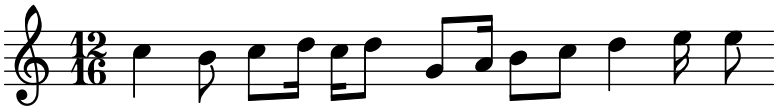


La différence est flagrante : le fait d'attribuer au stencil la valeur `#f` supprime totalement l'objet, tandis que le fait de le rendre `transparent` le laisse en place, mais de façon invisible.

couleur

Essayons enfin de rendre les barres de mesure invisibles en les colorant en blanc. La difficulté est de savoir si les barres blanches vont couper ou non les lignes de la portée aux endroits où elles se croisent. Vous verrez dans les exemples ci-dessous que cela peut se produire, sans qu'on le sache à l'avance. Les explications de ce phénomène et les solutions pour y remédier sont exposées dans Section "Blanchiment des objets" dans *Manuel de notation*. Pour le moment, acceptons cet inconvénient et concentrons-nous sur l'apprentissage de la gestion des couleurs.

La `grob-interface` indique que la valeur de la propriété `color` est une liste, sans plus d'explication. En fait, cette liste est une liste de valeurs en unités internes ; pour éviter d'avoir à chercher ce qu'il faut y mettre, il existe différents moyens d'indiquer la couleur. Le premier moyen consiste à utiliser l'une des couleurs *normales* de la première Section "Liste des couleurs" dans *Manuel de notation*. Pour mettre les barres de mesure en blanc, on écrit :
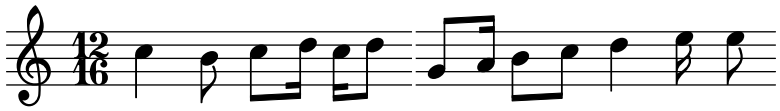
```
{
  \time 12/16
  \override Staff.BarLine #'color = #white
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



et nous constatons que les barres de mesure sont une fois de plus invisibles. Attention : aucune apostrophe ne précède `white` – il ne s'agit pas d'un symbole mais d'une **fonction**. Quand on l'invoque, elle fournit une liste de valeurs internes requises pour changer la couleur en blanc. Les autres couleurs aussi, dans la « liste normale », sont des fonctions. Pour en être certain, vous pouvez faire l'essai en choisissant une autre fonction de la liste en guise de couleur.

Le deuxième moyen de changer la couleur consiste à utiliser la deuxième Section "Liste des couleurs" dans *Manuel de notation*, dite noms de couleurs X11. Ceux-ci doivent obligatoirement être précédés d'une autre fonction, qui convertit les noms de couleurs X11 en une liste de valeurs internes, `x11-color`, comme ceci :
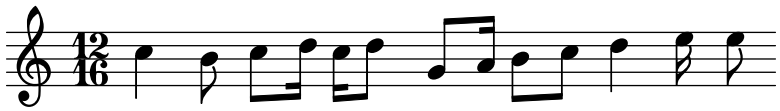
```
{
  \time 12/16
  \override Staff.BarLine #'color = #(x11-color 'white)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



Vous noterez que, dans ce cas, la fonction `x11-color` admet un symbole comme argument ; il faut donc placer une apostrophe devant le symbole et insérer les deux à l'intérieur de parenthèses.
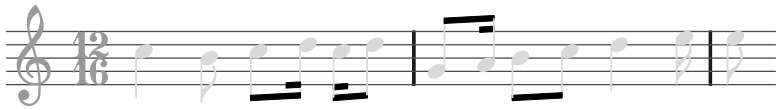
Il existe une troisième fonction, écrite pour convertir les valeurs RVB en couleurs internes – la fonction `rgb-color`. Elle comporte trois arguments, donnant respectivement l'intensité du rouge, du vert et du bleu. Ces arguments prennent des valeurs comprises entre 0 et 1. Ainsi, pour choisir la couleur rouge, la valeur serait `(rgb-color 1 0 0)` ; pour le blanc, ce serait `(rgb-color 1 1 1)` :

```
{
  \time 12/16
  \override Staff.BarLine #'color = #(rgb-color 1 1 1)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



Enfin, il existe une échelle de gris parmi les possibilités de couleurs X11. Elle va du noir (`'grey0`) au blanc (`'grey100`), avec un pas de 1. Essayons de l'utiliser en attribuant à tous les objets de notre exemple différentes nuances de gris :

```
{
  \time 12/16
  \override Staff.StaffSymbol   #'color = #(x11-color 'grey30)
  \override Staff.TimeSignature #'color = #(x11-color 'grey60)
  \override Staff.Clef          #'color = #(x11-color 'grey60)
  \override Voice.NoteHead      #'color = #(x11-color 'grey85)
  \override Voice.Stem          #'color = #(x11-color 'grey85)
  \override Staff.BarLine       #'color = #(x11-color 'grey10)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
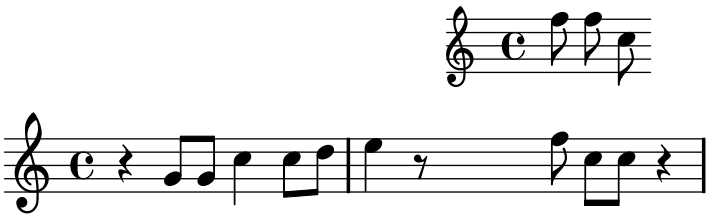


Vous remarquerez le contexte associé à chacun des objets. Une erreur sur ce point empêcherait la commande de fonctionner. Souvenez-vous que le contexte est celui dans lequel est placé le graveur approprié. Pour chaque graveur, on peut trouver son contexte par défaut en partant de l'objet lui-même, puis en cherchant le graveur qui le produit ; la page du graveur dans la RPI nous indique alors le contexte dans lequel le graveur se trouve normalement.

4.3.2 Taille des objets

Pour commencer, reprenons l'exemple qui se trouvait dans Section 3.1.3 [Expressions musicales imbriquées], page 46, qui montrait comment créer une nouvelle portée temporaire, du type Section "ossia" dans *Glossaire*.
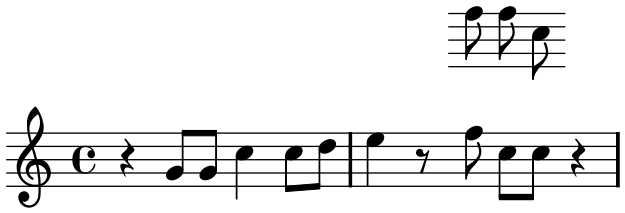
```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main" }
      { f8 f c }
    >>
    r4 |
  }
}
```



Normalement, les ossias ne comportent ni clef ni indication de mesure, et elles sont imprimées légèrement plus petit que la portée principale. Nous avons déjà appris à enlever la clef et la métrique – il suffit de régler le stencil de chacun sur `#f`, comme ceci :

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
      }
      {
        \override Staff.Clef #'stencil = ##f
        \override Staff.TimeSignature #'stencil = ##f
        { f8 f c }
      }
    >>
    r4 |
  }
```

```
}
```



La paire d'accolades ajoutée après la clause `\with` est nécessaire pour être sûr que les retouches (`\override`) ainsi que la musique qui se trouvent à l'intérieur soient bien appliquées à la portée d'ossia.
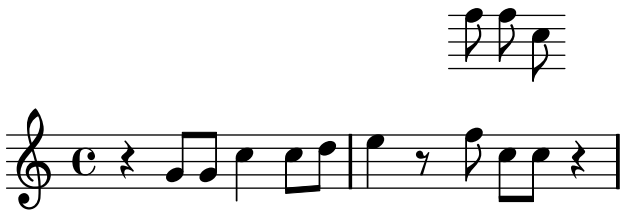
Mais alors, quelle différence y a-t-il à modifier le contexte de portée au moyen de `\with` ou à modifier les stencils de la clef et de la métrique avec `\override` ? La principale différence est que les changements opérés dans une clause `\with` sont réalisés au moment où le contexte est créé et restent par la suite les valeurs **par défaut** aussi longtemps que ce contexte existe, tandis que les commandes `\set` ou `\override` insérées dans la musique sont dynamiques – elles provoquent des changements synchronisés avec un point particulier de la musique. Si les changements sont annulés ou désactivés par `\unset` ou `\revert`, les réglages reprennent leurs valeurs par défaut, c'est-à-dire celles qui ont été fixées dans la clause `\with`, ou, en l'absence de celle-ci, les valeurs par défaut normales.

Certaines propriétés de contexte ne peuvent être modifiées que dans une clause `\with`. Il s'agit des propriétés qu'on ne peut évidemment plus changer après que le contexte a été créé. C'est le cas de `alignAboveContext` et de son pendant, `alignBelowContext` – une fois que la portée a été créée, son alignement est décidé et cela n'aurait aucun sens de vouloir le modifier par la suite.

Dans une clause `\with`, on peut aussi régler les valeurs par défaut des propriétés d'un objet. Il suffit d'utiliser la commande `\override` normale, sans s'occuper du nom de contexte puisqu'il ne fait pas de doute qu'il s'agit du contexte en cours de modification par la clause `\with`. Il se produirait même une erreur si le contexte était précisé.

Remplaçons donc l'exemple ci-dessus par celui-ci :

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
        % Don't print clefs in this staff
        \override Clef #'stencil = ##f
        % Don't print time signatures in this staff
        \override TimeSignature #'stencil = ##f
      }
      { f8 f c }
    >>
    r4 |
  }
}
```



Venons-en finalement au changement de taille des objets.

Certains objets sont créés comme des glyphes choisis dans une police de caractères. C'est le cas des têtes de notes, des altérations, des *markup*, des clefs, des métriques, des nuances et des paroles. Pour changer leur taille, on modifie la propriété `font-size`, comme nous le verrons rapidement. D'autres objets, comme les liaisons de phrasé ou de prolongation – en général les objets étendus – sont dessinés à la demande, si bien qu'aucune `font-size` ne leur est associée. Ces objets tirent généralement leur dimension des objets auxquels ils sont rattachés, de sorte qu'on ne doit pas avoir à les redimensionner à la main. D'autres propriétés, comme la hauteur des hampes et des barres de mesure, l'épaisseur des ligatures et d'autres lignes, et l'écartement des lignes de portée, doivent encore être modifiées de façon particulière.

Si l'on revient à l'exemple d'ossia, commençons par changer la taille de police. Nous pouvons employer deux méthodes. Soit nous changeons la taille de police de chaque type d'objet avec des commandes comme celle-ci pour les têtes de notes (`NoteHead`) :

```
\override NoteHead #'font-size = #-2
```
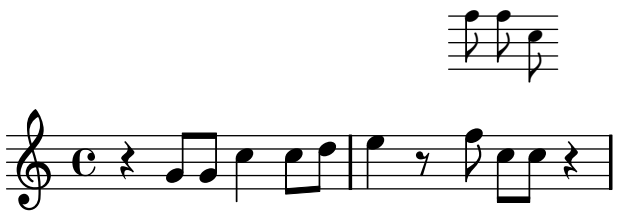
soit nous changeons la taille de toutes les polices à la fois grâce à la propriété `fontSize`, en utilisant `\set` ou en l'insérant dans une clause `\with` (mais alors sans le `\set`).

```
\set fontSize = #-2
```

Chacune de ces méthodes réduira la taille de police de deux points par rapport à sa valeur précédente, sachant que chaque point réduit ou augmente la taille d'environ 12 %.

Essayons sur l'exemple d'ossia :

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
        \override Clef #'stencil = ##f
        \override TimeSignature #'stencil = ##f
        % Reduce all font sizes by ~24%
        fontSize = #-2
      }
      { f8 f c }
    >>
    r4 |
  }
}
```
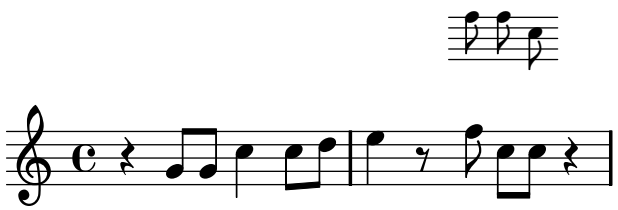


Ce n'est pas encore parfait. Les têtes de notes et les crochets sont plus petits mais, proportionnellement, les hampes sont trop longues et les lignes de la portée trop espacées. Il faut donc les réduire dans les mêmes proportions que les polices de caractères. La prochaine sous-section montrera comment faire.

4.3.3 Longueur et épaisseur des objets

Dans LilyPond, les écartements et longueurs sont généralement mesurés en « intervalles de lignes » (*staff-spaces* en anglais), c'est-à-dire l'écartement qui sépare deux lignes adjacentes dans la portée – plus rarement, il est question de demi-intervalles de lignes. Les propriétés d'épaisseur (`thickness`), quant à elles, sont généralement mesurées en unités d'une propriété interne appelée « épaisseur de ligne » (`line-thickness`). Par exemple, les lignes de crescendo/decrescendo présentent par défaut une épaisseur de 1 unité de `line-thickness`, alors que l'épaisseur d'une hampe est de 1,3. Toutefois, certaines propriétés d'épaisseur sont différentes : par exemple, l'épaisseur des ligature – `beam-thickness` – se mesure en espaces de portée.

Dans ces conditions, comment ajuster les longueurs à la taille des polices de caractères ? La solution consiste à utiliser une fonction spéciale appelée `magstep` – pseudo facteur de zoom –, créée précisément dans ce but. Elle comporte un argument, le changement de taille de police (#-2 dans l'exemple précédent), à partir duquel elle applique un facteur de mise à l'échelle qui réduit, ou augmente, les objets en question. Voici comment elle s'utilise :

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
        \override Clef #'stencil = ##f
        \override TimeSignature #'stencil = ##f
        fontSize = #-2
        % Reduce stem length and line spacing to match
        \override StaffSymbol #'staff-space = #(magstep -2)
      }
      { f8 f c }
    >>
    r4 |
  }
}
```



Puisque la longueur des hampes et plusieurs autres propriétés de longueur sont calculées par rapport à la valeur de la propriété `staff-space`, elles sont automatiquement mises à l'échelle. Vous remarquerez que cela n'affecte que la dimension verticale de l'ossia – la dimension horizontale étant déterminée par les objets de la portée principale de façon à rester synchronisée vis-à-vis d'elle, elle n'est pas affectée par tous ces changements de taille. Bien sûr, si l'échelle de

toute la portée principale était modifiée, tout le placement horizontal s'en trouverait affecté. Il en sera question plus bas dans cette section.

Voilà qui complète la création d'une ossia. Les taille et longueur de tous les objets peuvent être modifiées de manière analogue.

Pour de petits changements d'échelle, comme dans l'exemple ci-dessus, il n'est généralement pas utile d'ajuster l'épaisseur des différentes lignes telles que les barres de mesure, les ligatures, les soufflets de crescendo/decrescendo, les liaisons, etc. Si l'épaisseur d'un objet en particulier doit être ajustée, le mieux est de modifier sa propriété `thickness`. Nous avons vu plus haut, dans Section 4.2.1 [Propriétés des objets de rendu], page 93, un exemple de modification de l'épaisseur des liaisons. L'épaisseur de tous les objets tracés (c'est-à-dire ceux qui ne proviennent pas d'une police de caractère) peut être changée de la même manière.

4.4 Positionnement des objets

4.4.1 Comportement automatique

Dans la notation musicale, il y a des objets qui appartiennent à la portée et d'autres qui sont placés à l'extérieur de la portée. On les appelle respectivement les « objets de la portée » (*within-staff objects* en anglais) et les « objets extérieurs à la portée » (*outside-staff objects* en anglais).
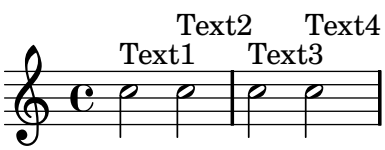
Les objets de la portée sont ceux qui sont placés sur la portée – les têtes de notes et les hampes, les altérations, etc. Leur position est généralement déterminée par la musique elle-même – ils sont placés verticalement sur des lignes spécifiques ou sont liés à d'autres objets placés de cette manière. Normalement, les collisions entre les têtes et queues de notes et les altérations dans des accords proches sont évitées automatiquement. Comme nous le verrons rapidement, il existe des commandes et des possibilités de retouche qui permettent de modifier ce comportement automatique.

Parmi les objets extérieurs à la portée, on compte des éléments comme les marques de reprise, les indications de texte ou de nuances. Dans LilyPond, la règle est de placer verticalement ces objets extérieurs à la portée le plus près possible de la portée, tout en évitant la collision avec d'autres objets. LilyPond utilise la propriété `outside-staff-priority` pour déterminer l'ordre selon lequel placer ces objets, de la manière suivante :

D'abord, LilyPond dresse la liste de tous les objets extérieurs à la portée. Puis ceux-ci sont classés suivant leur `outside-staff-priority`. Enfin, ils sont pris un par un, en commençant par les objets avec la `outside-staff-priority` la plus basse, et placés de façon à ne pas entrer en collision avec d'autres objets déjà placés. Cela signifie que, si deux *grobs* extérieurs à la portée doivent occuper la même place, c'est celui qui a la `outside-staff-priority` la plus basse qui est placé le plus près de la portée. Et si deux objets ont la même `outside-staff-priority`, le premier rencontré sera placé le plus près de la portée.

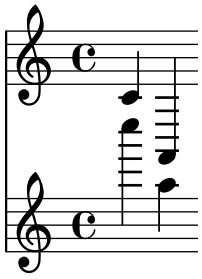
Dans l'exemple suivant, tous les *markup* ont la même priorité, dans la mesure où rien n'est indiqué explicitement. Vous remarquerez que `Text3` est également positionné près de la portée, juste en-dessous de `Text2`.

```
c2^"Text1"
c2^"Text2" |
c2^"Text3"
c2^"Text4" |
```



Les portées aussi sont positionnées, par défaut, le plus près possible les unes des autres, en ménageant tout de même une certaine séparation. Si des notes se rapprochent nettement d'une portée adjacente, elles ne forceront les portées à s'écarter que s'il y a un risque de chevauchement.

```
<<
  \new Staff {
    \relative c' { c4 a, }
  }
  \new Staff {
    \relative c'''' { c4 a, }
  }
>>
```
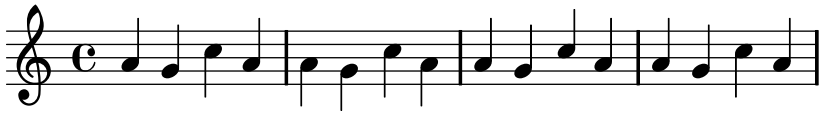


4.4.2 Objets inclus dans la portée

Nous avons vu que les commandes `\voiceXXX` jouent sur la direction des liaisons, des doigtés et sur toute autre chose liée à l'orientation des hampes. Ces commandes sont essentielles dans la musique polyphonique pour distinguer des lignes mélodiques entremêlées. Mais il arrive qu'on ait besoin de modifier ce comportement automatique. On peut le faire pour toutes les parties de la musique ou juste pour une note. La propriété qui contrôle ce comportement est la propriété `direction` de chaque objet. Expliquons d'abord ce qu'elle fait, puis nous présenterons un certain nombre de commandes déjà prêtes qui évitent, pour les modifications les plus courantes, d'avoir à encoder les retouches.

Certains objets comme les traits des liaisons se recourbent tantôt vers le haut, tantôt vers le bas ; d'autres encore, comme les hampes et les crochets, peuvent se décaler vers la gauche ou vers la droite selon qu'ils pointent vers le haut ou vers le bas. Ce comportement est géré automatiquement dès lors que `direction` est activé.

L'exemple ci-dessous montre dans la première mesure le comportement par défaut des hampes – celles des notes les plus hautes pointant vers le bas et celles des notes les plus basses pointant vers le haut ; viennent ensuite quatre notes avec les hampes forcées vers le bas, puis quatre autres avec les hampes forcées vers le haut, et pour finir quatre notes de nouveau avec le comportement par défaut.

```
a4 g c a |
\override Stem #'direction = #DOWN
a4 g c a |
\override Stem #'direction = #UP
a4 g c a |
\revert Stem #'direction
a4 g c a |
```



Nous utilisons ici les directions `DOWN` et `UP`. Elles correspondent respectivement aux valeurs `-1` et `+1`, que l'on peut utiliser à la place. La valeur `0` peut aussi être utilisée dans certains cas. Elle est interprétée comme un `UP` pour les hampes, et comme un « centré » pour d'autres objets. Il existe une direction, `CENTER`, qui correspond à la valeur `0`.

Quoi qu'il en soit, ces retouches manuelles sont rarement utilisées car il existe des équivalents sous forme de commandes prédéfinies. Voici un tableau des plus courantes. Lorsque ce n'est pas évident, leur signification est précisée.

Bas/Gauche	**Haut/Droite**	**Annulation**	**Effet**
`\arpeggioArrowDown`	`\arpeggioArrowUp`	`\arpeggioNormal`	Flèche en bas, en haut, ou pas de flèche
`\dotsDown`	`\dotsUp`	`\dotsNeutral`	Déplacement des points pour éviter les lignes de portée
`\dynamicDown`	`\dynamicUp`	`\dynamicNeutral`	
`\phrasingSlurDown`	`\phrasingSlurUp`	`\phrasingSlurNeutral`	Attention : à distinguer des commandes de liaison ci-dessous
`\slurDown`	`\slurUp`	`\slurNeutral`	
`\stemDown`	`\stemUp`	`\stemNeutral`	
`\textSpannerDown`	`\textSpannerUp`	`\textSpannerNeutral`	Le texte saisi en tant qu'extension est au-dessous/au-dessus de la portée
`\tieDown`	`\tieUp`	`\tieNeutral`	
`\tupletDown`	`\tupletUp`	`\tupletNeutral`	Les nolets sont au-dessous/au-dessus des notes

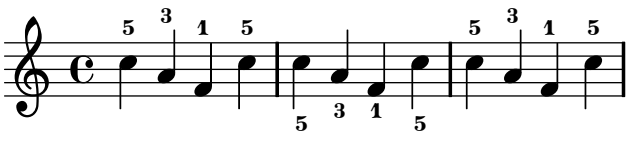
Attention : ces commandes prédéfinies **ne doivent pas** être précédées de `\once`. Pour limiter l'effet à une seule note, il faut soit utiliser la commande équivalente `\once \override`, soit utiliser la commande prédéfinie, suivie, après la note à modifier, de la commande `\xxxNeutral` correspondante.

Doigtés

Le placement des doigtés sur des notes simples peut aussi être contrôlé par la propriété `direction`, mais le changement de `direction` n'a pas d'effet sur les accords. Comme nous le verrons, il existe des commandes qui permettent de contrôler le doigté sur chaque note d'un accord, en plaçant l'indication de doigté au-dessus, en dessous, à gauche ou à droite de chaque note.

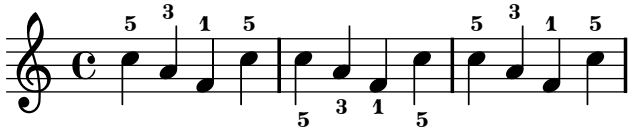
Tout d'abord, voici l'effet de `direction` sur le doigté lié à une note simple. La première mesure montre le comportement par défaut, et les deux suivantes montrent l'effet lorsqu'on indique `DOWN` et `UP` :

```
c4-5 a-3 f-1 c'-5 |
\override Fingering #'direction = #DOWN
c4-5 a-3 f-1 c'-5 |
\override Fingering #'direction = #UP
c4-5 a-3 f-1 c'-5 |
```



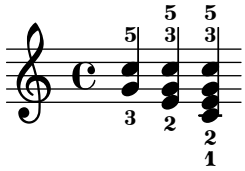
Le fait d'intervenir sur la propriété `direction` n'est sûrement pas la façon la plus simple de placer manuellement les doigtés au-dessus ou en dessous des notes ; mieux vaut utiliser `_` ou `^` devant le chiffre de doigté plutôt que `-`. Voici ce que donne l'exemple précédent avec cette méthode :

```
c4-5 a-3 f-1 c'-5 |
c4_5 a_3 f_1 c'_5 |
c4^5 a^3 f^1 c'^5 |
```



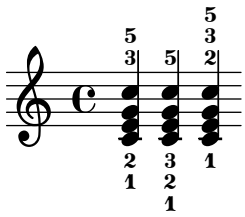
La propriété `direction` ne fonctionne pas pour les accords alors que les préfixes de direction, `_` et `^`, fonctionnent. Par défaut, le doigté est placé automatiquement à la fois au-dessus et au-dessous des notes d'un accord, comme ceci :

```
<c-5 g-3>4
<c-5 g-3 e-2>4
<c-5 g-3 e-2 c-1>4
```



mais il est possible de forcer manuellement vers le haut ou vers le bas le placement de tous ou certains chiffres de doigté, comme ceci :

```
<c-5 g-3 e-2 c-1>4
<c^5 g_3 e_2 c_1>4
<c^5 g^3 e^2 c_1>4
```



On peut aller encore plus loin dans le positionnement des doigtés pour chacune des notes d'un accord grâce à la commande `\set fingeringOrientations`. La syntaxe de cette commande est :

```
\set fingeringOrientations = #'([up] [left/right] [down])
```

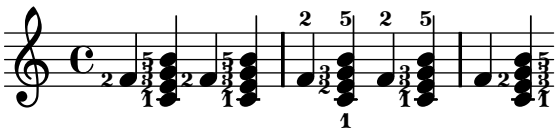
On utilise `\set` car `fingeringOrientations` est une propriété du contexte `Voice`, créée et utilisée par le graveur `New_fingering_engraver`.

On peut attribuer à cette propriété une liste composée de une à trois valeurs. Celles-ci déterminent si l'indication de doigté doit être placée au-dessus (lorsque `up` apparaît dans la liste), au-dessous (lorsque `down` apparaît), à gauche (lorsque `left` apparaît) ou à droite (lorsque `right` apparaît). En revanche, si une valeur n'est pas dans la liste, aucun doigté n'ira à cet emplacement. LilyPond garde ces contraintes en mémoire et recherche le meilleur emplacement pour le doigté des notes des accords suivants. Vous remarquerez que `left` et `right` s'excluent l'un l'autre – l'indication de doigté ne peut être placée que d'un côté ou de l'autre, pas des deux.

> **Note :** Pour contrôler à l'aide de cette commande le placement du doigté sur une note simple, il faut la saisir comme un accord composé d'une note unique, en l'encadrant de chevrons.

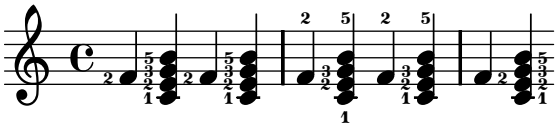
Voici quelques exemples :

```
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(up left down)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(up left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(right)
<f-2>4
<c-1 e-2 g-3 b-5>4
```



Si les indications de doigtés paraissent un peu serrées, on peut toujours réduire la taille de police (`font-size`). La valeur par défaut donnée dans la RPI à la page de l'objet `Fingering` étant `-5`, essayons `-7` :

```
\override Fingering #'font-size = #-7
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(up left down)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(up left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(right)
<f-2>4
<c-1 e-2 g-3 b-5>4
```



4.4.3 Objets hors de la portée

Les objets extérieurs à la portée sont placés automatiquement de façon à éviter les collisions. Les objets avec la plus petite valeur de la propriété `outside-staff-priority` sont placés au plus

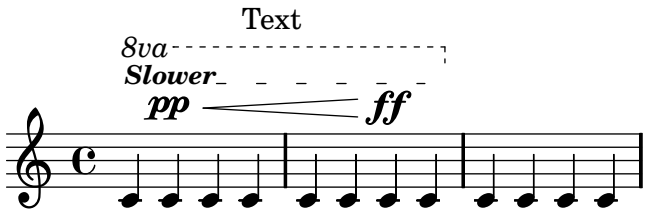
près de la portée, tandis que les autres sont écartés autant qu'il faut pour éviter les collisions. La `outside-staff-priority` est définie dans la `grob-interface` ; elle est donc une propriété commune à tous les objets de rendu. Par défaut, elle est réglée sur `#f` pour tous les objets de la portée, et porte une valeur numérique appropriée à chacun des objets extérieurs à la portée, à mesure qu'ils sont créés. Le tableau suivant montre la valeur numérique par défaut pour quelques-uns des objets extérieurs à la portée les plus courants qui sont placés, par défaut, dans les contextes `Staff` ou `Voice`.

Vous noterez au passage certaines particularités quant au nom des objets. En effet, des objets à extension sont automatiquement créés dans le but de pouvoir contrôler le positionnement vertical des extrémités d'un *grob* qui commencerait et se terminerait à des instants musicaux différents. C'est la raison pour laquelle modifier le `outside-staff-priority` du *grob* sous-jacent n'aura aucun effet. C'est par exemple le cas lorsque vous modifiez l'alignement d'un objet `Hairpin` à l'aide de `outside-staff-priority` ; puique le soufflet est associé à un objet `DynamicLineSpanner`, c'est sur celui-ci que doit porter l'effet de `outside-staff-priority`. L'instruction dérogatoire se place au début du bandeau qui constitue une ligne de base susceptible de contenir plusieurs soufflets ou indications de nuance.

Objet de rendu	**Priorité**	**Contrôle la position de :**
`MultiMeasureRestText`	450	Texte sur les silences qui couvrent des mesures entières
`TextScript`	450	Texte des *markup* (ou étiquettes)
`OttavaBracket`	400	Indication d'octaviation
`TextSpanner`	350	Bandeau ou extension de texte
`DynamicLineSpanner`	250	Toutes les marques de nuances
`VoltaBracketSpanner`	100	Bandeau de répétition
`TrillSpanner`	50	Bandeau de trille

Voici un exemple qui montre le placement par défaut de certains d'entre eux.

```
% Set details for later Text Spanner
\override TextSpanner #'(bound-details left text)
    = \markup { \small \bold Slower }
% Place dynamics above staff
\dynamicUp
% Start Ottava Bracket
\ottava #1
c'4 \startTextSpan
% Add Dynamic Text and hairpin
c4\pp\<
c4
% Add Text Script
c4^Text |
c4 c
% Add Dynamic Text and terminate hairpin
c4\ff c \stopTextSpan |
% Stop Ottava Bracket
\ottava #0
c,4 c c c |
```



Cet exemple montre comment créer des extensions de texte (*Text Spanners* en anglais) – texte avec des longues lignes au-dessus d'un passage musical. L'extension s'étend depuis la commande `\startTextSpan` jusqu'à la commande `\stopTextSpan` et le format de texte est défini par la commande `\override TextSpanner`. Pour de plus amples détails, voir Section "Indication textuelle avec ligne d'extension" dans *Manuel de notation*.

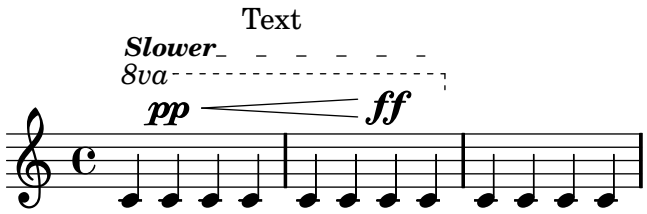
Il montre aussi comment créer des marques d'octaviation.

Vous aurez remarqué que les numéros de mesure, l'indication métronomique et les marques de répétition n'apparaissent pas. Par défaut, ils sont créés dans le contexte `Score` et leur `outside-staff-priority` est ignorée face aux objets qui sont créés dans le contexte `Staff`. Si vous voulez faire apparaître les numéros de mesure, l'indication métronomique ou les marques de répétition conformément à la valeur de leur `outside-staff-priority`, il vous faudra retirer respectivement le `Bar_number_engraver`, le `Metronome_mark_engraver` ou le `Mark_engraver` du contexte `Score` et les placer en tête du contexte `Staff`. Après quoi ces marques porteront les valeurs de `outside-staff-priority` par défaut suivantes :

Objet de rendu	**Priorité**
`RehearsalMark`	1500
`MetronomeMark`	1000
`BarNumber`	100
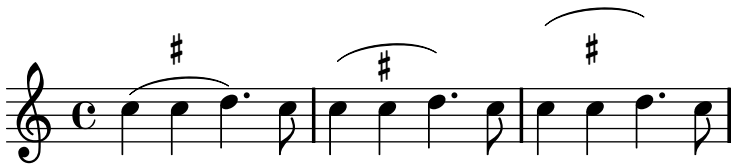
Si les valeurs de `outside-staff-priority` par défaut ne donnent pas les résultats escomptés, il suffit de modifier la priorité de l'un de ces objets. Supposons que vous vouliez placer l'indication d'octaviation sous le bandeau de texte, dans l'exemple précédent. Tout ce que nous devons faire, c'est regarder la priorité de `OttavaBracket` dans la RPI ou dans le tableau plus haut, et la ramener à une valeur plus basse que celle de `TextSpanner`, en gardant à l'esprit que `OttavaBracket` est créé dans le contexte `Staff` :

```
% Set details for later Text Spanner
\override TextSpanner #'(bound-details left text)
    = \markup { \small \bold Slower }
% Place dynamics above staff
\dynamicUp
% Place following Ottava Bracket below Text Spanners
\once \override Staff.OttavaBracket #'outside-staff-priority = #340
% Start Ottava Bracket
\ottava #1
c'4 \startTextSpan
% Add Dynamic Text
c4\pp
% Add Dynamic Line Spanner
c4\<
% Add Text Script
c4^Text |
c4 c
% Add Dynamic Text
c4\ff c \stopTextSpan |
% Stop Ottava Bracket
\ottava #0
c,4 c c c |
```



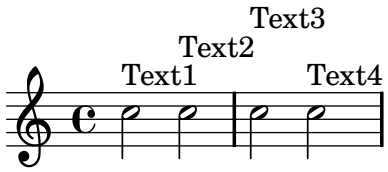
Les liaisons sont intrinsèquement des objets membres de la portée (*within-staff objects*) bien qu'elles la surplombent lorsque les notes auxquelles elles se rattachent sont relativement hautes. Ceci peut avoir pour conséquence de remonter d'autant les objets externes (*outside-staff objects*) tels les articulations. La propriété `avoid-slur` de l'articulation en question peut se voir déterminée à `'inside` dans le but de « ramener » cette articulation à l'intérieur de la liaison. Cette propriété `avoid-slur` ne sera toutefois effective que dans la mesure où la `outside-staff-priority` est désactivée (valeur `#f`). Dans le même esprit, il est possible d'affecter une valeur numérique particulière à la propriété `outside-staff-priority` d'une liaison dans le but de regrouper la liaison avec les objets externes. L'exemple suivant illustre ces deux différentes méthodes.

```
c4( c^\markup { \tiny \sharp } d4.) c8 |
c4(
\once \override TextScript #'avoid-slur = #'inside
\once \override TextScript #'outside-staff-priority = ##f
c4^\markup { \tiny \sharp } d4.) c8 |
\once \override Slur #'outside-staff-priority = #500
c4( c^\markup { \tiny \sharp } d4.) c8 |
```



Le fait de changer la `outside-staff-priority` peut aussi servir à contrôler le positionnement vertical des objets individuels, quoique le résultat ne soit pas toujours formidable. Imaginons que nous voulions placer « Text3 » au-dessus de « Text4 » dans l'exemple de la section Section 4.4.1 [Comportement automatique], page 108, plus haut. Il nous suffit pour cela de regarder dans la RPI ou dans le tableau plus haut la priorité de `TextScript`, et d'augmenter la priorité de « Text3 » jusqu'à une valeur très haute :

```
c2^"Text1"
c2^"Text2" |
\once \override TextScript #'outside-staff-priority = #500
c2^"Text3"
c2^"Text4" |
```
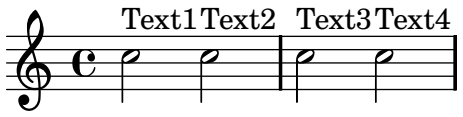


S'il est vrai que cela place « Text3 » au-dessus de « Text4 », ça le place aussi plus haut que « Text2 » tandis que « Text4 » dégringole. Ce n'est peut-être pas si bien que ça. En fait, ce que nous aimerions faire, c'est placer toutes les annotations à égale distance de la portée. Pour cela, nous avons besoin d'espacer horizontalement les notes pour laisser plus de place au texte. C'est possible grâce à la commande `textLengthOn`.

\textLengthOn

Par défaut, l'espacement horizontal d'un texte produit sous forme de *markup* (ou d'étiquette) n'est pas pris en compte, dans la mesure où ce qui est concerné n'entre pas dans la musique. La commande `\textLengthOn` inverse ce comportement, faisant en sorte que les notes soient espacées autant qu'il le faut pour s'adapter au texte :

```
\textLengthOn  % Cause notes to space out to accommodate text
c2^"Text1"
c2^"Text2" |
c2^"Text3"
c2^"Text4" |
```



La commande qui permet de revenir au comportement par défaut est `\textLengthOff`. Rappelez-vous que `\once` ne fonctionne qu'avec `\override`, `\set`, `\revert` ou `\unset`, et donc ne peut pas être utilisé avec `\textLengthOn`.

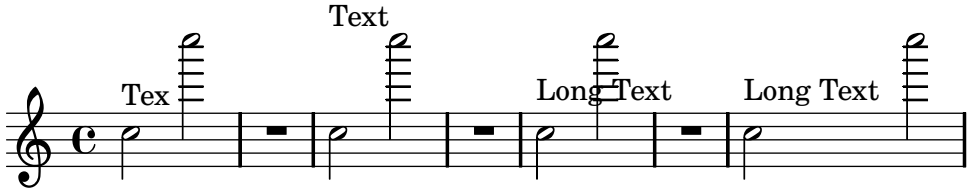
Les textes des *markup* éviteront également les notes qui s'échappent au-dessus de la portée. Si ce n'est pas notre souhait, il est possible de supprimer ce déplacement automatique vers le haut en attribuant à la priorité la valeur `#f`. Voici un exemple qui montre comment les textes des *markup* interagissent avec ces types de note.

```
% This markup is short enough to fit without collision
c2^"Tex" c'' |
R1 |

% This is too long to fit, so it is displaced upwards
c,,2^"Text" c'' |
R1 |

% Turn off collision avoidance
\once \override TextScript #'outside-staff-priority = ##f
c,,2^"Long Text   " c'' |
R1 |

% Turn off collision avoidance
\once \override TextScript #'outside-staff-priority = ##f
\textLengthOn        % and turn on textLengthOn
c,,2^"Long Text   "  % Spaces at end are honored
c''2 |
```

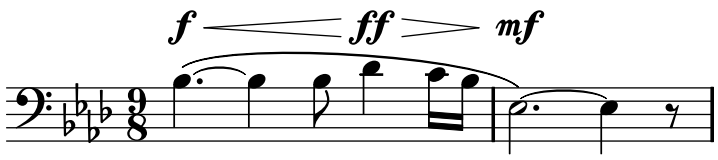


Nuances

Les indications de nuance se placent normalement sous la portée mais on peut les placer au-dessus avec la commande `dynamicUp`. Elles se positionnent verticalement par rapport à la note à
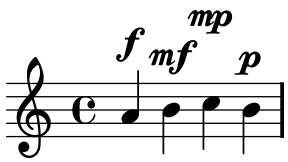
laquelle elles sont liées et se décalent vers le bas (ou le haut) en fonction des objets de la portée comme les liaisons de phrasé ou les numéros de mesure. Cela peut donner d'assez bons résultats, comme le montre cet exemple :

```
\clef "bass"
\key aes \major
\time 9/8
\dynamicUp
bes4.~\f\< \( bes4 bes8 des4\ff\> c16 bes\! |
ees,2.~\)\mf ees4 r8 |
```



De toute façon, si les notes et les nuances qui leur sont liées sont trop proches, le positionnement automatique évitera les collisions en déplaçant davantage les nuances suivantes, mais le résultat peut ne pas être très satisfaisant, comme le montre cet exemple artificiel :

```
\dynamicUp
a4\f b\mf c\mp b\p
```



Si une telle situation devait survenir dans de la musique « réelle », il serait préférable d'espacer un peu plus les notes, de façon que les indications de nuance puissent toutes se situer à la même distance de la portée. Il était possible de faire cela pour les textes de *markup* grâce à la commande `\textLengthOn` mais il n'existe pas d'équivalent pour les indications de nuance. Il nous faut donc chercher à faire cela avec la commande `\override`.

Dimensionnement des objets graphiques
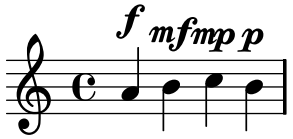
Tout d'abord, nous devons apprendre ce qui détermine la dimension des *grobs*. Tous les *grobs* portent en eux un point de référence qui est utilisé pour les positionner par rapport à leur objet parent. Ce point du *grob* est placé à une distance horizontale, `X-offset`, et à une distance verticale, `Y-offset`, de son parent. L'étendue horizontale de l'objet est fixée par une paire de nombres, `X-extent`, qui donnent la position du coin gauche et du coin droit par rapport au point de référence. De même, l'amplitude verticale est fixée par une paire de nombres, `Y-extent`. Ce sont des propriétés communes à tous les *grobs* et que gère la `grob-interface`.

Par défaut, la largeur des objets extérieurs à la portée est donnée comme étant nulle, si bien qu'ils peuvent se chevaucher horizontalement. Pour remédier à cela, on a ajouté l'infini à l'extension gauche et moins l'infini à l'extension droite, en attribuant à `extra-spacing-width` la valeur `'(+inf.0 . -inf.0)`. Pour être sûr que les objets ne se chevaucheront pas horizontalement, nous devons donc corriger cette valeur de `extra-spacing-width` en `'(0 . 0)`, afin que leur vraie largeur se manifeste. La commande pour y parvenir avec des indications de nuances est :

```
\override DynamicText #'extra-spacing-width = #'(0 . 0)
```
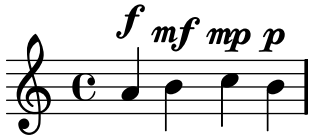
Voyons si ça marche sur notre exemple précédent :

```
\dynamicUp
\override DynamicText #'extra-spacing-width = #'(0 . 0)
a4\f b\mf c\mp b\p |
```



Bon, cela a mis un terme aux déplacements verticaux des nuances mais il reste deux problèmes. Il faudrait que les nuances soient un peu plus écartées et ce serait mieux si elles étaient toutes à la même distance de la portée. Le premier problème est simple à résoudre. Au lieu d'attribuer à `extra-spacing-width` la valeur zéro, nous pourrions mettre un peu plus. L'unité est la distance entre deux lignes de portée, donc en écartant le bord gauche d'une demi-unité et le bord droit d'une demi-unité, on obtient :

```
\dynamicUp
% Extend width by 1 staff space
\override DynamicText #'extra-spacing-width = #'(-0.5 . 0.5)
a4\f b\mf c\mp b\p
```



C'est mieux mais nous voulons peut-être aligner les indications de nuance sur une même ligne plutôt que de les voir monter et descendre avec les notes. La propriété qui gère cela est `staff-padding` ; la section suivante lui est consacrée.

4.5 Collisions d'objets

4.5.1 Déplacement d'objets

Aussi surprenant que cela puisse paraître, LilyPond n'est pas parfait. Certains éléments sur la partition peuvent se chevaucher, ce qui est regrettable mais, le plus souvent, facile à corriger. En général, quand on déplace des objets, c'est pour des raisons de lisibilité ou d'esthétique – ils rendraient mieux avec un peu plus ou un peu moins d'espace autour d'eux.

Il y a trois façons de résoudre les problèmes de chevauchement. Il est préférable de les aborder dans l'ordre suivant :

1. L'**orientation** d'un objet qui en chevauche un autre peut être changée grâce aux commandes prédéfinies dont la liste a été donnée plus haut à propos des objets de portée (voir Section 4.4.2 [Objets inclus dans la portée], page 109). Les queues de note, les liaisons de phrasé et de prolongation, les crochets, les nuances et les nolets peuvent facilement être repositionnés de cette manière. En contrepartie, vous n'avez le choix qu'entre deux positions, sans personnalisation possible.
2. Les **propriétés d'objet**, auxquelles LilyPond a recours pour positionner les objets, sont modifiables avec `\override`. Il y a deux avantages à changer ces propriétés : (a) d'autres objets pourront être déplacés automatiquement si nécessaire pour faire de la place, et (b) la même retouche peut s'appliquer à toutes les occurrences du même type d'objet. Ces propriétés sont :

- `direction`

  Ce point a déjà été traité en détails – voir Section 4.4.2 [Objets inclus dans la portée], page 109.

- `padding`, `right-padding`, `staff-padding`

  Au moment de positionner un objet, la valeur de sa propriété `padding` détermine l'espace à laisser libre entre celui-ci et le coin le plus proche de l'objet à côté duquel il est placé. Vous remarquerez que c'est la valeur `padding` de l'objet **à placer** qui compte ; la valeur `padding` de l'objet déjà placé est ignorée. Les espaces libres déterminés par `padding` s'appliquent à tous les objets associés à la `side-position-interface`.

  Le positionnement de groupes d'altérations est contrôlé par `right-padding`, et non plus `padding`. Cette propriété appartient à l'objet `AccidentalPlacement` qui, vous le remarquerez, prend place dans le contexte **Staff**. Dans le processus de composition, les têtes de notes sont disposées en premier, puis les altérations, s'il y en a, sont ajoutées à gauche des têtes de note suivant la propriété `right-padding` qui détermine l'espacement par rapport aux têtes de note. C'est pourquoi seule la propriété `right-padding` de l'objet `AccidentalPlacement` joue sur le positionnement des altérations.

  La propriété `staff-padding` est très proche de la propriété `padding` : `padding` contrôle l'espace minimum entre un objet qui accepte la `side-position-interface` et l'objet le plus proche (généralement une note ou une ligne de portée) ; `staff-padding` ne s'applique qu'aux objets qui sont toujours placés en-dehors de la portée – il contrôle l'espace minimum à insérer entre l'objet et la portée. Attention : par défaut, `staff-padding` concerne les objets positionnés par rapport à la portée et n'a aucun effet sur les objets qui sont positionnés par rapport à une note ; il est toutefois possible de le régler pour fonctionner avec ces derniers.

  Pour trouver quelle propriété `padding` employer pour l'objet que vous cherchez à repositionner, il vous faut consulter les propriétés de l'objet dans la RPI. Prenez garde que les propriétés `padding` ne sont pas forcément traitées dans l'objet en question ; il faut alors regarder les objets qui semblent s'en rapprocher.

  Toutes les valeurs `padding` sont exprimées en espaces de portée. Pour la plupart des objets, la valeur par défaut est aux alentours de 1,0 et parfois moins (cela dépend de chaque objet). Il est possible de la modifier lorsqu'on a besoin d'un espace vide plus grand (ou plus petit).

- `self-alignment-X`

  Cette propriété sert à aligner les objets sur la gauche, sur la droite ou à les centrer par rapport au point de référence des objets parents. Elle peut être utilisée avec tous les objets qui acceptent la `self-alignment-interface`. Il s'agit, en général, des objets qui contiennent du texte. Les valeurs admises sont `LEFT`, `RIGHT` et `CENTER`. On peut aussi attribuer à la place une valeur numérique entre `-1` et `+1`, où `-1` signifie alignement sur la gauche, `+1` alignement sur la droite, et les nombres intermédiaires déplacent progressivement le texte de la gauche vers la droite. Des valeurs numériques supérieures à `1` sont également admises pour déplacer le texte encore plus loin vers la gauche, ou des valeurs inférieures à `-1` pour déplacer le texte encore plus loin vers la droite. Un écart de `1` en valeur correspond à un déplacement de la moitié de la longueur du texte.

- `extra-spacing-width`

  Cette propriété est utilisée pour tous les objets qui acceptent la `item-interface`. Elle reçoit deux nombres, le premier étant ajouté au bord gauche et le second au bord droit. Des nombres négatifs déplacent le coin vers la gauche, des nombres positifs vers la droite, si bien que pour élargir un objet, le premier nombre doit être négatif et le second positif. Attention : tous les objets n'acceptent pas forcément les deux nombres. Par exemple, l'objet `Accidental` ne retient que le premier nombre (coin gauche).

- `staff-position`

  `staff-position` est une propriété de la `staff-symbol-referencer-interface`, qui s'applique aux objets positionnés par rapport à la portée. Elle indique, en demi-espaces de portée, la position verticale des objets par rapport à la ligne médiane de la portée. C'est bien pratique pour résoudre des problèmes de collision entre des objets comme les silences valant mesure entière, les liaisons et les notes de différentes voix.

- `force-hshift`

  Des notes très proches dans un accord, ou des notes simultanées dans différentes voix, peuvent être disposées sur deux colonnes, rarement plus, pour éviter que les têtes de notes ne se chevauchent. On parle alors de colonnes (ou empilement) de notes et un objet appelé `NoteColumn` est créé pour placer les notes sur la colonne.

  La propriété `force-hshift` appartient à `NoteColumn` (en réalité à la `note-column-interface`). Le fait de la modifier permet de déplacer un empilement selon l'unité appropriée aux colonnes de notes, à savoir la largeur des têtes de note de la première voix. Son utilisation est réservée à des situations complexes dans lesquelles les commandes habituelles `\shiftOn` (voir Section 3.2.2 [Instanciation explicite des voix], page 53) ne suffisent plus à résoudre les conflits. Elle est alors préférable à l'utilisation de la propriété `extra-offset`, dans la mesure où on n'a pas besoin d'exprimer la distance en espaces de portée et où le fait de déplacer les notes à l'intérieur ou à l'extérieur d'une `NoteColumn` affecte d'autres actions comme les fusions de notes.

3. Pour terminer, quand toutes les autres méthodes ont échoué, il est possible de repositionner verticalement les objets à la main par rapport à la ligne médiane de la portée, ou en les déplaçant à une distance donnée vers une nouvelle position. Les inconvénients sont qu'il faut individuellement, pour chaque objet, trouver les valeurs correctes de repositionnement souvent par tâtonnement, et que, puisque le mouvement est opéré après que LilyPond a placé tous les autres objets, c'est à l'utilisateur de résoudre tous les problèmes de collision qui pourraient survenir. Et le pire avec cette méthode est que, le jour où la musique est modifiée, il faut de nouveau rechercher les valeurs de repositionnement. Les propriétés à utiliser pour ce type de repositionnement manuel sont :

   `extra-offset`
   : Cette propriété s'applique à tout objet acceptant la `grob-interface`. Elle reçoit une paire de nombre qui indiquent le déplacement supplémentaire dans le sens horizontal et vertical. Des nombres négatifs déplacent l'objet vers la gauche ou vers la droite. L'unité utilisée est l'espace de portée. Le déplacement supplémentaire intervient une fois que la composition des objets est achevée, si bien qu'un objet peut être repositionné à n'importe quel endroit sans perturber quoi que ce soit.

   `positions`
   : Cette propriété est très utile pour ajuster manuellement l'inclinaison et la hauteur des barres de ligature, des liaisons et des nolets. Elle est suivie de deux nombres qui donnent la position des bords gauche et droit des barres, liaisons, etc. par rapport à la ligne médiane de la portée. L'unité de référence est l'intervalle de lignes de portée. Attention toutefois au fait que les liaisons et phrasés ne peuvent pas être repositionnés n'importe où. LilyPond commence par dresser la liste des emplacements possibles pour les liaisons et choisit par défaut la liaison qui « semble la meilleure ». Si la propriété `positions` a été retouchée, la liaison la plus proche de la position demandée sera retenue dans la liste.

Il est possible qu'un objet ne dispose pas de toutes ces propriétés. Il est donc nécessaire de consulter la RPI pour vérifier quelles sont les propriétés disponibles pour l'objet en question.

Voici une liste d'objets les plus couramment impliqués dans les collisions, avec le nom de l'objet à consulter dans la RPI afin de trouver les propriétés à retoucher pour obtenir un déplacement.

Type d'objet	**Nom d'objet**
Articulations	`Script`
Barres de ligature	`Beam`
Doigté	`Fingering`
Liaisons de phrasé	`Slur`
Liaisons de prolongation	`Tie`
Nolets	`TupletBracket`
Nuances (verticalement)	`DynamicLineSpanner`
Nuances (horizontalement)	`DynamicText`
Reprises / marques de texte	`RehearsalMark`
Texte, p.ex. `^"texte"`	`TextScript`

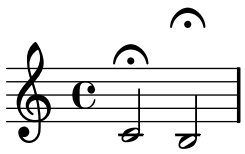
4.5.2 Correction des collisions d'objets

Voyons maintenant comment les propriétés décrites dans la section précédente peuvent nous aider à résoudre les collisions.

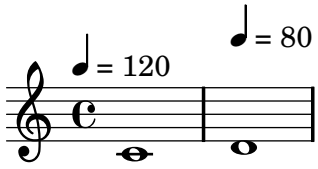
la propriété padding

En jouant sur la propriété `padding` (littéralement « rembourrage »), on augmente (ou on diminue) la distance entre des symboles qui sont imprimés au-dessus ou en dessous des notes.

```
c2\fermata
\override Script #'padding = #3
b2\fermata
```



```
% This will not work, see below:
\override MetronomeMark #'padding = #3
\tempo 4=120
c1 |
% This works:
\override Score.MetronomeMark #'padding = #3
\tempo 4=80
d1 |
```
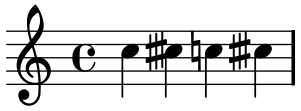


Vous remarquerez dans le second exemple à quel point il est important de préciser le nom du contexte qui contient l'objet. Puisque l'objet `MetronomeMark` appartient au contexte `Score`, le fait de modifier la propriété dans le contexte `Voice` passera inaperçu. Pour plus de détails, voir Section "Modification de propriétés" dans *Manuel de notation*.

Si on augmente la propriété `padding` d'un objet alors que celui-ci fait partie d'un ensemble d'objets positionnés en fonction de leur `outside-staff-priority`, cet objet sera déplacé, ainsi que tous les autres objets du groupe.

right-padding

La propriété `right-padding` joue sur l'espacement entre une altération et la note sur laquelle elle porte. On ne l'utilise pas souvent, mais l'espacement par défaut peut se révéler inadéquat avec certaines altérations ou certains glyphes utilisés en musique microtonale. Le stencil de l'altération devra alors être construit sous la forme d'un `markup` qui contiendra le ou les symboles requis, comme ceci :

```
sesquisharp = \markup { \sesquisharp }
\relative c'' {
  c4
  % This prints a sesquisharp but the spacing is too small
  \once \override Accidental
    #'stencil = #ly:text-interface::print
  \once \override Accidental #'text = #sesquisharp
  cis4 c
  % This improves the spacing
  \once \override Score.AccidentalPlacement #'right-padding = #0.6
  \once \override Accidental
    #'stencil = #ly:text-interface::print
  \once \override Accidental #'text = #sesquisharp
  cis4 |
}
```
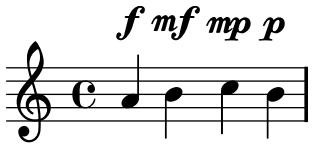


Cette méthode utilise, pour le stencil des altérations, une retouche qui ne sera pas reprise par la suite. Le type de stencil est obligatoirement une procédure, qui consiste ici à imprimer le contenu de la propriété `text` de `Accidental`, déclaré comme étant un signe sesqui-dièse. Celui-ci est ensuite repoussé devant la tête de note par la retouche de `right-padding`.

la propriété staff-padding

`staff-padding` sert à aligner des objets tels que des nuances sur une ligne fictive à une hauteur donnée par rapport à la portée plutôt qu'à une hauteur qui varie en fonction de la position de la note sur laquelle porte l'objet. Ce n'est pas une propriété de `DynamicText` mais de `DynamicLineSpanner`, car la ligne fictive est destinée à s'appliquer autant à **toutes** les nuances, notamment celles qui sont créées comme des bandeaux en longueur (en anglais *Spanners*). Tel est donc le moyen d'aligner les indications de nuance, comme dans cet exemple repris de la section précédente :

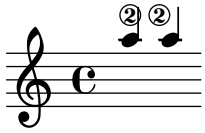
```
\dynamicUp
% Extend width by 1 unit
\override DynamicText #'extra-spacing-width = #'(-0.5 . 0.5)
% Align dynamics to a base line 2 units above staff
\override DynamicLineSpanner #'staff-padding = #2
a4\f b\mf c\mp b\p
```



la propriété self-alignment-X

L'exemple suivant montre comment résoudre une collision entre une indication de corde et une hampe, en alignant le coin droit sur le point de référence de la note parente :
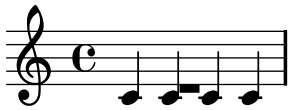
```
\voiceOne
<a\2>
\once \override StringNumber #'self-alignment-X = #RIGHT
<a\2>
```



la propriété staff-position

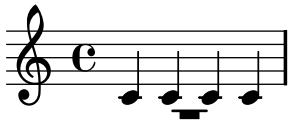
Dans une voix, un silence valant une mesure entière peut chevaucher les notes d'une autre voix. Vu que ces silences sont centrés entre les deux barres de mesure, il serait très compliqué de programmer LilyPond pour repérer ces risques de collision dans la mesure où, normalement, toutes les collisions entre notes ou entre notes et silences se produisent sur des notes et silences simultanés. Voici un exemple de collision de ce type :

```
<< { c4 c c c } \\ { R1 } >>
```



Ici, la meilleure solution consiste à déplacer le symbole de pause vers le bas – puisque cette pause appartient à la voix deux. Par défaut, dans la `\voiceTwo` (c'est-à-dire dans la seconde voix d'une construction `<<{...} \\ {...}>>`), la propriété `staff-position` est réglée sur `-4` pour les `MultiMeasureRest` ; nous avons donc besoin de la déplacer, disons, de quatre demi-intervalles vers le bas, ce qui donne `-8`.

```
<<
  { c4 c c c }
  \\
  \override MultiMeasureRest #'staff-position = #-8
  { R1 }
>>
```



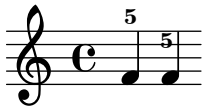
C'est mieux que d'utiliser, par exemple, `extra-offset`, car la ligne supplémentaire au-dessus du silence est insérée automatiquement.

la propriété extra-offset

La propriété `extra-offset` offre la possibilité de contrôler entièrement le placement d'un objet, aussi bien horizontalement que verticalement.

Dans l'exemple suivant, la seconde indication de doigté est déplacée légèrement vers la gauche et de 1,8 intervalle de lignes vers le bas :

```
\stemUp
f4-5
\once \override Fingering #'extra-offset = #'(-0.3 . -1.8)
f4-5
```



la propriété positions
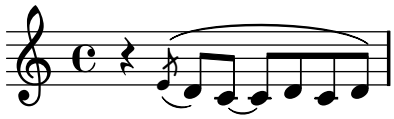
La propriété `positions` permet de contrôler manuellement la position et l'inclinaison des nolets, coulés, liaisons de phrasé et barres de ligature. Voici un exemple avec une horrible liaison de phrasé – horrible pour avoir tenté de contourner la liaison de l'acciaccature.

```
r4 \acciaccatura e8\( d8 c~ c d c d\)
```
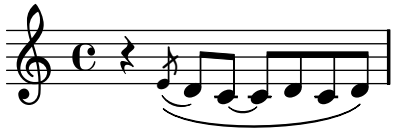


Nous pourrions tout simplement déplacer la liaison de phrasé au-dessus des notes, et ce serait la meilleure solution :

```
r4
\phrasingSlurUp
\acciaccatura e8\( d8 c~ c d c d\)
```



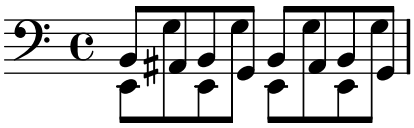
Mais si, pour une quelconque raison, cette solution n'était pas envisageable, l'autre solution consiste à déplacer légèrement vers le bas l'extrémité gauche de la liaison de phrasé, grâce à la propriété `positions`. Cela corrige en même temps la forme plutôt disgracieuse de la liaison.

```
r4
\once \override PhrasingSlur #'positions = #'(-4 . -3)
\acciaccatura e8\( d8 c~ c d c d\)
```



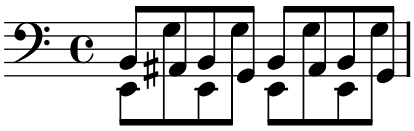
Voici un autre exemple, tiré du début de la partie de main gauche du Prélude de Chopin, Op. 28, No 2. Comme nous pouvons le constater, les barres de ligature chevauchent les notes les plus hautes :

```
{
  \clef "bass"
  << { b,8 ais, b, g, } \\ { e,8 g e, g } >>
  << { b,8 ais, b, g, } \\ { e,8 g e, g } >>
}
```



On peut y remédier en déplaçant manuellement vers le haut les deux extrémités des ligatures de croches, non plus à 2 intervalles au-dessus de la ligne médiane mais, disons, à 3 :

```
{
  \clef "bass"
  <<
    \override Beam #'positions = #'(3 . 3)
    { b,8 ais, b, g, }
  \\
    { e,8 g e, g }
  >>
  << { b,8 ais, b, g, } \\ { e,8 g e, g } >>
}
```
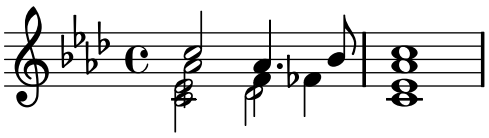


Vous remarquerez que la retouche continue de s'appliquer au second bloc de croches de la première voix mais qu'il ne s'applique à aucune barre de la deuxième voix.

la propriété force-hshift

Maintenant, nous sommes prêts à appliquer les dernières corrections à l'exemple de Chopin présenté à la fin de Section 3.2.1 [J'entends des Voix], page 48, que nous avions laissé dans cet état :
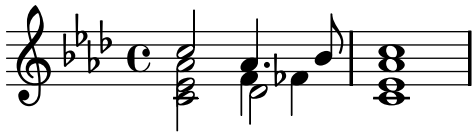
```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 }
    \\
    { aes2 f4 fes }
    \\
    {
      \voiceFour
      <ees c>2 des
    }
  >> |
  <c ees aes c>1 |
}
```



Les deux plus basses notes du premier accord (c'est-à-dire celles de la troisième voix) ne devraient pas être décalées de l'empilement des deux plus hautes notes. Pour y remédier, nous réglons le `force-hshift` – qui est une propriété de `NoteColumn` – de ces notes sur zéro. Ensuite, la note la plus basse du second accord serait mieux à droite des notes plus hautes. Pour cela, nous réglons le `force-hshift` de cette note sur 0,5 – c'est-à-dire la moitié de la largeur d'une tête de note vers la droite de la colonne des notes plus hautes.

Et voici le résultat final :

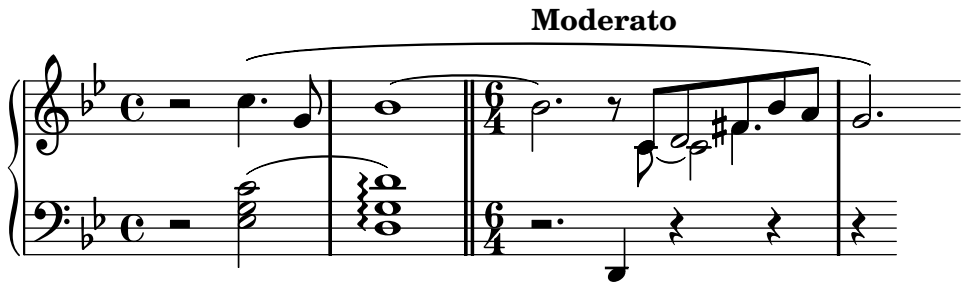
```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 }
    \\
    { aes2 f4 fes }
    \\
    {
      \voiceFour
      \once \override NoteColumn #'force-hshift = #0
      <ees c>2
      \once \override NoteColumn #'force-hshift = #0.5
      des2
    }
  >> |
  <c ees aes c>1 |
}
```



4.5.3 Exemple concret

Pour terminer ce chapitre consacré aux retouches, voici, étape par étape, la mise en forme d'un exemple concret nécessitant un certain nombre de retouches jusqu'à l'obtention du résultat attendu. Cet exemple a été choisi en raison des problèmes inhabituels de notation qu'il soulevait et pour vous apprendre à les résoudre grâce au Manuel de notation. Il n'est pas représentatif d'une opération normale de gravure ; que ces difficultés ne vous découragent donc pas ! Des difficultés comme celles-ci ne sont, heureusement, pas courantes !

Cet exemple est tiré de la Première Ballade de Chopin, Op. 23, mesures 6 à 9 ; cela correspond à la transition entre le Lento d'ouverture et le Moderato. Voici, pour commencer, ce à quoi nous voulons que la partition ressemble ; pour limiter les complications, nous n'avons pas fait apparaître les indications de nuance, de doigté ni de pédale.



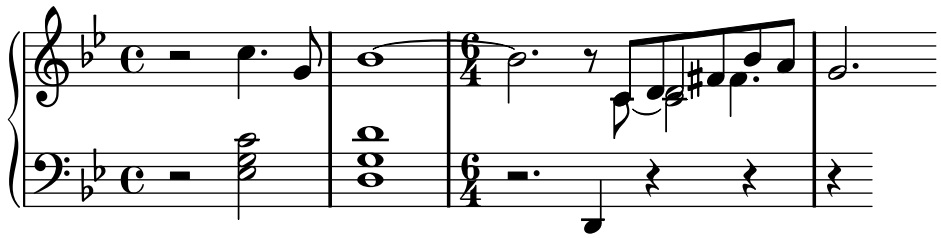
Nous constatons tout d'abord que, dans la troisième mesure, la main droite compte quatre voix. Ce sont les cinq croches avec une barre, le do avec liaison, le ré blanche qui se fond avec le ré croche, et le fa dièse noire pointée qui se fond lui aussi avec la croche de même hauteur. Tout le reste se réduit à une seule voix. Le plus simple est donc de créer temporairement ces quatre voix au moment opportun. Si vous avez oublié comment faire, reportez-vous à Section 3.2.1 [J'entends des Voix], page 48. Commençons par saisir les notes comme appartenant à deux

variables, mettons en place l'ossature des portées dans un bloc `\Score` et voyons ce que LilyPond propose par défaut :

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4. g8 |
    bes1~ |
    \time 6/4
    bes2. r8
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    g2.  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2 |
  <d g, d>1 |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



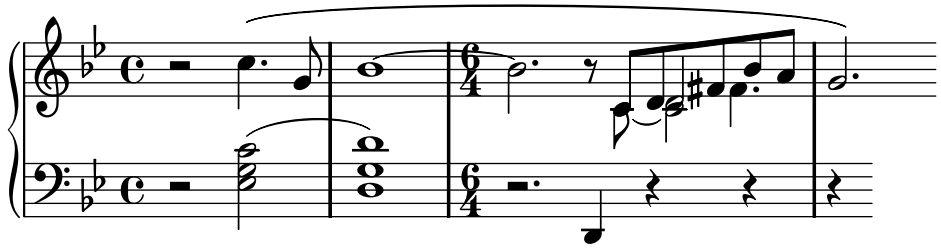
Toutes les notes sont correctes mais l'allure générale est loin d'être satisfaisante. La liaison se heurte à l'indication de mesure lors du changement de métrique, la ligature des croches n'est pas bonne dans la troisième mesure, les notes ne sont pas fusionnées et il manque plusieurs éléments de notation. Commençons par le plus simple. Nous pouvons corriger la ligature des croches en la créant manuellement et nous pouvons facilement ajouter les limites droite et gauche de la liaison de phrasé, puisque tout cela a déjà été traité dans le tutoriel. Voici le résultat :

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    bes1~ |
    \time 6/4
    bes2. r8
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1) |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
```

```
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



La première mesure est maintenant correcte. La seconde contient un arpège et doit se terminer par une double barre. Comment faire, puisque cela n'a pas été traité dans le Manuel d'initiation ? C'est alors qu'il faut nous reporter au Manuel de notation. Quand on cherche « arpège » et « barre de mesure » dans l'index, on voit aisément qu'il faut ajouter `\arpeggio` à un accord pour produire un arpège et qu'une double barre est le résultat de la commande `\bar "||"`. Rien de plus facile ! Nous devons ensuite corriger la collision entre la liaison et l'indication de mesure. Le mieux est de déplacer la liaison vers le haut. La méthode pour déplacer les objets a déjà été présentée dans Section 4.5.1 [Déplacement d'objets], page 118, et l'on sait que, pour des objets positionnés par rapport à la portée, il nous faut modifier leur propriété `staff-position`, exprimée en demi-intervalles de lignes par rapport à la ligne médiane de la portée. Voici donc la retouche à insérer juste devant la première note liée ; elle est censée déplacer la liaison vers le haut de 3,5 demi-intervalles de lignes au-dessus de la ligne médiane :

```
\once \override Tie #'staff-position = #3.5
```

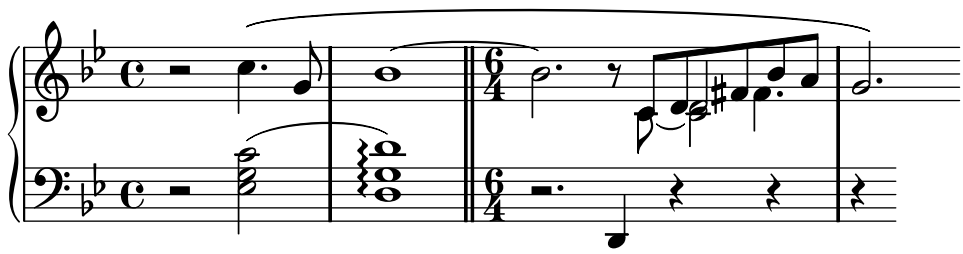
Cela s'adjoint à la deuxième mesure, pour donner :

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
    bes1~ |
    \bar "||"
    \time 6/4
    bes2. r8
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
```

```
    >> |
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



Venons-en à la troisième mesure et au début de la section Moderato. Dans le Tutoriel, il est indiqué comment insérer du texte en gras à l'aide de la commande `\markup` ; pas de problème, du coup, pour ajouter « Moderato » en gras. Mais comment faire pour fusionner les notes de différentes voix ? C'est là que le Manuel de notation peut nous venir en aide. Une recherche sur « fusionnement de notes » dans l'index nous renvoie au chapitre Section "Résolution des collisions" dans *Manuel de notation*, et plus précisément aux commandes permettant de fusionner des notes en fonction de leur type et selon qu'elles sont pointées ou non. Dans notre exemple, pour la durée de la polyphonie de la troisième mesure, nous avons besoin de fusionner deux types de notes ; grâce aux informations trouvées dans le Manuel de notation, nous ajoutons

```
\mergeDifferentlyHeadedOn
\mergeDifferentlyDottedOn
```

au début de cette section et

```
\mergeDifferentlyHeadedOff
\mergeDifferentlyDottedOff
```

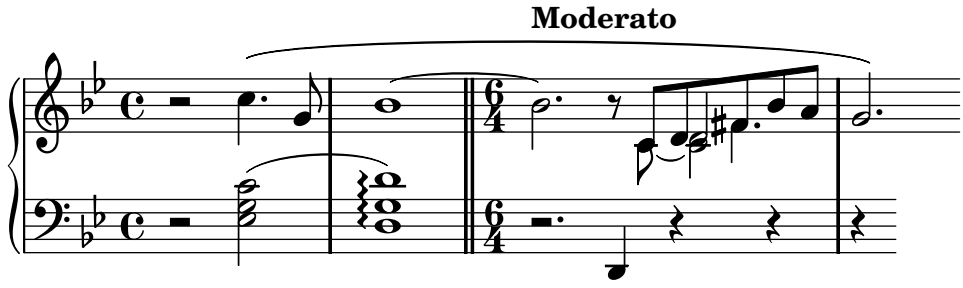
à la fin, ce qui donne :

```
rhMusic = \relative c'' {
  \new Voice {
```

```
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
    bes1~ |
    \bar "||"
    \time 6/4
    bes2.^\markup { \bold "Moderato" } r8
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    \mergeDifferentlyHeadedOff
    \mergeDifferentlyDottedOff
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```

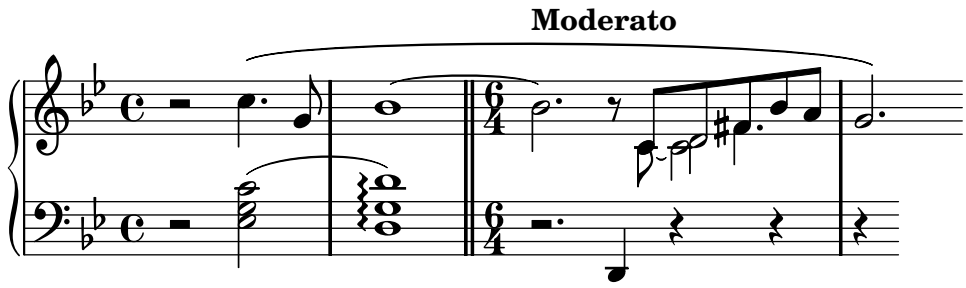


Ces retouches ont permis de fusionner les deux fa dièse mais pas les deux ré. Pourquoi ? La réponse se trouve dans la même section du Manuel de notation : les notes à fusionner doivent avoir des hampes dans des directions opposées et deux notes ne peuvent pas être fusionnées s'il y a une troisième note dans la même colonne. Ici, les deux ré ont leur hampe orientée vers le haut et il y a une troisième note, do. Nous savons changer l'orientation de la hampe avec `\stemDown` et le Manuel de notation nous indique également comment déplacer le do – en produisant un décalage grâce à l'une des commandes `\shift`. Mais laquelle ? Le do appartient à la deuxième voix et n'est pas décalé ; les deux ré appartiennent respectivement à la première et à la troisième voix, et l'un n'est pas décalé tandis que l'autre l'est. Il nous faut donc décaler largement le do avec la commande `\shiftOnn` pour éviter une interférence avec les deux ré. Voici ce que donnent ces modifications :

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
    bes1~ |
    \bar "||"
    \time 6/4
    bes2.^\markup { \bold "Moderato" } r8
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        % Move the c2 out of the main note column so the merge will work
        c,8~ \shiftOnn c2
      }
      \new Voice {
        \voiceThree
        % Stem on the d2 must be down to permit merging
        s8 \stemDown d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    \mergeDifferentlyHeadedOff
    \mergeDifferentlyDottedOff
    g2.\)  % continuation of main voice
  }
}
```

```
lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



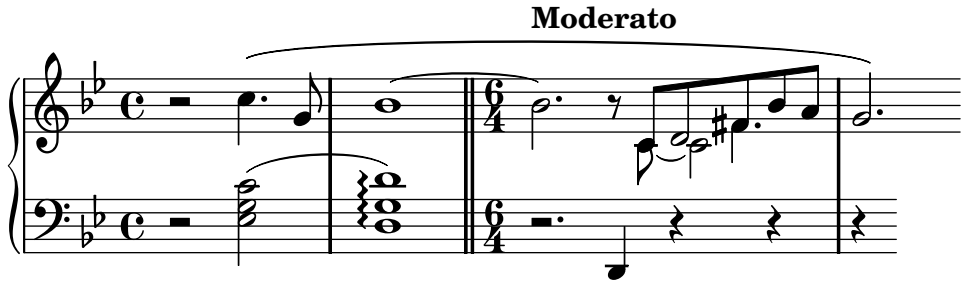
Pas loin ! Il ne reste plus que deux problèmes : les ré une fois fusionnés ne devraient plus avoir de hampe vers le bas, et le do serait mieux à la droite des ré. Nous savons remédier à ces deux problèmes grâce aux retouches précédentes : nous allons rendre la hampe transparente et déplacer le do avec la propriété `force-hshift`. Et voici le résultat final :

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
    bes1~ |
    \bar "||"
    \time 6/4
    bes2.^\markup { \bold "Moderato" } r8
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~
        % Reposition the c2 to the right of the merged note
        \once \override NoteColumn #'force-hshift = #1.0
```

```
        % Move the c2 out of the main note column so the merge will work
        \shiftOnn
        c2
      }
      \new Voice {
        \voiceThree
        s8
        % Stem on the d2 must be down to permit merging
        \stemDown
        % Stem on the d2 should be invisible
        \once \override Stem #'transparent = ##t
        d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    \mergeDifferentlyHeadedOff
    \mergeDifferentlyDottedOff
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```
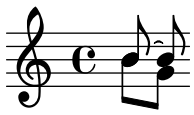


4.6 Autres retouches

4.6.1 Autres utilisations des retouches
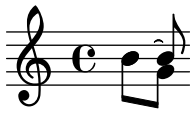
Liaison entre plusieurs voix

Voici un exemple qui montre comment créer une liaison de prolongation entre des notes appartenant à des voix différentes. En temps normal, seules deux notes appartenant à une même voix peuvent être ainsi liées. La solution consiste à utiliser deux voix, dont l'une avec les notes liées
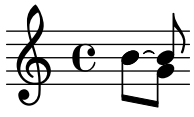


et à rendre transparente la première hampe de cette voix ; on a alors l'impression que la liaison couvre les deux voix.

```
<<
  {
    \once \override Stem #'transparent = ##t
    b8~ b\noBeam
  }
\\
  { b8[ g] }
>>
```



Pour être sûr que la hampe que nous avons rendue transparente n'empiète pas trop sur le trait de liaison, nous pouvons l'allonger en réglant la longueur (`length`) sur `8`,

```
<<
  {
    \once \override Stem #'transparent = ##t
    \once \override Stem #'length = #8
    b8~ b8\noBeam
  }
\\
  { b[ g8] }
>>
```

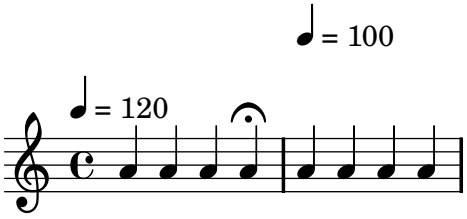


Rendu MIDI d'un point d'orgue
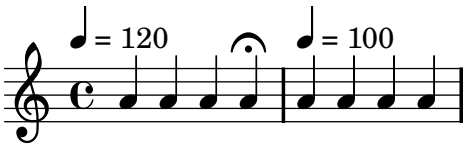
En ce qui concerne les objets extérieurs à la portée, quand on veut les faire disparaître de la partition imprimée, il est généralement préférable de modifier leur propriété `stencil` plutôt que leur propriété `transparent`. Le fait d'attribuer à la propriété `stencil` d'un objet la valeur `#f` supprimera entièrement celui-ci de la partition. Il ne risquera donc pas de gêner le placement d'autres objets.

Par exemple, si nous voulons changer le réglage de métronome pour simuler un point d'orgue dans le fichier MIDI, nous ne voulons surtout pas que cette indication métronomique apparaisse sur la partition ni qu'elle influence l'espacement entre les deux systèmes ou la position des annotations voisines sur la portée. Le fait d'attribuer à la propriété `stencil` la valeur `#f` est donc la bonne solution. Nous montrons ci-dessous le résultat des deux méthodes :

```
\score {
  \relative c'' {
    % Visible tempo marking
    \tempo 4=120
    a4 a a
    \once \override Score.MetronomeMark #'transparent = ##t
    % Invisible tempo marking to lengthen fermata in MIDI
    \tempo 4=80
    a4\fermata |
    % New tempo for next section
    \tempo 4=100
    a4 a a a |
  }
  \layout { }
  \midi { }
}
```




```
\score {
  \relative c'' {
    % Visible tempo marking
    \tempo 4=120
    a4 a a
    \once \override Score.MetronomeMark #'stencil = ##f
    % Invisible tempo marking to lengthen fermata in MIDI
    \tempo 4=80
    a4\fermata |
    % New tempo for next section
    \tempo 4=100
    a4 a a a |
  }
  \layout { }
  \midi { }
}
```



Les deux méthodes permettent d'enlever l'indication métronomique qui allonge le point d'orgue de la partition, et toutes deux modifient le rythme MIDI comme souhaité mais, dans la première, l'indication métronomique transparente repousse vers le haut l'indication de tempo, contrairement à la seconde (avec le stencil désactivé) qui la laisse à sa place.

Voir aussi

Glossaire musicologique : Section "système" dans *Glossaire*.

4.6.2 Utilisation de variables dans les retouches

Les commandes de retouche sont souvent longues et pénibles à taper, et ne tolèrent pas la moindre erreur. Lorsqu'on a besoin de faire plusieurs fois les mêmes retouches, il est préférable de définir des variables qui les contiennent.

Imaginons que nous voulions accentuer certains mots dans des paroles en les mettant en italique. Au lieu des commandes `\italic` et `\bold`, qui ne fonctionnent dans les paroles que si elles sont enchâssées dans un `\markup` – ce qui les rend pénibles à saisir – pouvons-nous employer les commandes `\override` et `\revert` ?

```
\override Lyrics . LyricText #'font-shape = #'italic
\override Lyrics . LyricText #'font-series = #'bold

\revert Lyrics . LyricText #'font-shape
\revert Lyrics . LyricText #'font-series
```

Là encore, ce serait extrêmement pénible à saisir, surtout s'il y avait beaucoup de mots à retoucher de cette façon. Plutôt que cette solution, nous déclarons ces commandes comme étant deux variables, et les utilisons comme ci-après – quoique on choisirait sans doute pour les variables des noms plus courts pour simplifier la frappe. Par ailleurs, le fait de recourir à une variable ne nous expose plus à l'oubli des espaces entourant les points lorsqu'explicités au beau milieu d'un bloc `\lyriqcmode` !

```
emphasize = {
  \override Lyrics.LyricText #'font-shape = #'italic
  \override Lyrics.LyricText #'font-series = #'bold
}

normal = {
  \revert Lyrics.LyricText #'font-shape
  \revert Lyrics.LyricText #'font-series
}

global = { \key c \major \time 4/4 \partial 4 }

SopranoMusic = \relative c' { c4 | e4. e8 g4 g    | a4   a   g  }
AltoMusic    = \relative c' { c4 | c4. c8 e4 e    | f4   f   e  }
TenorMusic   = \relative c  { e4 | g4. g8 c4.  b8 | a8 b c d e4 }
BassMusic    = \relative c  { c4 | c4. c8 c4 c    | f8 g a b c4 }

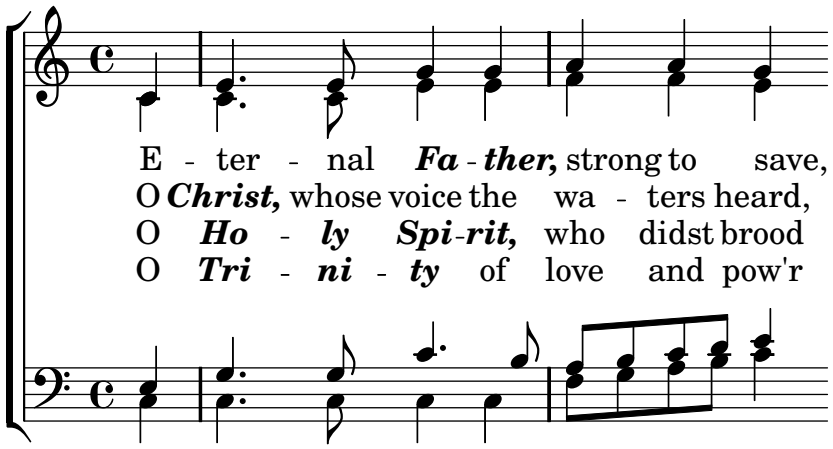
VerseOne = \lyrics {
  E -- | ter -- nal \emphasize Fa -- ther, | \normal strong to save,
}
```

```
VerseTwo = \lyricmode {
  O | \emphasize Christ, \normal whose voice the | wa -- ters heard,
}

VerseThree = \lyricmode {
  O | \emphasize Ho -- ly Spi -- rit, | \normal who didst brood
}

VerseFour = \lyricmode {
  O | \emphasize Tri -- ni -- ty \normal of | love and pow'r
}

\score {
  \new ChoirStaff <<
    \new Staff <<
      \clef "treble"
      \new Voice = "Soprano"  { \voiceOne \global \SopranoMusic }
      \new Voice = "Alto" { \voiceTwo \AltoMusic }
      \new Lyrics \lyricsto "Soprano" { \VerseOne }
      \new Lyrics \lyricsto "Soprano" { \VerseTwo }
      \new Lyrics \lyricsto "Soprano" { \VerseThree }
      \new Lyrics \lyricsto "Soprano" { \VerseFour }
    >>
    \new Staff <<
      \clef "bass"
      \new Voice = "Tenor" { \voiceOne \TenorMusic }
      \new Voice = "Bass"  { \voiceTwo \BassMusic }
    >>
  >>
}
```



4.6.3 Feuilles de style

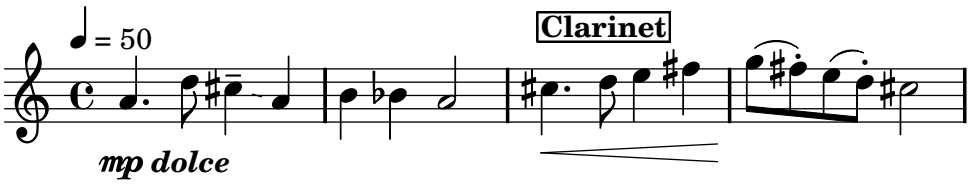
La sortie que produit LilyPond peut être largement modifiée – voir Chapitre 4 [Retouche de partition], page 88 pour plus de détails. Mais que faire si vous avez beaucoup de fichiers auxquels vous souhaitez appliquer vos retouches ? Ou si vous souhaitez simplement séparer les retouches de la musique elle-même ? Rien de plus facile.

Prenons un exemple. Ne vous inquiétez pas si vous ne comprenez pas les parties avec tous les `#()`. Celles-ci sont expliquées dans Section 4.6.6 [Retouches avancées avec Scheme], page 144.

```
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a |
  b4 bes a2 |
  \inst "Clarinet"
  cis4.\< d8 e4 fis |
  g8(\! fis)-. e( d)-. cis2 |
}
```



Il y a quelques problèmes de chevauchement ; nous allons arranger cela en utilisant les techniques de Section 4.5.1 [Déplacement d'objets], page 118. On peut aussi faire quelque chose pour les définitions de `mpdolce` et `inst`. Elles produisent le résultat que nous désirons, mais nous pourrions aussi vouloir les utiliser dans une autre pièce. Il suffirait de les copier et coller au début de chaque fichier, mais c'est fastidieux. De plus, cela laisse les définitions dans nos fichiers de musique, et je trouve personnellement tous ces `#()` assez laids. Stockons-les dans un autre fichier :

```
%%% enregistrez ceci dans un fichier nommé "definitions.ily"
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

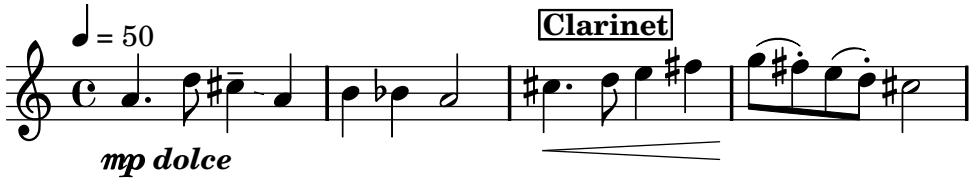
inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
```

```
    'direction UP
    'text (markup #:bold (#:box string))))
```

Nous rappellerons ce fichier par une simple commande `\include` au début de notre fichier de musique. Lui attribuer l'extension `.ily` nous permet de distinguer aisément qu'il s'agit d'un fichier voué à être inclus dans un fichier maître ; il n'est pas destiné à être compilé isolément. Maintenant, modifions notre musique (enregistrez ce fichier sous '`musique.ly`').

```
\include "definitions.ily"

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a |
  b4 bes a2 |
  \inst "Clarinet"
  cis4.\< d8 e4 fis |
  g8(\! fis)-. e( d)-. cis2 |
}
```



C'est mieux, mais effectuons encore quelques retouches. Le glissando est peu visible, c'est pourquoi nous allons l'épaissir et le rapprocher des têtes de note. Déplaçons l'indication métronomique au-dessus de la clef, au lieu de la laisser au-dessus de la première note. Et pour finir, mon professeur de composition déteste les chiffrages de mesure en « C », nous allons donc le transformer en « 4/4 ».

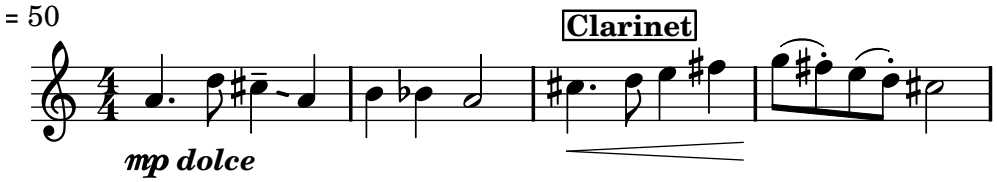
Cependant, ne changez pas le fichier '`musique.ly`'. Remplacez le fichier '`definitions.ily`' par ceci :

```
%%%  definitions.ily
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

\layout{
  \context {
    \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
```

```
    \override MetronomeMark #'padding = #'3
  }
  \context {
    \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context {
    \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



C'est encore mieux ! Mais supposons maintenant que je veuille publier cette pièce. Mon professeur de composition n'aime pas les chiffrages de mesure en « C », mais moi je les aime bien. Copions l'actuel '`definitions.ily`' dans le fichier '`publication-web.ily`', et modifions ce dernier. Puisque la musique est destinée à produire un fichier PDF affiché sur écran, nous allons aussi augmenter la taille globale de police.
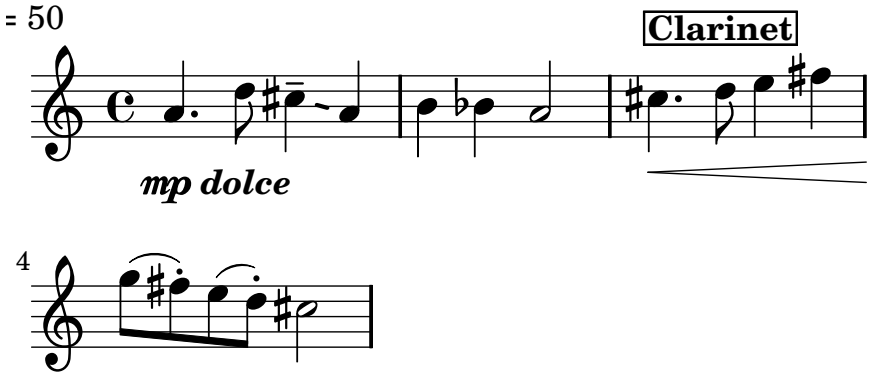
```
%%%  definitions.ily
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

#(set-global-staff-size 23)

\layout{
  \context {
    \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context {
    \Staff
```

```
  }
  \context {
    \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



Il ne nous reste plus qu'à remplacer `\include "definitions.ily"` par `\include "publication-web.ily"` dans notre fichier de musique.

Il est possible, bien sûr, de rendre cela encore plus pratique. Nous pourrions créer un fichier '`definitions.ily`' qui ne contiendrait que les définitions de `mpdolce` et de `inst`, un fichier '`publication-web.ily`' qui ne contiendrait que la section `layout` décrite ci-dessus et un fichier '`universite.ily`' qui ne contiendrait que les retouches pour produire le résultat que mon professeur préfère. Le début du fichier '`musique.ly`' ressemblerait alors à

```
\include "definitions.ily"

%%%  Décommentez seulement une de ces deux lignes !
\include "publication-web.ily"
%\include "universite.ily"
```

Cette approche peut être utile même si vous ne produisez qu'un seul jeu de partitions. J'utilise personnellement une demi-douzaine de fichiers de « feuille de style » pour mes projets. Je commence chaque fichier de musique par `\include "../global.ily"` qui contient :

```
%%%   global.ily
\version "2.14.2"

#(ly:set-option 'point-and-click #f)

\include "../init/init-defs.ly"
\include "../init/init-mise-en-page.ly"
\include "../init/init-en-tetes.ly"
\include "../init/init-papier.ly"
```

4.6.4 Autres sources de documentation

La référence des propriétés internes contient beaucoup d'informations sur LilyPond. Cependant vous pouvez en découvrir encore plus en consultant les fichiers internes de LilyPond. Pour cela, il vous faut d'abord connaître le répertoire *ad hoc* sur votre système. L'emplacement du répertoire dépend (a) du fait que, pour vous procurer LilyPond, vous avez téléchargé un paquet précompilé sur lilypond.org, ou vous l'avez installé grâce à votre gestionnaire de paquetages (c'est-à-dire

distribué avec Linux ou installé avec fink ou cygwin), ou encore vous l'avez compilé directement à partir des sources ; et (b) du système d'exploitation sur lequel il tourne.

Téléchargé depuis lilypond.org

- Linux

  '*INSTALLDIR*`/lilypond/usr/share/lilypond/current/`'
- MacOS X

  '*INSTALLDIR*`/LilyPond.app/Contents/Resources/share/lilypond/current/`' Pour accéder à ce dossier, deux possibilités : soit, dans un Terminal, taper `cd` suivi du chemin complet ci-dessus ; soit Control-cliquer (ou clic droit) sur l'application LilyPond et sélectionner « Afficher le contenu du paquet ».
- Windows

  Dans l'Explorateur Windows, voir '*INSTALLDIR*`/LilyPond/usr/share/lilypond/current/`'

Installé par un gestionnaire de paquetages ou compilé d'après les sources

PREFIX`/share/lilypond/`*X.Y.Z*`/`, où *PREFIX* est déterminé par votre gestionnaire de paquetages ou par le script `configure`, et *X.Y.Z* est le numéro de version de LilyPond.

Dans ce répertoire, deux sous-répertoires sont particulièrement intéressants :

- `ly/` – contient les fichiers en format LilyPond
- `scm/` – contient les fichiers en format Scheme

Commençons par examiner quelques fichiers contenus dans '`ly/`'. Nous ouvrons '`ly/property-init.ly`' dans un éditeur de texte – celui que vous avez l'habitude d'utiliser pour les fichiers '`.ly`' fera très bien l'affaire. Ce fichier contient les définitions de toutes les commandes standard prédéfinies de LilyPond, comme `\stemUp` et `\slurDotted`. Vous pouvez constater que ce n'est rien d'autre que des définitions de variables composées d'un ou plusieurs groupes de commandes `\override`. Par exemple, `\tieDotted` est défini comme :

```
tieDotted = {
  \override Tie #'dash-period = #0.75
  \override Tie #'dash-fraction = #0.1
}
```

Si vous n'aimez pas les valeurs par défaut, les commandes prédéfinies peuvent être facilement redéfinies, comme n'importe quelle autre variable, en tête de votre fichier d'entrée.

Voici les fichiers les plus utiles dans le répertoire '`ly/`' :

Nom de fichier	**Contenu**
'`ly/engraver-init.ly`'	Définitions des Contextes de graveurs
'`ly/paper-defaults-init.ly`'	Réglages papier par défaut
'`ly/performer-init.ly`'	Définitions des Contextes d'interprétation
'`ly/property-init.ly`'	Définitions de toutes les commandes prédéfinies courantes
'`ly/spanner-init.ly`'	Définitions des commandes prédéfinies pour les bandeaux

Les autres réglages (comme les définitions de commandes *markup*) sont conservés comme fichiers '`.scm`' (Scheme). Le langage de programmation Scheme offre une interface programmable dans le processus interne de LilyPond. De plus amples explications sur ces fichiers dépasseraient le cadre de ce manuel dans la mesure où elles requièrent la connaissance du langage Scheme. Les utilisateurs qui souhaiteraient comprendre le fonctionnement de ces fichiers de configuration

doivent être avertis que des connaissances techniques substantielles et beaucoup de temps sont nécessaires (voir le Section “Tutoriel Scheme” dans *Extension de LilyPond*).

Si c’est votre cas, les fichiers Scheme les plus utiles à connaître sont :

Nom de fichier	**Contenu**
‘`scm/auto-beam.scm`’	Régles par défaut des ligatures subalternes
‘`scm/define-grobs.scm`’	Réglages par défaut des propriétés de *grobs*
‘`scm/define-markup-commands.scm`’	Spécification de toutes les commandes de *markup*
‘`scm/midi.scm`’	Réglages par défaut pour les sorties MIDI
‘`scm/output-lib.scm`’	Réglages affectant l’apparence des frets, couleurs, altérations, barres de mesure, etc.
‘`scm/parser-clef.scm`’	Définition des clefs prises en charge
‘`scm/script.scm`’	Réglages par défaut des articulations

4.6.5 Options ralentissant le traitement

LilyPond peut effectuer des vérifications supplémentaires lors du traitement des fichiers ; cependant, le rendu nécessitera alors plus de temps. En contrepartie, il y aura moins d’ajustements manuels à réaliser. Si une indication textuelle ou des paroles débordent dans la marge, ces vérifications auront pour effet de resserrer la ligne suffisamment pour faire tenir le texte entre les marges.

Pour pouvoir fonctionner en toutes circonstances, ces vérifications doivent être activées ; il faut pour cela placer les retouches dans un `\context` à l’intérieur d’un bloc `\layout`, plutôt que directement dans le fragment musical, comme ceci :

```
\score {
  { ...notes... }
  \layout {
    \context {
      \Score
      % Pour s'assurer que les indications textuelles et paroles
      % seront à l'intérieur des marges de la page.
      \override PaperColumn #'keep-inside-line = ##t
      \override NonMusicalPaperColumn #'keep-inside-line = ##t
    }
  }
}
```

4.6.6 Retouches avancées avec Scheme
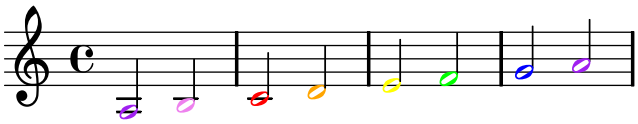
Nous avons vu à quel point le résultat obtenu avec LilyPond peut être largement personnalisé à l’aide de commandes comme `\override` et `\tweak`. Et pourtant l’utilisation de Scheme ouvre des possibilités encore plus grandes. Le code écrit dans le langage de programmation Scheme peut être intégré directement dans le processus interne de LilyPond. Bien sûr, il faut pour cela connaître un minimum de programmation en langage Scheme. Pour des explications complètes là-dessus, consultez le Section “Tutoriel Scheme” dans *Extension de LilyPond*.

En guise d’illustration – et ce n’est qu’une possibilité parmi tant d’autres – nous allons attribuer à une propriété non pas une valeur préétablie mais une procédure Scheme qui sera lancée à chaque utilisation de la propriété par LilyPond. De cette façon, nous obtenons un réglage dynamique de la propriété par le simple fait d’invoquer la procédure. Dans cet exemple, nous colorons les têtes de notes en fonction de leur position sur la portée.

```
#(define (color-notehead grob)
   "Color the notehead according to its position on the staff."
```

```
    (let ((mod-position (modulo (ly:grob-property grob 'staff-position)
                                7)))
      (case mod-position
        ;;   Return rainbow colors
        ((1) (x11-color 'red    ))  ; for C
        ((2) (x11-color 'orange ))  ; for D
        ((3) (x11-color 'yellow ))  ; for E
        ((4) (x11-color 'green  ))  ; for F
        ((5) (x11-color 'blue   ))  ; for G
        ((6) (x11-color 'purple ))  ; for A
        ((0) (x11-color 'violet ))  ; for B
        )))

\relative c' {
  % Arrange to obtain color from color-notehead procedure
  \override NoteHead #'color = #color-notehead
  a2 b | c2 d | e2 f | g2 a |
}
```



Vous trouverez dans Section “les fonctions callback” dans *Extension de LilyPond* d’autres exemples d’utilisation de ces interfaces programmables.

Annexe A Modèles

Cette annexe du manuel d'initiation propose des patrons de partition Lilypond, prets à l'emploi. Il vous suffira d'y ajouter quelques notes, de lancer LilyPond, et d'aprécier le résultat.

A.1 Portée unique
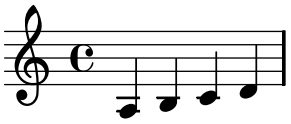
A.1.1 Notes seules

Cet exemple simpliste se compose d'une portée agrémentée de quelques notes. Il convient tout à fait pour un instrument seul ou un fragment mélodique. Recopiez-le dans un nouveau fichier, ajoutez-y d'autres notes et c'est pret !

```
\version "2.14.2"
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

\score {
  \new Staff \melody
  \layout { }
  \midi { }
}
```



A.1.2 Notes et paroles

Ce canevas comporte une simple ligne mélodique agrémentée de paroles. Recopiez-le, ajoutez-y d'autres notes et paroles. Les ligatures automatiques sont ici désactivées, comme il est d'usage en matière de musique vocale. Pour activer la fonction de ligature automatique, modifiez ou commentez la ligne en question.

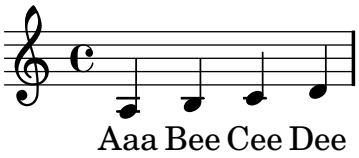
```
\version "2.14.2"
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score{
```

```
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melody
    }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



A.1.3 Notes et accords
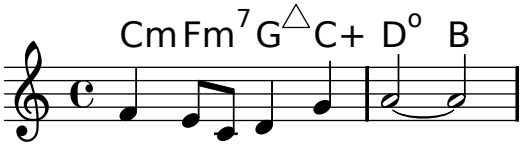
Vous avez besoin de la partition d'une mélodie avec les accords ? N'allez pas plus loin !

```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  f4 e8[ c] d4 g
  a2 ~ a
}

harmonies = \chordmode {
  c4:m f:min7 g:maj c:aug
  d2:dim b:sus
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Staff \melody
  >>
  \layout{ }
  \midi { }
}
```



A.1.4 Notes, paroles et accords

Ce cannevas comporte tous les éléments d'une chanson : la mélodie, les paroles, les accords.

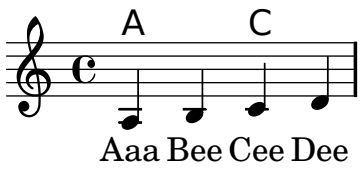
```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

harmonies = \chordmode {
  a2 c
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Voice = "one" { \autoBeamOff \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



A.2 Modèles pour piano

A.2.1 Piano seul

Voici une simple partition pour piano avec quelques notes.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4
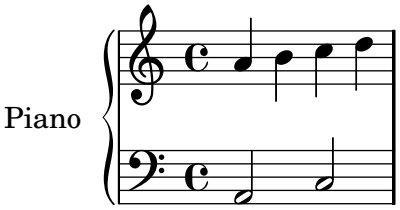
  a4 b c d
}

lower = \relative c {
```

```
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  \new PianoStaff <<
    \set PianoStaff.instrumentName = #"Piano  "
    \new Staff = "upper" \upper
    \new Staff = "lower" \lower
  >>
  \layout { }
  \midi { }
}
```



A.2.2 Chant et accompagnement

Il s'agit du format classique pour le chant : une portée pour la mélodie et les paroles au-dessus de l'accompagnement au piano.

```
melody = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

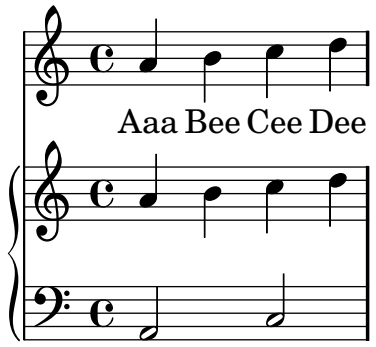
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4
```

```
  a2 c
}

\score {
  <<
    \new Voice = "mel" { \autoBeamOff \melody }
    \new Lyrics \lyricsto mel \text
    \new PianoStaff <<
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
  }
  \midi { }
}
```



A.2.3 Piano et paroles entre les portées

Lorsque la mélodie est doublée au piano, cela ne nécessite pas forcément une portée spécifique. Les paroles peuvent s'insérer entre les deux portées de la partition pour piano.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

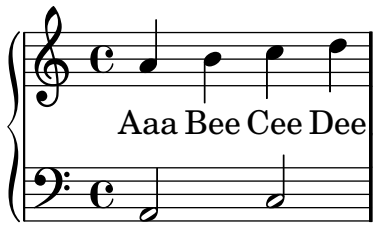
lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

text = \lyricmode {
  Aaa Bee Cee Dee
```

```
}

\score {
  \new GrandStaff <<
    \new Staff = upper { \new Voice = "singer" \upper }
    \new Lyrics \lyricsto "singer" \text
    \new Staff = lower { \lower }
  >>
  \layout {
    \context {
      \GrandStaff
      \accepts "Lyrics"
    }
    \context {
      \Lyrics
      \consists "Bar_engraver"
    }
  }
  \midi { }
}
```



A.2.4 Piano et nuances entre les portées

Nombre d'ouvrages pour piano font apparaître les nuances entre les deux portées. Bien que cela nécessite quelques subtilités, voici de quoi obtenir un tel résultat.

```
global = {
  \key c \major
  \time 4/4
}

upper = \relative c'' {
  \clef treble
  a4 b c d
}

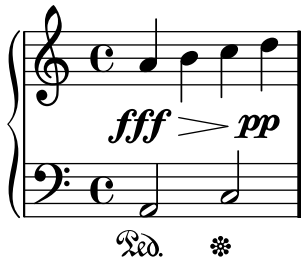
lower = \relative c {
  \clef bass
  a2 c
}

dynamics = {
  s2\fff\> s4 s\!\pp
}
```

```
pedal = {
  s2\sustainOn s\sustainOff
}

\score {
  \new PianoStaff = "PianoStaff_pf" <<
    \new Staff = "Staff_pfUpper" << \global \upper >>
    \new Dynamics = "Dynamics_pf" \dynamics
    \new Staff = "Staff_pfLower" << \global \lower >>
    \new Dynamics = "pedal" \pedal
  >>
  \layout { }
}

\score {
  \new PianoStaff = "PianoStaff_pf" <<
    \new Staff = "Staff_pfUpper" << \global \upper \dynamics \pedal >>
    \new Staff = "Staff_pfLower" << \global \lower \dynamics \pedal >>
  >>
  \midi { }
}
```



A.3 Quatuor à cordes

A.3.1 Quatuor à cordes

Voici un canevas pour quatuor à cordes. Notez l'utilisation de la variable `\global` pour traiter la métrique et la tonalité.

```
global= {
  \time 4/4
  \key c \major
}

violinOne = \new Voice \relative c'' {
  \set Staff.instrumentName = #"Violin 1 "

  c2 d
  e1

  \bar "|."
}

violinTwo = \new Voice \relative c'' {
  \set Staff.instrumentName = #"Violin 2 "
```

```
  g2 f
  e1

  \bar "|."
}

viola = \new Voice \relative c' {
  \set Staff.instrumentName = #"Viola "
  \clef alto

  e2 d
  c1

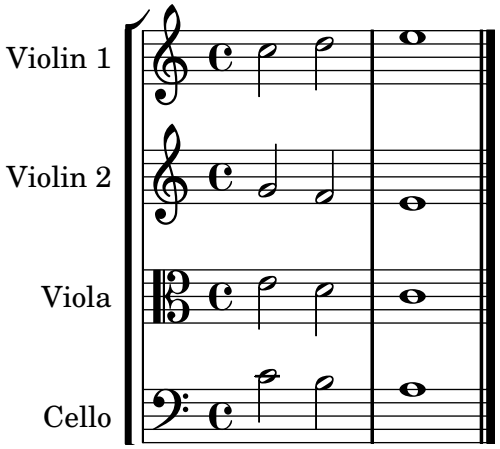
  \bar "|."
}

cello = \new Voice \relative c' {
  \set Staff.instrumentName = #"Cello "
  \clef bass

  c2 b
  a1

  \bar "|."
}

\score {
  \new StaffGroup <<
    \new Staff << \global \violinOne >>
    \new Staff << \global \violinTwo >>
    \new Staff << \global \viola >>
    \new Staff << \global \cello >>
  >>
  \layout { }
  \midi { }
}
```



A.3.2 Parties pour quatuor à cordes

Grâce à ce canevas, vous pouvez obtenir une partition d'excellente facture pour quatuor à cordes mais aussi, si le besoin s'en faisait sentir, une partie séparée par instrument. Par ailleurs, cet exemple illustre l'utilisation de la fonction `\tag` dans le but d'extraire des parties séparées.

Il vous faudra découper ce canevas en plusieurs fichiers séparés ; leur nom respectif est indiqué en commentaire : '`piece.ly`' comporte tout ce qui a trait à la musique, les autres fichiers – '`score.ly`', '`vn1.ly`', '`vn2.ly`', '`vla.ly`', et '`vlc.ly`' – vous permettront d'obtenir les parties selon le pupitre.

N'oubliez pas de supprimer les commentaires superflus des fichiers individualisés !

```
%%%%% piece.ly
%%%%% (This is the global definitions file)

global= {
  \time 4/4
  \key c \major
}

Violinone = \new Voice { \relative c''{
  \set Staff.instrumentName = #"Violin 1 "

  c2 d e1

\bar "|." }}   %*********************************
Violintwo = \new Voice { \relative c''{
  \set Staff.instrumentName = #"Violin 2 "

  g2 f e1

\bar "|." }}   %*********************************
Viola = \new Voice { \relative c' {
  \set Staff.instrumentName = #"Viola "
  \clef alto

  e2 d c1

\bar "|." }}   %*********************************
Cello = \new Voice { \relative c' {
  \set Staff.instrumentName = #"Cello "
  \clef bass

  c2 b a1

\bar "|."}}   %*********************************

music = {
  <<
    \tag #'score \tag #'vn1 \new Staff { << \global \Violinone >> }
    \tag #'score \tag #'vn2 \new Staff { << \global \Violintwo>> }
    \tag #'score \tag #'vla \new Staff { << \global \Viola>> }
    \tag #'score \tag #'vlc \new Staff { << \global \Cello>> }
  >>
```

```
}

%%% These are the other files you need to save on your computer

%%%%% score.ly
%%%%% (This is the main file)


%\include "piece.ly"             %%% uncomment this line when using a separate file
#(set-global-staff-size 14)
\score {
  \new StaffGroup \keepWithTag #'score \music
  \layout { }
  \midi { }
}


%{ Uncomment this block when using separate files

%%%%% vn1.ly
%%%%% (This is the Violin 1 part file)

\include "piece.ly"
\score {
  \keepWithTag #'vn1 \music
  \layout { }
}


%%%%% vn2.ly
%%%%% (This is the Violin 2 part file)

\include "piece.ly"
\score {
  \keepWithTag #'vn2 \music
  \layout { }
}

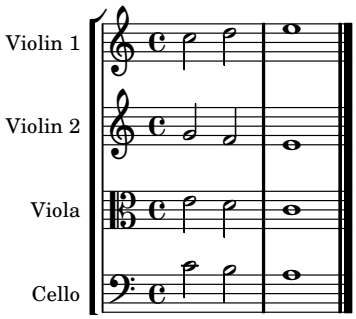

%%%%% vla.ly
%%%%% (This is the Viola part file)

\include "piece.ly"
\score {
  \keepWithTag #'vla \music
  \layout { }
}


%%%%% vlc.ly
%%%%% (This is the Cello part file)
```

```
\include "piece.ly"
\score {
  \keepWithTag #'vlc \music
  \layout { }
}

%}
```



A.4 Ensemble vocal

A.4.1 Partition pour chœur à quatre voix mixtes

Ce fichier constitue un canevas standard de partition pour chœur à quatre voix mixtes. Lorsque les ensembles s'étoffent, il est judicieux de recourrir à une section spécifique incluse dans chacune des parties, tout particulièrement pour gérer la métrique et la tonalité qui, la plupart du temps, sont communes à tous les pupitres. Comme il est d'usage pour les hymnes, les quatre voix sont réparties sur deux portées.

```
\paper {
  top-system-spacing #'basic-distance = #10
  score-system-spacing #'basic-distance = #20
  system-system-spacing #'basic-distance = #20
  last-bottom-spacing #'basic-distance = #10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
```
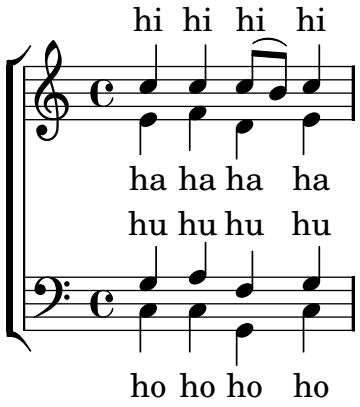
```
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Lyrics = "sopranos" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup #'staff-affinity = #DOWN
    }
    \new Staff = "women" <<
      \new Voice = "sopranos" {
        \voiceOne
        << \global \sopMusic >>
      }
      \new Voice = "altos" {
        \voiceTwo
        << \global \altoMusic >>
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup #'staff-affinity = #DOWN
    }
    \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" \sopWords
    \context Lyrics = "altos" \lyricsto "altos" \altoWords
    \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
    \context Lyrics = "basses" \lyricsto "basses" \bassWords
```

```
  >>
}
```



A.4.2 Partition pour chœur SATB avec réduction pour piano

Ce canevas ajoute une réduction pour piano à une partition standard pour chœur à quatre voix mixtes. Ceci illustre l'un des avantages de LilyPond : une expression musicale peut être réutilisée sans effort. Toute modification apportée à l'une des voix, mettons `tenorMusique`, sera automatiquement reportée dans la réduction pour piano.

```
\paper {
  top-system-spacing #'basic-distance = #10
  score-system-spacing #'basic-distance = #20
  system-system-spacing #'basic-distance = #20
  last-bottom-spacing #'basic-distance = #10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
}
altoWords =\lyricmode {
  ha ha ha ha
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
```

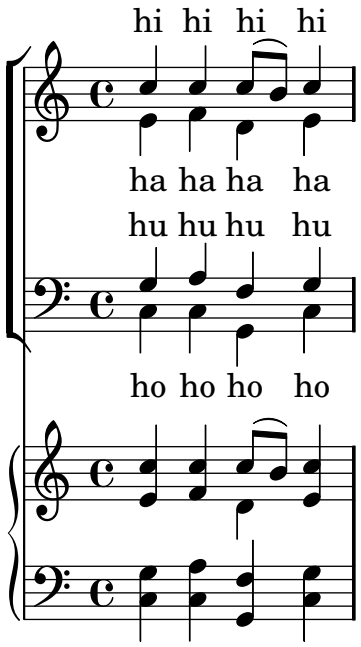
```
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  <<
    \new ChoirStaff <<
      \new Lyrics = "sopranos" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup #'staff-affinity = #DOWN
      }
      \new Staff = "women" <<
        \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
        \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
      >>
      \new Lyrics = "altos"
      \new Lyrics = "tenors" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup #'staff-affinity = #DOWN
      }

      \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" \sopWords
      \context Lyrics = "altos" \lyricsto "altos" \altoWords
      \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
      \context Lyrics = "basses" \lyricsto "basses" \bassWords
    >>
    \new PianoStaff <<
      \new Staff <<
        \set Staff.printPartCombineTexts = ##f
        \partcombine
        << \global \sopMusic >>
        << \global \altoMusic >>
      >>
      \new Staff <<
        \clef bass
        \set Staff.printPartCombineTexts = ##f
        \partcombine
        << \global \tenorMusic >>
        << \global \bassMusic >>
      >>
```

```
    >>
  >>
}
```



A.4.3 Partition pour chœur SATB avec alignement des contextes

Ce canevas ressemble beaucoup à celui pour chœur à quatre voix mixtes. La différence réside dans le fait que les paroles sont positionnées en ayant recours à `alignAboveContext` et `alignBelowContext`.

```
global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}
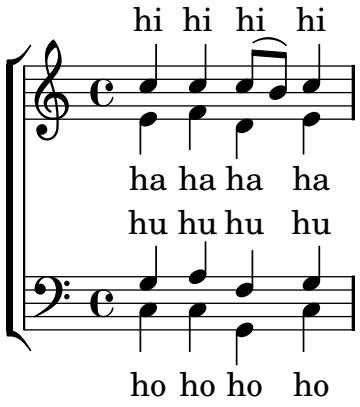
tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
```

```
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Staff = "women" <<
      \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
      \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = #"women" } \lyricsto "sopranos" \sopWords
    \new Lyrics \with { alignBelowContext = #"women" } \lyricsto "altos" \altoWords
    % we could remove the line about this with the line below, since we want
    % the alto lyrics to be below the alto Voice anyway.
    % \new Lyrics \lyricsto "altos" \altoWords

    \new Staff = "men" <<
      \clef bass
      \new Voice = "tenors" { \voiceOne << \global \tenorMusic >> }
      \new Voice = "basses" { \voiceTwo << \global \bassMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = #"men" } \lyricsto "tenors" \tenorWords
    \new Lyrics \with { alignBelowContext = #"men" } \lyricsto "basses" \bassWords
    % again, we could replace the line above this with the line below.
    % \new Lyrics \lyricsto "basses" \bassWords
  >>
}
```



A.4.4 Partition pour chœur SATB, sur quatre portées
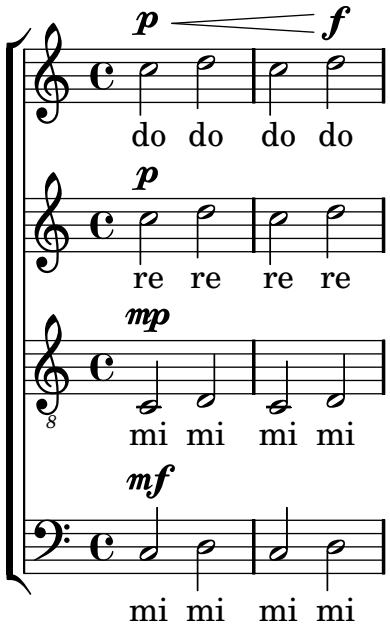
Modèle pour chœur à quatre voix mixtes, chaque pupitre ayant sa propre portée.

```
global = {
  \key c \major
  \time 4/4
```

```
  \dynamicUp
}
sopranonotes = \relative c'' {
  c2 \p \< d c d \f
}
sopranowords = \lyricmode { do do do do }
altonotes = \relative c'' {
  c2\p d c d
}
altowords = \lyricmode { re re re re }
tenornotes = {
  \clef "G_8"
  c2\mp d c d
}
tenorwords = \lyricmode { mi mi mi mi }
bassnotes = {
  \clef bass
  c2\mf d c d
}
basswords = \lyricmode { mi mi mi mi }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "soprano" <<
        \global
        \sopranonotes
      >>
      \lyricsto "soprano" \new Lyrics \sopranowords
    >>
    \new Staff <<
      \new Voice = "alto" <<
        \global
        \altonotes
      >>
      \lyricsto "alto" \new Lyrics \altowords
    >>
    \new Staff <<
      \new Voice = "tenor" <<
        \global
        \tenornotes
      >>
      \lyricsto "tenor" \new Lyrics \tenorwords
    >>
    \new Staff <<
      \new Voice = "bass" <<
        \global
        \bassnotes
      >>
      \lyricsto "bass" \new Lyrics \basswords
    >>
  >>
```

```
}
```



A.4.5 Couplet pour solo et refrain à deux voix

Ce canevas illustre la manière d'agencer une œuvre vocale où le couplet est chanté en solo et le refrain à deux voix. Vous noterez le recours aux silences invisibles dans la variable `\global` ; ils permettent de positionner les changements de métrique et autres éléments communs à toutes les parties et pour l'intégralité du morceau.

```
global = {
  \key g \major

  % verse
  \time 3/4
  s2.*2
  \break

  % refrain
  \time 2/4
  s2*2
  \bar "|."
}

SoloNotes = \relative g' {
  \clef "treble"

  % verse
  g4 g g |
  b4 b b |

  % refrain
  R2*2 |
}
```

```
SoloLyrics = \lyricmode {
  One two three |
  four five six |
}

SopranoNotes = \relative c'' {
  \clef "treble"

  % verse
  R2.*2 |

  % refrain
  c4 c |
  g4 g |
}

SopranoLyrics = \lyricmode {
  la la |
  la la |
}

BassNotes = \relative c {
  \clef "bass"

  % verse
  R2.*2 |

  % refrain
  c4 e |
  d4 d |
}
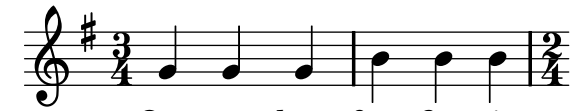
BassLyrics = \lyricmode {
  dum dum |
  dum dum |
}

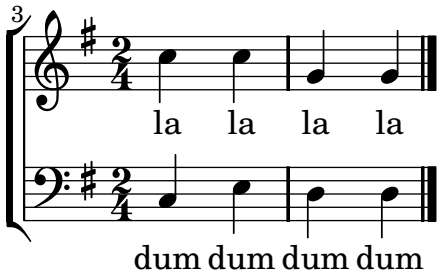
\score {
  <<
    \new Voice = "SoloVoice" << \global \SoloNotes >>
    \new Lyrics \lyricsto "SoloVoice" \SoloLyrics

    \new ChoirStaff <<
      \new Voice = "SopranoVoice" << \global \SopranoNotes >>
      \new Lyrics \lyricsto "SopranoVoice" \SopranoLyrics

      \new Voice = "BassVoice" << \global \BassNotes >>
      \new Lyrics \lyricsto "BassVoice" \BassLyrics
    >>
  >>
  \layout {
    ragged-right = ##t
```

```
    \context { \Staff
      % these lines prevent empty staves from being printed
      \RemoveEmptyStaves
      \override VerticalAxisGroup #'remove-first = ##t
    }
  }
}
```





A.4.6 Hymnes et cantiques

Le code ci-dessous illustre la manière d'agencer un cantique liturgique dans lequel chaque ligne débute et se termine par une mesure incomplète. Vous noterez par ailleurs l'affichage des paroles indépendamment de la musique.

```
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
}

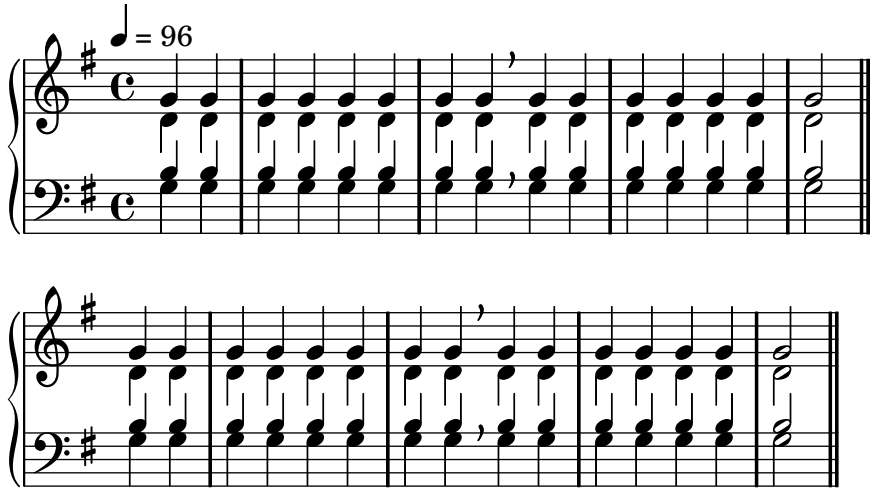
global = {
 \key g \major
}

\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

A.4.7 Psalmodie

Cet exemple illustre la manière de présenter un cantique tel qu'on le trouve dans l'église anglicane. Vous noterez comment sont ajoutés les couplets indépendamment de la musique. Dans le but de vous montrer plusieurs styles, comparez le code des deux couplets.

```
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
}

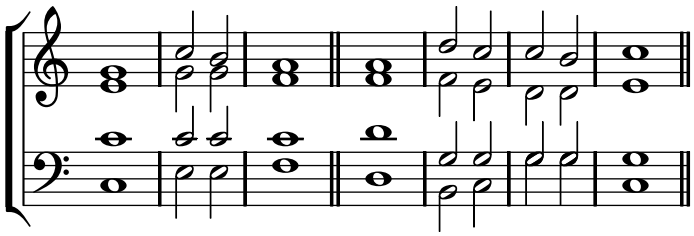
dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}

tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}

% 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

\markup {
  \fill-line {
    \column {
      \left-align {
        \null \null \null
        \line {
          \fontsize #5 O
          \fontsize #3 come
          let us \bold sing | unto \dot the | Lord : let
        }
        \line {
          us heartily
          \concat { re \bold joice }
          in the | strength of | our
        }
        \line {
          sal | vation.
        }
        \null
        \line {
          \hspace #2.5 8. Today if ye will hear his voice *
        }
        \line {
          \concat { \bold hard en }
          \tick not your \tick hearts : as in the pro-
        }
        \line {
          vocation * and as in the \bold day of tempt- \tick
        }
        \line {
          -ation \tick in the \tick wilderness.
        }
      }
    }
  }
}
```



O come let us **sing** | unto • the | Lord : let us heartily re**joice** in the | strength of | our sal | vation.

8. Today if ye will hear his voice * **hard**en ' not your ' hearts : as in the provocation * and as in the **day** of tempt- ' -ation ' in the ' wilderness.

A.5 Orchestre

A.5.1 Orchestre, chœur et piano

Ce canevas illustre l'utilisation de contextes `StaffGroup` pour regrouper les instruments selon leur famille, imbriqués dans un `GrandStaff`, ainsi que le recours à la fonction `\transpose` pour les instruments transpositeurs. Dans tous les identificateurs, la musique est stockée en ut. Les notes peuvent tout aussi bien être saisies en ut ou dans la tonalité particulière de l'instrument avant d'être transposées puis affectées à une variable.

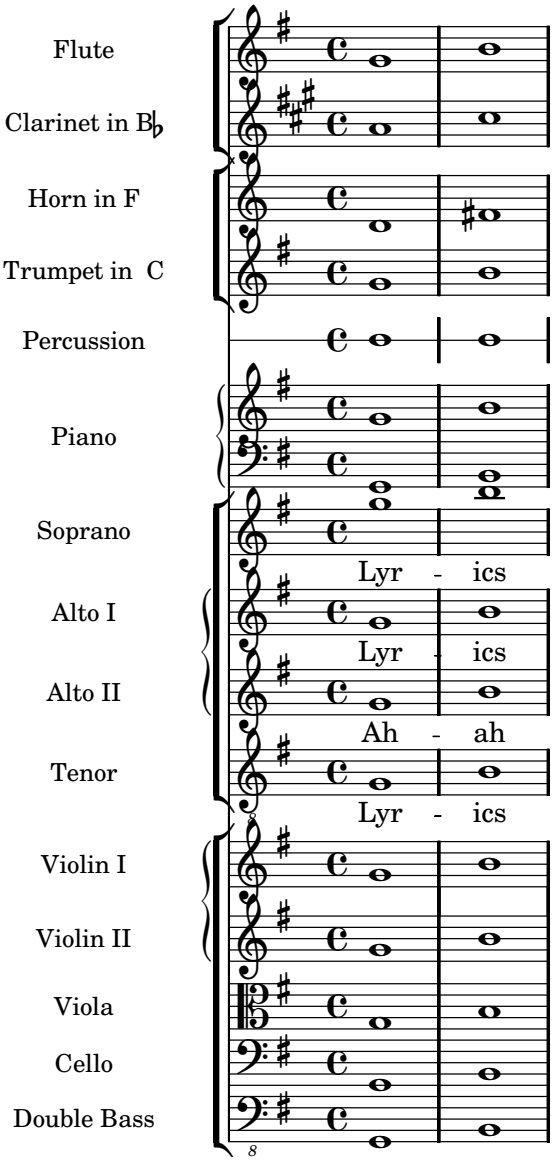
```
#(set-global-staff-size 17)
\paper {
  indent = 3.0\cm  % space for instrumentName
  short-indent = 1.5\cm  % space for shortInstrumentName
}

fluteMusic = \relative c' { \key g \major g'1 b }
% Pitches as written on a manuscript for Clarinet in A
% are transposed to concert pitch.
clarinetMusic = \transpose c' a
  \relative c'' { \key bes \major bes1 d }
trumpetMusic = \relative c { \key g \major g''1 b }
% Key signature is often omitted for horns
hornMusic = \transpose c' f
  \relative c { d'1 fis }
percussionMusic = \relative c { \key g \major g1 b }
sopranoMusic = \relative c'' { \key g \major g'1 b }
sopranoLyrics = \lyricmode { Lyr -- ics }
altoIMusic = \relative c' { \key g \major g'1 b }
altoIIMusic = \relative c' { \key g \major g'1 b }
altoILyrics =  \sopranoLyrics
altoIILyrics = \lyricmode { Ah -- ah }
tenorMusic = \relative c' { \clef "treble_8" \key g \major g1 b }
tenorLyrics = \sopranoLyrics
pianoRHMusic = \relative c { \key g \major g''1 b }
```

```
pianoLHMusic = \relative c { \clef bass \key g \major g1 b }
violinIMusic = \relative c' { \key g \major g'1 b }
violinIIMusic = \relative c' { \key g \major g'1 b }
violaMusic = \relative c { \clef alto \key g \major g'1 b }
celloMusic = \relative c { \clef bass \key g \major g1 b }
bassMusic = \relative c { \clef "bass_8" \key g \major g,1 b }

\score {
  <<
    \new StaffGroup = "StaffGroup_woodwinds" <<
      \new Staff = "Staff_flute" {
        \set Staff.instrumentName = #"Flute"
        % shortInstrumentName, midiInstrument, etc.
        % may be set here as well
        \fluteMusic
      }
      \new Staff = "Staff_clarinet" {
        \set Staff.instrumentName =
        \markup { \concat { "Clarinet in B" \flat } }
        % Declare that written Middle C in the music
        %  to follow sounds a concert B flat, for
        %  output using sounded pitches such as MIDI.
        \transposition bes
        % Print music for a B-flat clarinet
        \transpose bes c' \clarinetMusic
      }
    >>
    \new StaffGroup = "StaffGroup_brass" <<
      \new Staff = "Staff_hornI" {
        \set Staff.instrumentName = #"Horn in F"
        \transposition f
        \transpose f c' \hornMusic
      }
      \new Staff = "Staff_trumpet" {
        \set Staff.instrumentName = #"Trumpet in  C"
        \trumpetMusic
      }
    >>
    \new RhythmicStaff = "RhythmicStaff_percussion" <<
      \set RhythmicStaff.instrumentName = #"Percussion"
      \percussionMusic
    >>
    \new PianoStaff <<
      \set PianoStaff.instrumentName = #"Piano"
      \new Staff { \pianoRHMusic }
      \new Staff { \pianoLHMusic }
    >>
    \new ChoirStaff = "ChoirStaff_choir" <<
      \new Staff = "Staff_soprano" {
        \set Staff.instrumentName = #"Soprano"
        \new Voice = "soprano"
        \sopranoMusic
```

```
        }
        \new Lyrics \lyricsto "soprano" { \sopranoLyrics }
        \new GrandStaff = "GrandStaff_altos"
        \with { \accepts Lyrics } <<
          \new Staff = "Staff_altoI"  {
            \set Staff.instrumentName = #"Alto I"
            \new Voice = "altoI"
            \altoIMusic
          }
          \new Lyrics \lyricsto "altoI" { \altoILyrics }
          \new Staff = "Staff_altoII" {
            \set Staff.instrumentName = #"Alto II"
            \new Voice = "altoII"
            \altoIIMusic
          }
          \new Lyrics \lyricsto "altoII" { \altoIILyrics }
        >>
        \new Staff = "Staff_tenor" {
          \set Staff.instrumentName = #"Tenor"
          \new Voice = "tenor"
          \tenorMusic
        }
        \new Lyrics \lyricsto "tenor" { \tenorLyrics }
      >>
      \new StaffGroup = "StaffGroup_strings" <<
        \new GrandStaff = "GrandStaff_violins" <<
          \new Staff = "Staff_violinI" {
            \set Staff.instrumentName = #"Violin I"
            \violinIMusic
          }
          \new Staff = "Staff_violinII" {
            \set Staff.instrumentName = #"Violin II"
            \violinIIMusic
          }
        >>
        \new Staff = "Staff_viola" {
          \set Staff.instrumentName = #"Viola"
          \violaMusic
        }
        \new Staff = "Staff_cello" {
          \set Staff.instrumentName = #"Cello"
          \celloMusic
        }
        \new Staff = "Staff_bass" {
          \set Staff.instrumentName = #"Double Bass"
          \bassMusic
        }
      >>
    >>
    \layout { }
  }
```



A.6 Exemples de notation ancienne

A.6.1 Transcription de musique mensurale

Lorsque l'on transcrit de la musique mensurale, un *incipit* permet d'indiquer la tonalité et le tempo d'origine. Si les musiciens sont de nos jours habitués aux barres de mesures qui présentent la structure rythmique d'une œuvre, elles n'étaient pas en vigueur à l'époque où ces pièces ont été composées, d'autant plus que la « métrique » pouvait changer au fil des notes. Un compromis consiste à imprimer des barres entre les portées plutôt que sur chacune d'elles.

```
global = {
  \set Score.skipBars = ##t

  % incipit
  \once \override Score.SystemStartBracket #'transparent = ##t
  \override Score.SpacingSpanner #'spacing-increment = #1.0 % tight spacing
  \key f \major
  \time 2/2
  \once \override Staff.TimeSignature #'style = #'neomensural
  \override Voice.NoteHead #'style = #'neomensural
  \override Voice.Rest #'style = #'neomensural
  \set Staff.printKeyCancellation = ##f
```

```
  \cadenzaOn % turn off bar lines
  \skip 1*10
  \once \override Staff.BarLine #'transparent = ##f
  \bar "||"
  \skip 1*1 % need this extra \skip such that clef change comes
            % after bar line
  \bar ""

  % main
  \revert Score.SpacingSpanner #'spacing-increment % CHECK: no effect?
  \cadenzaOff % turn bar lines on again
  \once \override Staff.Clef #'full-size-change = ##t
  \set Staff.forceClef = ##t
  \key g \major
  \time 4/4
  \override Voice.NoteHead #'style = #'default
  \override Voice.Rest #'style = #'default

  % FIXME: setting printKeyCancellation back to #t must not
  % occur in the first bar after the incipit.  Dto. for forceClef.
  % Therefore, we need an extra \skip.
  \skip 1*1
  \set Staff.printKeyCancellation = ##t
  \set Staff.forceClef = ##f

  \skip 1*7 % the actual music

  % let finis bar go through all staves
  \override Staff.BarLine #'transparent = ##f

  % finis bar
  \bar "|."
}

discantusNotes = {
  \transpose c' c'' {
    \set Staff.instrumentName = #"Discantus  "

    % incipit
    \clef "neomensural-c1"
    c'1. s2   % two bars
    \skip 1*8 % eight bars
    \skip 1*1 % one bar

    % main
    \clef "treble"
    d'2. d'4 |
    b e' d'2 |
    c'4 e'4.( d'8 c' b |
    a4) b a2 |
    b4.( c'8 d'4) c'4 |
    \once \override NoteHead #'transparent = ##t c'1 |
```

```
    b\breve |
  }
}

discantusLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi -- |
  la -- te De -- |
  o, om --
  nis ter -- |
  ra, __ om- |
  "..." |
  -us. |
}

altusNotes = {
  \transpose c' c'' {
    \set Staff.instrumentName = #"Altus  "

    % incipit
    \clef "neomensural-c3"
    r1        % one bar
    f1. s2    % two bars
    \skip 1*7 % seven bars
    \skip 1*1 % one bar

    % main
    \clef "treble"
    r2 g2. e4 fis g | % two bars
    a2 g4 e |
    fis g4.( fis16 e fis4) |
    g1 |
    \once \override NoteHead #'transparent = ##t g1 |
    g\breve |
  }
}

altusLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi -- la -- te | % two bars
  De -- o, om -- |
  nis ter -- ra, |
  "..." |
  -us. |
}
```

```
tenorNotes = {
  \transpose c' c' {
    \set Staff.instrumentName = #"Tenor  "

    % incipit
    \clef "neomensural-c4"
    r\longa   % four bars
    r\breve   % two bars
    r1        % one bar
    c'1. s2   % two bars
    \skip 1*1 % one bar
    \skip 1*1 % one bar

    % main
    \clef "treble_8"
    R1 |
    R1 |
    R1 |
    r2 d'2. d'4 b e' | % two bars
    \once \override NoteHead #'transparent = ##t e'1 |
    d'\breve |
  }
}

tenorLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi -- la -- te | % two bars
  "..." |
  -us. |
}

bassusNotes = {
  \transpose c' c' {
    \set Staff.instrumentName = #"Bassus  "

    % incipit
    \clef "bass"
    r\maxima  % eight bars
    f1. s2    % two bars
    \skip 1*1 % one bar

    % main
    \clef "bass"
    R1 |
    R1 |
    R1 |
    R1 |
    g2. e4 |
    \once \override NoteHead #'transparent = ##t e1 |
```

```
    g\breve |
  }
}

bassusLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi- |
  "..." |
  -us. |
}

\score {
  \new StaffGroup = choirStaff <<
    \new Voice =
      "discantusNotes" << \global \discantusNotes >>
    \new Lyrics =
      "discantusLyrics" \lyricsto discantusNotes { \discantusLyrics }
    \new Voice =
      "altusNotes" << \global \altusNotes >>
    \new Lyrics =
      "altusLyrics" \lyricsto altusNotes { \altusLyrics }
    \new Voice =
      "tenorNotes" << \global \tenorNotes >>
    \new Lyrics =
      "tenorLyrics" \lyricsto tenorNotes { \tenorLyrics }
    \new Voice =
      "bassusNotes" << \global \bassusNotes >>
    \new Lyrics =
      "bassusLyrics" \lyricsto bassusNotes { \bassusLyrics }
  >>
  \layout {
    \context {
      \Score
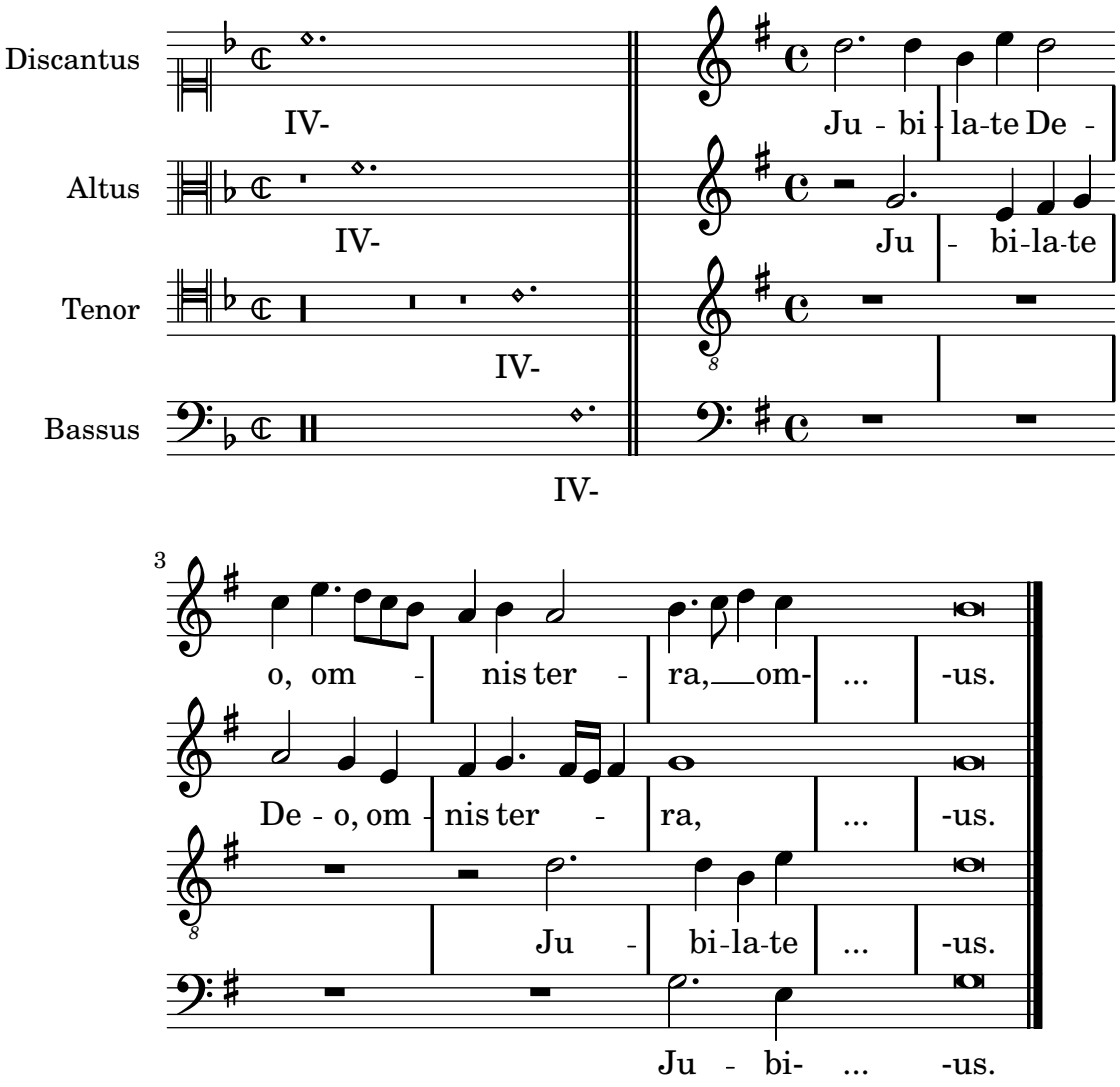
      % no bars in staves
      \override BarLine #'transparent = ##t

      % incipit should not start with a start delimiter
      \remove "System_start_delimiter_engraver"
    }
    \context {
      \Voice

      % no slurs
      \override Slur #'transparent = ##t

      % Comment in the below "\remove" command to allow line
      % breaking also at those barlines where a note overlaps
      % into the next bar.  The command is commented out in this
```

```
      % short example score, but especially for large scores, you
      % will typically yield better line breaking and thus improve
      % overall spacing if you comment in the following command.
      %\remove "Forbid_line_break_engraver"
    }
  }
}
```



A.6.2 Transcription du grégorien

Voici comment vous pourriez transcrire du grégorien. Pour mémoire, il n'y a en grégorien ni de découpage en mesure, ni de hampe ; seules sont utilisées des têtes de note blanches ou noires, ainsi que des signes spécifiques permettant d'indiquer des silences de différentes durées.
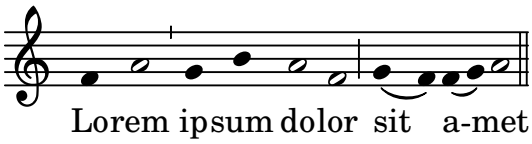
```
\include "gregorian.ly"

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
    }
  }
}
```



A.7 Autres modèles

A.7.1 Symboles de jazz

Bien que compliqué de prime abord, voici un canevas tout à fait indiqué pour les ensembles de jazz. Vous noterez que tous les instruments sont notés en ut (`\key c \major`), la tonalité de concert. Les notes seront automatiquement transposée dès lors qu'elles seront inscrites dans une section `\transpose`.

```
\header {
  title = "Song"
  subtitle = "(tune)"
  composer = "Me"
  meter = "moderato"
  piece = "Swing"
```

```
  tagline = \markup {
    \column {
      "LilyPond example file by Amelie Zapf,"
      "Berlin 07/07/2003"
    }
  }
}

%#(set-global-staff-size 16)
\include "english.ly"

%%%%%%%%%%% Some macros %%%%%%%%%%%%%%%%%%

sl = {
  \override NoteHead #'style = #'slash
  \override Stem #'transparent = ##t
}
nsl = {
  \revert NoteHead #'style
  \revert Stem #'transparent
}
crOn = \override NoteHead #'style = #'cross
crOff = \revert NoteHead #'style

%% insert chord name style stuff here.

jazzChords = { }

%%%%%%%%%%% Keys'n'thangs %%%%%%%%%%%%%%%%%

global = { \time 4/4 }

Key = { \key c \major }

% ############ Horns ############

% ------ Trumpet ------
trpt = \transpose c d \relative c'' {
  \Key
  c1 | c | c |
}
trpHarmony = \transpose c' d {
  \jazzChords
}
trumpet = {
  \global
  \set Staff.instrumentName = #"Trumpet"
  \clef treble
  <<
    \trpt
  >>
}
```

```
% ------ Alto Saxophone ------
alto = \transpose c a \relative c' {
  \Key
  c1 | c | c |
}
altoHarmony = \transpose c' a {
  \jazzChords
}
altoSax = {
  \global
  \set Staff.instrumentName = #"Alto Sax"
  \clef treble
  <<
    \alto
  >>
}

% ------ Baritone Saxophone ------
bari = \transpose c a' \relative c {
  \Key
  c1
  c1
  \sl
  d4^"Solo" d d d
  \nsl
}
bariHarmony = \transpose c' a \chordmode {
  \jazzChords s1 s d2:maj e:m7
}
bariSax = {
  \global
  \set Staff.instrumentName = #"Bari Sax"
  \clef treble
  <<
    \bari
  >>
}

% ------ Trombone ------
tbone = \relative c {
  \Key
  c1 | c | c
}
tboneHarmony = \chordmode {
  \jazzChords
}
trombone = {
  \global
  \set Staff.instrumentName = #"Trombone"
  \clef bass
  <<
```

```
    \tbone
  >>
}

% ############ Rhythm Section #############

% ------ Guitar ------
gtr = \relative c'' {
  \Key
  c1
  \sl
  b4 b b b
  \nsl
  c1
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 c2:min7+ d2:maj9
}
guitar = {
  \global
  \set Staff.instrumentName = #"Guitar"
  \clef treble
  <<
    \gtr
  >>
}

%% ------ Piano ------
rhUpper = \relative c'' {
  \voiceOne
  \Key
  c1 | c | c
}
rhLower = \relative c' {
  \voiceTwo
  \Key
  e1 | e | e
}

lhUpper = \relative c' {
  \voiceOne
  \Key
  g1 | g | g
}
lhLower = \relative c {
  \voiceTwo
  \Key
  c1 | c | c
}

PianoRH = {
```

```
  \clef treble
  \global
  \set Staff.midiInstrument = #"acoustic grand"
  <<
    \new Voice = "one" \rhUpper
    \new Voice = "two" \rhLower
  >>
}
PianoLH = {
  \clef bass
  \global
  \set Staff.midiInstrument = #"acoustic grand"
  <<
    \new Voice = "one" \lhUpper
    \new Voice = "two" \lhLower
  >>
}

piano = {
  <<
    \set PianoStaff.instrumentName = #"Piano"
    \new Staff = "upper" \PianoRH
    \new Staff = "lower" \PianoLH
  >>
}

% ------ Bass Guitar ------
Bass = \relative c {
  \Key
  c1 | c | c
}
bass = {
  \global
  \set Staff.instrumentName = #"Bass"
  \clef bass
  <<
    \Bass
  >>
}

% ------ Drums ------
up = \drummode {
  \voiceOne
  hh4 <hh sn> hh <hh sn>
  hh4 <hh sn> hh <hh sn>
  hh4 <hh sn> hh <hh sn>
}
down = \drummode {
  \voiceTwo
  bd4 s bd s
  bd4 s bd s
  bd4 s bd s
```

```
}

drumContents = {
  \global
  <<
    \set DrumStaff.instrumentName = #"Drums"
    \new DrumVoice \up
    \new DrumVoice \down
  >>
}

%%%%%%%% It All Goes Together Here %%%%%%%%%%%%%%%%%%%%%

\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \trumpet
      \new Staff = "altosax" \altoSax
      \new ChordNames = "barichords" \bariHarmony
      \new Staff = "barisax" \bariSax
      \new Staff = "trombone" \trombone
    >>
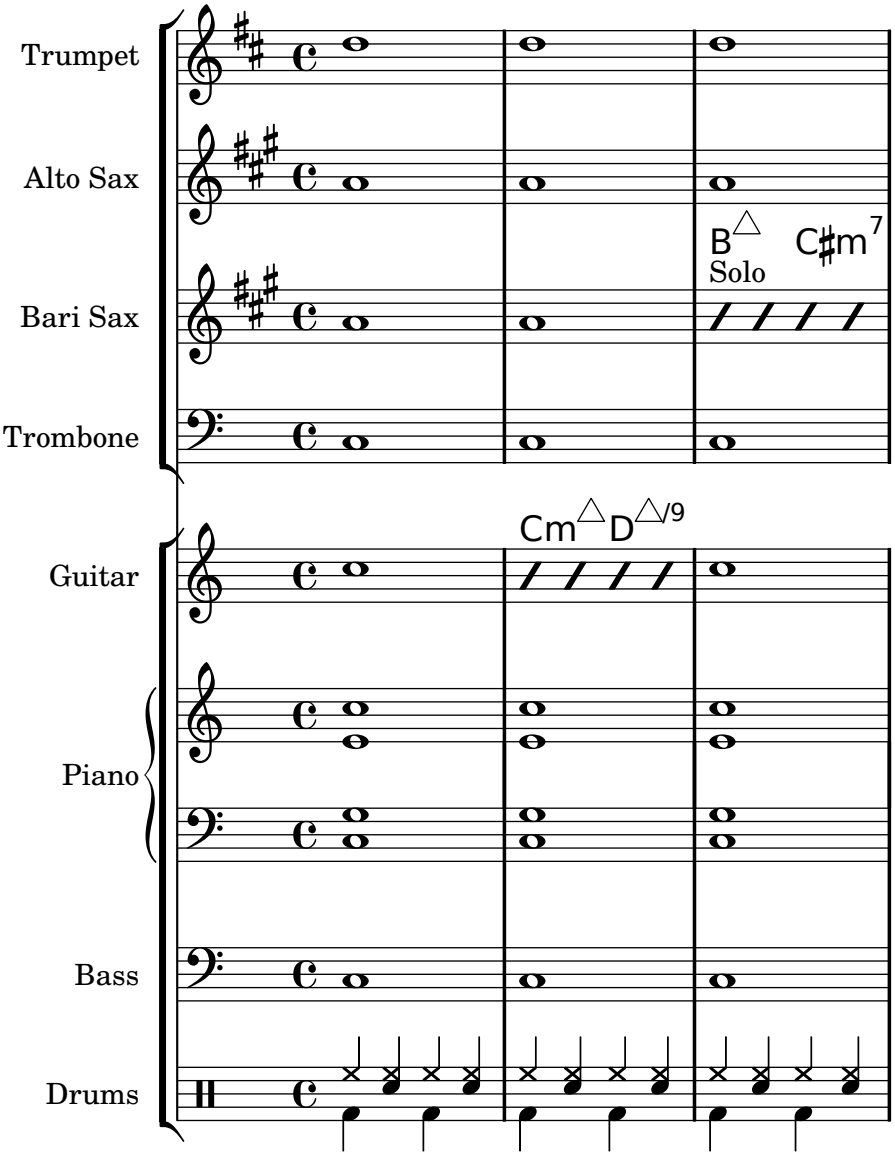
    \new StaffGroup = "rhythm" <<
      \new ChordNames = "chords" \gtrHarmony
      \new Staff = "guitar" \guitar
      \new PianoStaff = "piano" \piano
      \new Staff = "bass" \bass
      \new DrumStaff \drumContents
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
    \context {
      \Score
      \override BarNumber #'padding = #3
      \override RehearsalMark #'padding = #2
      skipBars = ##t
    }
  }
  \midi { }
}
```

Song
(tune)

Me

moderato

Swing


Trumpet
Alto Sax
Bari Sax
Trombone
Guitar
Piano
Bass
Drums
B△
Solo
C♯m7
Cm△
D△/9

Annexe B 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.

Annexe C Index de LilyPond

!

! 25

%

% 18
%{ ... %} 18

’

' 14

(

(...) 23

,

, ... 14

.

... . 18

<

< 25, 31
< ... > 31
<< 28, 31
<< ... >> 28
<< ... \\ ... >> 31
<< \\ >> 48

>

> 25, 31
>> 28, 31

[

[..... 26
[...] 26

]

] 26

^

^ 24

_

_ 24

\

\! 25
\(... \) 23
\< 25
\> 25
\\ 31, 48
\acciaccatura 27
\addlyrics 32
\addlyrics – exemple 93
\addlyrics, exemple 97
\appoggiatura 27
\autoBeamOff 26, 57
\autoBeamOn 26
\book 41, 42
\clef 17
\consists 68
\context 66
\f 25
\ff 25
\grace 27
\header 38, 42
\key 22
\layout 42, 69
\lyricmode 57
\lyricsto 56
\major 22
\markup 25
\mf 25
\midi 42
\minor 22
\mp 25
\new 29, 60
\new ChoirStaff 57
\new Lyrics 56
\new Staff 29
\new Voice 53
\once 90, 95
\oneVoice 53
\override 89
\overrideProperty 91
\p 25
\partial 26
\pp 25
\relative 14
\remove 68
\revert 90, 95
\score 41, 43
\set 63
\set, exemple d’utilisation 112
\shiftOff 56
\shiftOn 56
\shiftOnn 56
\shiftOnnn 56
\startTextSpan 113
\stopTextSpan 113
\tempo 16
\textLengthOff 116
\textLengthOn 116
\time 16
\times 27
\tweak 91
\tweak, exemple 91, 92
\unset 63
\version 18

\voiceFour ... 53
\voiceFourStyle ... 51
\voiceNeutralStyle ... 51
\voiceOne ... 53
\voiceOneStyle ... 51
\voiceThree ... 53
\voiceThreeStyle ... 51
\voiceTwo ... 53
\voiceTwoStyle ... 51
\with ... 66
\with, exemple ... 104, 105, 106, 107

Callback functions

Callback functions ... 145

Difficult tweaks

Difficult tweaks ... 91

Scheme tutorial

Scheme tutorial ... 97, 98, 99, 144

~

~ ... 23

A

absolu, mode ... 39
accents ... 24
acciaccatura ... 27
acciaccature ... 27
Accidental, exemple de dérogation ... 122
AccidentalPlacement, exemple de dérogation ... 122
accolades ... 18
accords et durée ... 31
accords ou voix ... 48
accords, notes simultanées ... 31
addlyrics ... 32
ajout de texte ... 25
alignAboveContext, exemple d'utilisation de la propriété ... 104, 105, 106, 107
alignement d'objets sur une ligne de base ... 122
alignement des paroles ... 32
altérations ... 21
altérations à l'armure ... 22
altérations et mode relatif ... 14
ambitus, graveur ... 68
anacrouse ... 26
apostrophe ... 14
appoggiature ... 27
armure, altérations à l' ... 22
armure, définition de l' ... 22
articulation ... 24
articulation, liaisons d' ... 23
articulations et liaisons ... 115
assignation de variables ... 37
autoBeamOff ... 26, 57
autoBeamOn ... 26

B

bandeaux ... 113
BarLine, exemple de dérogation ... 100, 101, 102, 103
barre de ligature ... 26
Beam, exemple de dérogation ... 125
bémol ... 21
bémol, double ... 21
bien lire le manuel ... 19
blanche ... 15
bloc d'en-tête ... 38
bloc de commentaire ... 18
bloc score, contenu ... 43
book ... 41, 42
book, bloc implicite ... 42
book, livre, ouvrage ... 41
bound-details, exemple d'utilisation de la propriété ... 113, 114
break-visibility exemple d'utilisation de la propriété ... 101
break-visibility, propriété ... 101

C

calques (layers) ... 48
caractère souligné (paroles) ... 32
caractères autorisés dans les variables ... 37
casse, prise en compte de ... 18
casse, prise en compte de la ... 1
center ... 109
chansons ... 32
chevauchement de notation ... 121
chiffre de mesure ... 16
chœur, partie de ... 57
chœur, système pour ... 30
ChoirStaff ... 30, 57
ChordNames ... 29
clavier, portée pour ... 30
clé ... 17
clef ... 17
Clef, exemple de dérogation ... 103, 104, 105, 106, 107
cliquables, exemples ... 19
collision d'objets à l'intérieur d'une portée ... 123
collisions de notes ... 56
color, détermination de la propriété par une procédure Scheme ... 144
color, exemple d'utilisation de la propriété ... 102, 103
color, propriété ... 102
combinaison d'expressions en parallèle ... 28
commentaires ... 18
compilation ... 1
compulser le manuel ... 19
conseils de construction des fichiers ... 19
consists ... 68
construction des fichiers, conseils ... 19
context ... 66
contexte ... 29
contexte de notation ... 29
contexte de voix ... 48
contexte, détermination du ... 95
contexte, identification correcte du ... 95
contexte, nommage ... 61
contexte, propriétés ... 63
contexte, spécification en mode lyrique ... 98
contextes implicites ... 41

contextes, création de ... 60
contextes, les différents ... 59
contrôle des nolets, liaisons, phrasés et ligatures ... 124
conventions de nommage des objets ... 89
conventions de nommage des propriétés ... 89
couleur X11 ... 102
couleur, exemple d'utilisation de la propriété ... 91, 92
couleur,exemple d'utilisation de la propriété ... 90
couleurs rgb ... 103
couplets multiples et musique vocale ... 58
courantes, erreurs ... 19
création d'objet invisible ... 135
crescendo ... 25
crochet de nolet ... 92
crochets, imbrication ... 47
crochets, types de ... 47

D

décalage (padding) ... 119, 121
décalage, commandes ... 56
decrescendo ... 25
défaut, retour aux propriétés par ... 95
dépannage ... 19
déplacement d'objets en collision ... 118
déplacement d'objets se chevauchant ... 118
déplacement de grobs en collision ... 118
dérogation pour une seule fois ... 95
dérogation, exemple ... 93
dièse ... 21
dièse, double ... 21
dimensionnement des grobs ... 117
direction des hampes et voix ... 52
direction, exemple d'utilisation de la propriété ... 92, 109, 110
distance ... 107
doigtés ... 24
doigtés, accords ... 110
doigtés, exemple ... 111, 112
doigtés, exemple de dérogation ... 110
doigtés, positionnement ... 110
double bémol ... 21
double dièse ... 21
down ... 109
durée, liaisons de ... 23
durées ... 15
DynamicLineSpanner, exemple de dérogation ... 122
DynamicText, exemple de dérogation ... 117, 118, 122

E

écartement des lignes, modification ... 107
éditeurs de texte ... 1
empilement de notes ... 56
en-tête ... 42
en-têtes ... 38
engravers ... 62
épaisseur ... 107
Épaisseur, exemple d'utilisation de la propriété ... 95, 96
erreurs courantes ... 19
`es` ... 21
`eses` ... 21
espacement des notes ... 55
espacement des portées ... 81
espaces multiples, insensibilité ... 18
étiquette ... 25
étiquette de texte et collision ... 116
exemple, premier ... 1
exemple, SATB ... 74
exemples cliquables ... 19
expression ... 28
expression musicale ... 27
expression musicale composite ... 43
expressions ... 18
expressions parallèles ... 28
expressions parallèles et hauteur relative ... 28
extenseur ... 88
extension de texte ... 113
extra-offset, exemple d'utilisation de la propriété ... 123
extra-offset, propriété ... 120
`extra-spacing-width` ... 117
extra-spacing-width, exemple d'utilisation de la propriété ... 117, 118, 122
extra-spacing-width, propriété ... 119

F

fa, clef de ... 17
fichier PDF ... 1
fichiers, conseils de construction ... 19
Fingering, exemple de dérogation ... 123
fingeringOrientations, exemple d'utilisation de la propriété ... 112
font-series, exemple d'utilisation de la propriété ... 137
font-shape, exemple d'utilisation de la propriété ... 97, 137
font-size, exemple d'utilisation de la propriété ... 91
fontSize , exemple d'utilisation de la propriété ... 107
fontSize, exemple d'utilisation de la propriété ... 106
fontSize, valeur par défaut et réglage ... 66
force-hshift, exemple d'utilisation de la propriété ... 126, 133
force-hshift, propriété ... 120
format d'entrée ... 41

G

gestion manuelle des nolets, liaisons, phrasés et ligatures ... 124
`grace` ... 27
`GrandStaff` ... 30
graphical objects ... 81
graveurs ... 62
graveurs, ajout ... 68
graveurs, suppression ... 68
grob ... 88
grob, dimensionnement ... 117
grobs ... 81
grobs, évitement des collisions ... 118
grobs, positionnement ... 123
grobs, propriétés ... 93

H

hampe, modification de longueur ... 107
hampes en bas ... 52

hampes en haut ... 52
hauteur relative et expressions parallèles ... 28
hauteur relative et musique simultanée ... 28
hauteurs ... 14
hauteurs, valeurs absolues ... 39
`header` ... 38, 42

I

identificateurs ... 37, 43, 84
imbrication d'expressions musicales ... 54
imbrication de constructions simultanées ... 54
implicite, bloc book ... 42
implicites, contextes ... 41
indication d'octaviation ... 113
indication du tempo ... 16
indication métronomique ... 16
indications métronomiques, modification du positionnement ... 114
insensibilité aux espaces multiples ... 18
interface ... 88, 97
interfaces, propriétés des ... 97
invisible, silence ... 31
invisibles, objets ... 135
`is` ... 21
`isis` ... 21
italic, exemple ... 97

K

`key` ... 22

L

lancer LilyPond sous MacOS X ... 2
lancer LilyPond sous Unix ... 13
lancer LilyPond sous Windows ... 6
`layout` ... 42, 69
layout, effets selon l'emplacement ... 43
legato ... 23
levée ... 26
liaison de tenue avec changement de voix ... 135
Liaison, exemple de dérogation ... 95, 96
liaisons d'articulation ... 23
liaisons d'articulation et de prolongation, différences ... 23
liaisons de phrasé ... 23
liaisons de prolongation ... 23
liaisons de tenue ... 23
liaisons et articulations ... 115
liaisons et constructions simultanées ... 49
liaisons et outside-staff-priority ... 115
liaisons, gestion manuelle ... 124
ligature ... 26
ligatures automatiques ... 26
ligatures de nolet, gestion manuelle ... 124
ligatures et paroles ... 57
ligatures explicites ... 26
ligatures manuelles ... 26
ligatures, gestion manuelle ... 124
ligne d'extension ... 32
ligne de commentaire ... 18
lire la partition ... 1
livre ... 41
longueur ... 107
`lyricmode` ... 57
`Lyrics` ... 29, 56
Lyrics, création d'un contexte ... 56
`lyricsto` ... 56
LyricText, exemple de dérogation ... 97, 137

M

MacOS X, lancer LilyPond ... 2
macros ... 37
magstep ... 107
magstep, exemple d'utilisation de la fonction ... 107
majeur ... 22
`major` ... 22
Manuel des références internes ... 93
manuel, lecture ... 19
`Manuels` ... 1
`markup` ... 25
markup, exemple ... 108
marques de repère, modification du positionnement ... 114
masquage d'objets ... 135
mélisme ... 32
mesure incomplète ... 26
méthodes de retouche ... 89
métrique ... 16
métronome, indication ... 16
MetronomeMark, exemple de dérogation ... 121, 136
`midi` ... 42
mineur ... 22
`minor` ... 22
mise en forme ... 42
mode absolu ... 39
mode lyrique, spécification de contexte en ... 98
mode relatif ... 14
mode relatif et altérations ... 14
mode relatif et polyphonie ... 51
modèles, création ... 79
modèles, modification des ... 71
modification de la taille des objets ... 104
modification des propriétés d'un contexte ... 63
modification du positionnement des indications métronomiques ... 114
modification du positionnement des marques de repère ... 114
modification du positionnement des numéros de mesure ... 114
modifier le positionnement des nuances ... 116
MultiMeasureRest, exemple de dérogation ... 123
multiples portées ... 29
multiples voix sur une portée ... 31
musique concurrente ... 48
musique simultanée ... 48
musique simultanée et hauteur relative ... 28

N

neutral ... 109
`new` ... 29, 60
`new Staff` ... 29
noire ... 15
nolet, crochet ... 92
nolets ... 27

nolets imbriqués ... 92
nommage des contextes ... 61
noms de note absolus ... 39
notation des silences ... 16
notation simple ... 13
notation, contexte ... 29
note column ... 56
note pointée ... 15
note, durée ... 15
NoteColumn, exemple de dérogation ... 126, 133
NoteHead, exemple de dérogation ... 90, 91, 103, 144
notes d'ornement ... 27
notes, nom des ... 39
notes, répartition selon le texte ... 116
nouveaux contextes ... 60
nuances ... 25
nuances, modifier le positionnement ... 116
numéro de version ... 18
numéros de mesure, modification du positionnement ... 114

O

objet ... 88
objet de rendu ... 88
objet, propriétés ... 88
objets de la portée ... 108
objets de rendu, propriétés ... 93
objets extérieurs à la portée ... 108
objets graphiques ... 81
objets, alignement sur une ligne de base ... 122
objets, conventions de nommage des ... 89
objets, évitement des collisions ... 118
objets, invisibles ... 135
objets, masquage d' ... 135
objets, modification de taille ... 104
objets, positionnement ... 123
objets, suppression d' ... 135
objets, taille ... 104
`once` ... 90, 95
`oneVoice` ... 53
ornementation ... 27
ossias ... 46
outside-staff-priority, exemple d'utilisation de la propriété ... 115, 116
`override` ... 89
override, commande ... 89
override, exemple ... 93
override, syntaxe ... 89
`overrideProperty` ... 91
overrideProperty, commande ... 91

P

padding, exemple d'utilisation de la propriété ... 121
padding, propriété ... 119
parallèles, expressions ... 28
paroles ... 32
paroles et ligatures ... 57
paroles et portées mutiples ... 36
paroles, affectation à une voix ... 56
paroles, alignement des ... 32
paroles, mot de plusieurs syllabes ... 32
`partial` ... 26
partition ... 41, 43
partition vocale avec plusieurs couplets ... 58
partition, lire ... 1
partitions multiples ... 42
PDF ... 1
phrasé, liaisons de ... 23
phrasés, gestion manuelle ... 124
PhrasingSlur, exemple de dérogation ... 124
piano, portée pour ... 30
`PianoStaff` ... 30
point d'orgue, rendu en MIDI ... 136
polyphonie ... 28, 31, 48
polyphonie et mode relatif ... 51
portée double ... 30
portée pour piano ... 30
portée, objets de la ... 108
portée, objets extérieurs à la ... 108
portée, positionnement ... 46
portées multiples ... 28, 29
portées multiples et paroles ... 36
portées, espacement ... 81
portées, regroupement de ... 30
portées, temporaires ... 46
positionnement des grobs ... 123
positionnement des objets ... 123
positions, exemple d'utilisation de la propriété ... 124, 125
positions, propriété ... 120
premier exemple ... 1
prise en compte de la casse ... 1, 18
prolongation, liaisons de ... 23
propriété, types de ... 98
propriétés d'un contexte, définition avec \context ... 66
propriétés d'un contexte, définition avec \with ... 66
propriétés d'un contexte, modification ... 63
propriétés des interfaces ... 97
propriétés des objets ... 88
propriétés des objets de rendu ... 93
propriétés des objets graphiques (grobs) ... 93
propriétés et contextes ... 63
propriétés, conventions de nommage des ... 89
propriétéset sous-propriétés ... 81

R

Références internes ... 93
Références internes, exemple d'utilisation ... 93
réglage de propriétés au sein des contextes ... 63
regroupement de portées ... 30
`relative` ... 14
`remove` ... 68
rendu, objets de ... 88
résolution des chevauchements de notation ... 121
rétablissement ... 95
retouches et utilisation de variables ... 137
retouches, méthodologie ... 89
retour ... 95
retour à un contexte Voice unique ... 53
`revert` ... 90, 95
revert, commande ... 90
rgb, couleur ... 103
`rgb-color` ... 103
rhythmes ... 15

right-padding, exemple d'utilisation de la propriété 122
right-padding, propriété 119, 122
ronde 15

S

SATB, squelette 74
SATB, structure 58
`score` 41, 43
`Score` 29
score, partition 41
Script, exemple de dérogation 121
self-alignment-X, exemple d'utilisation de la propriété 123
self-alignment-X, propriété 119
sensibilité à la casse 1, 18
`set` 63
shift, commandes 56
`shiftOff` 56
`shiftOn` 56
`shiftOnn` 56
`shiftOnnn` 56
silence invisible 31
silences 16
simple, notation 13
Slur, exemple de dérogation 94, 95, 96
sol, clef de 17
sous-propriétés 81
spanner 88
spanners 113
spécification des durées 15
squelettes 19
squelettes, création 79
staccato 24
`Staff` 29
staff-padding, exemple d'utilisation de la propriété 122
staff-padding, propriété 119
staff-position, exemple d'utilisation de la propriété 123, 132
staff-position, propriété 120
staff-space, exemple d'utilisation de la propriété .. 107
StaffSymbol, exemple de dérogation 103, 107
`startTextSpan` 113
stem down 52
stem up 52
Stem, exemple de dérogation 103, 109, 133, 135
stencil, exemple d'utilisation de la propriété 100, 102, 104, 107, 122, 136
stencil, propriété 100
stencil, utilisation de la propriété 136
`stopTextSpan` 113
StringNumber, exemple de dérogation 123
structure d'hymne 58
structure d'une partition vocale 57
structure de ficher 41
suppression d'objets 135
système 30
système pour chœur 30

T

taille d'objets 104
taille, modification 107
`tempo` 16
tempo, indication 16
tenue, liaisons de 23
text, exemple d'utilisation de la propriété 122
texte, ajout de 25
texte, exemple d'utilisation de la propriété 92
`textLengthOff` 116
`textLengthOn` 116
TextScript, exemple de dérogation 115, 116
TextSpanner, exemple de dérogation 113, 114
thickness, exemple d'utilisation de la propriété ... 94, 95, 96
Tie, exemple de dérogation 132
`time` 16
`times` 27
TimeSignature, exemple de dérogation 101, 102, 103, 104, 105, 106, 107
titre 38
trait d'union (paroles) 32
transparence 101
transparence, exemple d'utilisation de la propriété 92
transparent, exemple d'utilisation de la propriété 101, 133, 135, 136
transparent, propriété 101
transparent, utilisation de la propriété 135
triolet, crochet 92
triolets 27
triolets imbriqués 92
tuplet-number, exemple de fonction 92
`TupletBracket` 92
TupletNumber, exemple de dérogation 92
`tweak` 91
tweak, commande 91

U

Unix, lancer LilyPond 13
`unset` 63
up 109
usage unique, dérogation à 95
ut, clef d' 17
utilisation de variables 37
utilisation de variables dans les retouches 137

V

variables 37, 43, 84
variables, caractères autorisés dans les 37
variables, définition 37
variables, utilisation dans les retouches 137
variables, utilisation de 37
version 18
versionage 18
virgule 14
`Voice` 29
Voice, contexte 48
Voice, création de contextes 53
Voice, retour à un seul contexte 53
`voiceFour` 53
`voiceOne` 53
`voiceThree` 53
`voiceTwo` 53

voix et constructions simultanées 49
voix et direction des hampes 52
voix multiples 48
voix multiples sur une portée 31
voix ou accords 48
voix temporaires 54
voix, imbrication 54
voix, nommage 49

W

Windows, lancer LilyPond 6
`with` 66

X

X11, couleurs 102
`x11-color` 102
x11-color, exemple d'utilisaation de la fonction .. 144
x11-color, exemple d'utilisation 103