LilyPond

Le système de gravure musicale

Manuel d'initiation

L'équipe de développement de LilyPond

Copyright © 1999–2008 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 2.12.1

Table des matières

Préface **1**

1 Introduction **2**

1.1 Contexte 2
Gravure 2
Gravure automatisée 3
Gravure des symboles musicaux 5
Représentation de la musique 6
Exemples d'application 8
1.2 À propos de la documentation 9
À propos du manuel d'initiation 9
À propos du glossaire musicologique 9
À propos du manuel de notation 9
À propos du manuel d'utilisation 10
À propos des morceaux choisis 10
À propos des références du programme 10
Autres sources de documentation 11

2 Tutoriel **12**

2.1 Premiers pas 12
2.1.1 Compilation d'un fichier 12
2.1.2 Notation simple 13
2.1.3 Travail sur les fichiers d'entrée 18
2.1.4 Bien lire le manuel 19
2.2 Notation sur une seule portée 19
2.2.1 Altérations et armure 19
2.2.2 Liaisons 21
2.2.3 Articulations et nuances 22
2.2.4 Ajout de texte 24
2.2.5 Barres de ligature automatiques et manuelles 24
2.2.6 Commandes rythmiques avancées 25
2.3 Notes simultanées 26
2.3.1 Les expressions musicales en clair 26
2.3.2 Plusieurs portées 28
2.3.3 Regroupements de portées 29
2.3.4 Combinaison de notes en accords 30
2.3.5 Polyphonie sur une portée 30
2.4 Chansons 31
2.4.1 Écriture de chants simples 31
2.4.2 Alignement des paroles sur une mélodie 32
2.4.3 Paroles pour plusieurs portées 35
2.5 Dernières précisions 36
2.5.1 Organisation du code source avec des variables 36
2.5.2 Numéro de version 38
2.5.3 Ajout de titres 38
2.5.4 Noms de note absolus 38
2.5.5 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 63
3.3.1 Tout savoir sur les contextes 63
3.3.2 Création d'un contexte 64
3.3.3 Tout savoir sur les graveurs 66
3.3.4 Modification des propriétés d'un contexte 67
Définition des propriétés de contexte avec `\with` 70
Définition des propriétés de contexte avec `\context` 70
3.3.5 Ajout et suppression de graveurs 71
3.4 Extension des modèles 74
3.4.1 Soprano et violoncelle 74
3.4.2 Partition pour chœur à quatre voix mixtes 77
3.4.3 Écriture d'une partition à partir de zéro 81

4 Retouche de partition 85

4.1 Retouches élémentaires 85
4.1.1 Introduction aux retouches 85
4.1.2 Objets et interfaces 85
4.1.3 Conventions de nommage des objets et propriétés 86
4.1.4 Méthodes de retouche 86
4.2 Le manuel des références internes 90
4.2.1 Propriétés des objets de rendu 90
4.2.2 Propriétés listées par interface 94
4.2.3 Types de propriétés 95
4.3 Apparence des objets 96
4.3.1 Visibilité et couleur des objets 96
4.3.2 Taille des objets 100
4.3.3 Longueur et épaisseur des objets 103
4.4 Positionnement des objets 104
4.4.1 Comportement automatique 105
4.4.2 Objets inclus dans la portée 106
4.4.3 Objets hors de la portée 109
4.5 Collisions d'objets 114
4.5.1 Déplacement d'objets 115
4.5.2 Correction des collisions d'objets 117
4.5.3 Exemple concret 122
4.6 Autres retouches 130
4.6.1 Autres utilisations des retouches 130
4.6.2 Utilisation de variables dans les retouches 132
4.6.3 Autres sources de documentation 133
4.6.4 Options ralentissant le traitement 135
4.6.5 Retouches avancées avec Scheme 135

5 Travail sur des projets LilyPond 137

5.1 Suggestions pour la saisie de fichiers LilyPond 137
5.1.1 Suggestions générales 137
5.1.2 Gravure de musique existante 138
5.1.3 Projets d'envergure 138
5.1.4 Économie de saisie grâce aux identificateurs et fonctions 138
5.1.5 Feuilles de style 140
5.2 Quand ça ne fonctionne pas 144
5.2.1 Mise à jour d'anciens fichiers 144
5.2.2 Résolution de problèmes – tout remettre à plat 144
5.2.3 Exemples minimalistes 145
5.3 Conducteurs et parties 145

Annexe A Modèles 147

A.1 Portée unique 147
A.1.1 Notes seules 147
A.1.2 Notes et paroles 147
A.1.3 Notes et accords 147
A.1.4 Notes, paroles et accords 147
A.2 Modèles pour claviers 147
A.2.1 Piano seul 147
A.2.2 Chant et accompagnement 147
A.2.3 Piano et paroles entre les portées 147
A.2.4 Piano et nuances entre les portées 147
A.3 Quatuor à cordes 147
A.3.1 Quatuor à cordes 147
A.3.2 Parties pour quatuor à cordes 147
A.4 Ensemble vocal 147
A.4.1 Partition pour chœur à quatre voix mixtes 147
A.4.2 Partition pour chœur SATB avec réduction pour piano 147
A.4.3 Partition pour chœur SATB avec alignement des contextes 147
A.5 Exemples de notation ancienne 147
A.5.1 Transcription de musique mensurale 147
A.5.2 Transcription du grégorien 147
A.6 Symboles de jazz 147
A.7 Squelettes pour lilypond-book 147
A.7.1 LaTeX 147
A.7.2 Texinfo 148
A.7.3 xelatex 148

Annexe B Tutoriel Scheme 149

B.1 Scheme et les retouches 149

Annexe C Licence GNU de documentation libre 150

Annexe D Index de LilyPond 156

Préface

Ce doit être pendant une répétition de l'Orchestre des Jeunes d'Eindhoven, un jour de 1995, que Jan, du pupitre des altistes tordus, aborda Han-Wen, un corniste déjanté, pour lui parler d'un mirifique projet dans lequel il venait de se lancer. Il s'agissait d'un système automatisé de gravure musicale — le préprocesseur MPP pour MusiXTeX pour être précis. Han-Wen, qui voulait justement imprimer certaines parties d'une oeuvre, jeta un premier coup d'oeil à ce programme, et devint très vite accro. Le MPP s'avéra bientôt une voie sans issue. Après avoir ratiociné et échangé un grand nombre de courriels enflammés, Han-Wen lança le projet LilyPond en 1996, dans lequel, cette fois, ce fut au tour de Jan de se sentir entraîné.

De bien des points de vue, coder un programme informatique, c'est comme apprendre à jouer d'un instrument. Au début c'est sympa, on découvre comment ça fonctionne, et on aborde tout ce qu'on n'arrive pas encore à faire comme autant de défis. L'exaltation de la nouveauté s'estompant, on doit s'entraîner encore et encore. Les gammes, les études deviennent vite ennuyeuses, et peuvent, si l'on n'est pas encouragé par d'autres — professeurs, chefs ou public — en décourager plus d'un. Pourtant, pour peu que l'on persévère, l'instrument devient progressivement une partie de notre vie. Si certains jours il semble naturel d'en jouer, c'est un vrai bonheur. Et si d'autres jours on ne peut tout simplement rien en tirer, on continue quand même à travailler, coûte que coûte.

De même, développer LilyPond peut être une tâche harassante. Certains jours, c'est un monceau de bugs duquel il faut se dépêtrer. Pourtant, il fait maintenant partie de notre vie, et nous nous accrochons. Notre principale motivation vient sans doute de l'utilité réelle de notre logiciel. En flânant sur Internet, nous trouvons beaucoup de gens qui utilisent LilyPond pour réaliser d'impressionnantes partitions : c'est incroyable, mais en même temps très flatteur.

Les utilisateurs ne se contentent pas de nous encourager en utilisant notre logiciel ; nombre d'entre eux nous aident aussi en faisant des suggestions et en signalant des bogues. Aussi, nous voudrions remercier ici tous les utilisateurs qui nous ont signalé des bugs, ont fait des suggestions ou ont contribué d'une façon ou d'une autre au développement de LilyPond.

Jouer de la musique ou en graver, il y a là plus qu'une comparaison séduisante. Même si l'on s'amuse beaucoup en programmant tous ensemble, et qu'on éprouve une satisfaction profonde à aider les gens, au bout du compte, notre travail sur LilyPond est avant tout une manière d'exprimer notre amour sincère de la musique. Puisse-t-il vous aider à créer de nombreuses et belles oeuvres !

Han-Wen et Jan

Utrecht/Eindhoven, Pays-Bas, juillet 2002.

1 Introduction

Ce chapitre constitue une première présentation de LilyPond et de sa documentation.

1.1 Contexte

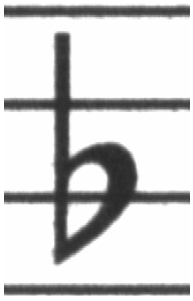
Cette partie présente les objectifs de LilyPond ainsi que son architecture.

Gravure

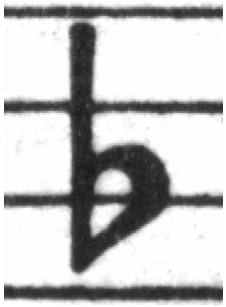
L'art de la typographie musicale se nomme la *gravure*. Ce terme est issu du processus traditionnel d'impression musicale. Il y a seulement quelques dizaines d'années, on faisait les partitions en coupant et en embossant une plaque de zinc ou d'étain en image miroir. Cette plaque était ensuite encrée, les dépressions créées par les creux et les bosses retenant l'encre. Une image était formée en pressant du papier sur la plaque. La découpe et l'embossage étaient entièrement faits à la main. Il était pénible d'appliquer une correction, quand celle-ci n'était pas impossible ; la gravure devait donc être parfaite du premier coup. La gravure demandait une qualification hautement spécialisée : un artisan devait accomplir environ cinq ans de formation avant de mériter le titre de maître graveur, et il lui fallait cinq années d'expérience supplémentaires pour devenir vraiment habile.

De nos jours, toutes les partitions récentes sont produites avec des ordinateurs. Ceci a des avantages évidents : le coût des impressions a diminué, et le travail d'édition peut être envoyé par courriel. Malheureusement, l'utilisation dominante des ordinateurs a également diminué la qualité graphique des partitions. L'impression informatisée leur donne un aspect fade et mécanique qui les rend désagréables à jouer.

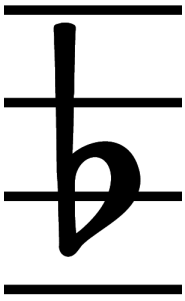
Les images ci-dessous illustrent la différence entre la gravure traditionelle et l'impression typique par ordinateur, et la troisième image montre comment LilyPond mime l'aspect traditionnel. L'image de gauche est une numérisation d'un symbole bémol d'une édition publiée en 2000. Celle du centre montre un bémol d'une gravure à la main de l'édition Bärenreiter de la même musique. L'image de gauche illustre des défauts typiques de l'impression informatique : les lignes de portée sont minces, l'épaisseur de trait du bémol est la même que les lignes fines, et il y a un aspect rigide avec des angles pointus. Par contraste, le bémol Bärenreiter possède un aspect gras et arrondi, presque voluptueux. Notre symbole bémol est créé, entre autres, à partir de celui-là. Il est arrondi, et son épaisseur de trait s'harmonise avec nos lignes de portée, lesquelles sont également plus épaisses que celles de l'édition informatique.



Henle (2000)
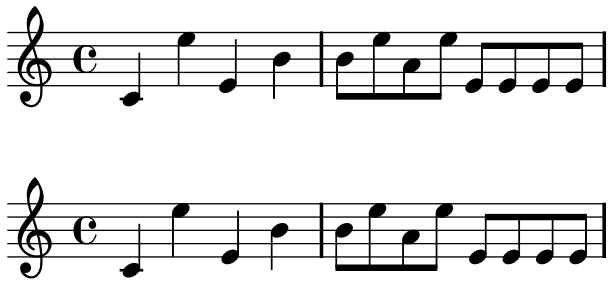


Bärenreiter (1950)



Fonte Feta de LilyPond (2003)

En matière d'espacement, la répartition de l'espace devrait refléter les durées entre les notes. Cependant, beaucoup de partitions modernes se contentent des durées avec une précision mathématique, ce qui mène à de mauvais résultats. Dans l'exemple suivant, un motif est imprimé deux fois : une fois en utilisant un espacement mathématique exact, et une autre fois avec des corrections. Pouvez-vous les repérer ?



L'extrait n'utilise que des notes de même durée ; l'espacement devrait le refléter. Malheureusement, notre oeil nous trompe quelque peu ; il ne se contente pas de remarquer la distance entre les têtes de notes, il prend en compte également la distance entre les hampes consécutives. Ainsi, par compensation, les notes avec une combinaison « hampe vers le haut »/« hampe vers le bas » doivent être éloignées l'une de l'autre, et les notes avec une combinaison « hampe vers le bas »/ « hampe vers le haut » rapprochées, le tout dépendant de la position verticale des notes. Les deux premières mesures sont imprimées avec cette correction, les deux suivantes sans. Les notes dans les deux dernières mesures forment des blocs de notes « hampe vers le bas »/« hampe vers le haut ».

Les musiciens sont généralement plus absorbés par l'exécution que par l'étude de l'aspect graphique d'une partition, donc discutailler sur les détails typographiques peut paraître peu important. Il n'en est rien. Dans de longues pièces avec des rythmes monotones, les corrections d'espacement engendrent de subtiles variations dans la mise en forme de chaque ligne, donnant à chacune une signature visuelle distincte. Sans cette signature, toutes les lignes auraient le même aspect, et ressembleraient à un labyrinthe. Si un musicien regarde ailleurs un instant ou se déconcentre momentanement, il peut avoir du mal à se retrouver sur la page.

De même, l'aspect robuste des symboles sur d'épaisses lignes de portée ressort mieux quand la partition est éloignée du lecteur, comme sur un pupitre par exemple. Une organisation minutieuse des espaces vides permet de minimiser l'espace qu'occupe la musique, tout en évitant que les symboles s'amassent les uns contre les autres. Le résultat permet de réduire le nombre de pages à tourner, ce qui est un grand avantage.

Ceci est une caractéristique commune à toute typographie. La disposition doit être belle, non seulement pour des raisons esthétiques, mais également pour l'aide apportée au lecteur dans la tâche qu'il doit accomplir. Pour du matériel d'exécution comme les partitions de musique, cela prend une double importance : les musiciens ont une quantité limitée d'attention. Moins ils en ont besoin pour lire, plus ils peuvent se concentrer sur la musique elle-même. Autrement dit, une meilleure typographie permet une meilleure interprétation.

Ces exemples démontrent que la typographie musicale est un art subtil et complexe, et que la produire demande une expertise considérable, que les musiciens ne possèdent généralement pas. LilyPond est le fruit de nos efforts pour restaurer l'excellence graphique de la gravure à la main à l'ère de l'ordinateur, et la rendre accessible à tous les musiciens. Nous avons conçu nos algorithmes, fontes et paramètres de programme pour retrouver la qualité d'édition des anciennes partitions que nous aimons tant lire et jouer.

Gravure automatisée

Comment pouvons-nous implémenter la typographie ? Si les artisans ont besoin de plus de dix ans pour devenir de vrais maîtres, comment nous, simples programmeurs, pourrions-nous jamais écrire un programme pour faire leur travail ?

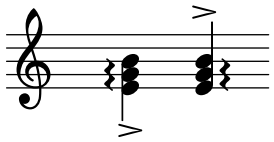
La réponse est : nous ne le pouvons pas. La typographie se base sur le jugement visuel humain, donc les humains ne peuvent pas être complètement remplacés. Si LilyPond arrive à traiter la plupart des situations correctement, ce sera déjà une grande avancée sur les logiciels existants. Les problèmes restants peuvent être résolus à la main. Au fil des ans, le logiciel peut être affiné pour faire de plus en plus de choses automatiquement, pour que les ajustements manuels soient de moins en moins nécessaires.

Quand nous avons commencé, nous avons écrit le programme Lilypond entièrement dans le language de programmation C++ ; les fonctions du programme étaient figées par les développeurs. Ceci s'est avéré insatisfaisant pour plusieurs raisons :
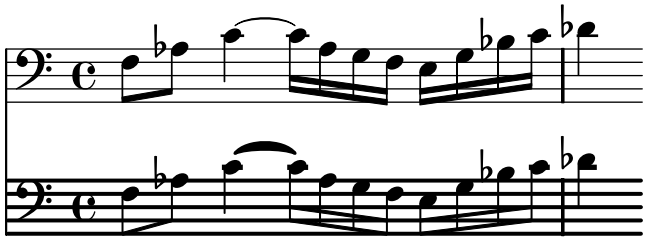
- Quand Lilypond fait des erreurs, les utilisateurs ont besoin de contredire les décisions de formatage. Les utilisateurs doivent donc avoir accès au moteur de formatage. Par conséquent, les règles et les propriétés ne peuvent pas être fixées par nous au moment de la compilation, mais doivent être accessibles aux utilisateurs au moment de l'exécution.
- La gravure est une question de jugement visuel, et donc de goût. Aussi bien informés que nous le sommes, les utilisateurs peuvent être en désaccord avec nos décisions personnelles. Par conséquent, les définitions du modèle typographique doivent également être accessibles à l'utilisateur.
- Enfin, nous affinons continuellement les algorithmes de formatage, donc nous avons besoin d'une approche souple des règles. Le langage C++ oblige à une certaine méthode de groupage des règles qui ne convient pas bien au fonctionnement de la notation musicale.

Ces problèmes ont été résolus en intégrant un interpréteur pour le langage de programmation Scheme, et en réécrivant des parties de LilyPond en Scheme. L'architecture actuelle de formatage est construite autour de la notion d'objets graphiques, décrits par des fonctions et des variables Scheme. Cette architecture comprend les règles de formatage, le style typographique, et des décisions individuelles de formatage. L'utilisateur a un accès direct à la plupart de ces contrôles.

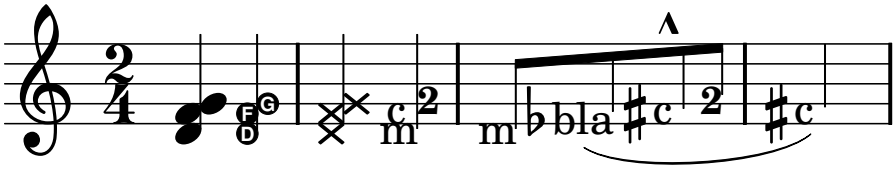
Les variables Scheme contrôlent les décisions de mise en page. Par exemple, beaucoup d'objets graphiques ont une variable de direction qui encode le choix entre haut et bas (ou gauche et droite). Vous pouvez voir ici deux accords, avec des accents, et des arpèges. Dans le premier accord, les objets graphiques sont tous dirigés vers le bas (ou la gauche). Dans le second accord ils sont tous dirigés vers le haut (droite).



Le processus de formatage d'une partition consiste à lire et écrire les variables d'objets graphiques. Certaines variables ont une valeur prédéfinie. Par exemple, l'épaisseur d'un grand nombre de lignes – une caractéristique du style typographique – est une variable avec une valeur prédéfinie. Vous êtes libres d'altérer cette valeur, ce qui vous donne une partition avec une impression typographique différente.



Les règles de formatage ont aussi des variables prédéfinies : chaque objet possède des variables contenant des procédures. Ces procédures exécutent le formatage, et en les substituant par d'autres, nous pouvons changer l'apparence des objets. Dans l'exemple suivant, la règle du choix de têtes de notes est changée au cours de l'extrait de musique.



Gravure des symboles musicaux

Le processus de formatage décide où placer les symboles. Cependant, cela ne peut être fait qu'à partir du moment où il a été décidé *quels* symboles doivent être imprimés, c'est-à-dire quelle notation utiliser.
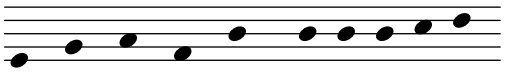
La notation musicale usuelle est un système d'écriture qui a évolué à travers les dix derniers siècles. La forme qui est aujourd'hui communément utilisée date du début de la Renaissance. Bien que la forme basique — les têtes de notes sur une portée de cinq lignes — n'ait pas changé, les détails continuent d'évoluer pour exprimer les innovations de la notation contemporaine. Par conséquent, elle comprend quelque 500 ans de musique, avec des applications allant des mélodies monodiques à de monstrueux contrepoints pour grand orchestre.

Comment pouvons nous appréhender un tel monstre à plusieurs têtes, et le confiner dans l'espace réduit d'un programme informatique ? Notre solution consiste à diviser le problème de la notation — par opposition à la gravure, ou typographie — en morceaux digestes et programmables : chaque type de symbole est géré par un module séparé, couramment appelé greffon[1]. Chaque greffon est entièrement modulaire et indépendant, et donc peut être développé et amélioré séparément. De tels greffons sont nommés `graveurs`[2], par analogie avec les artisans qui traduisent les idées musicales en symboles graphiques.

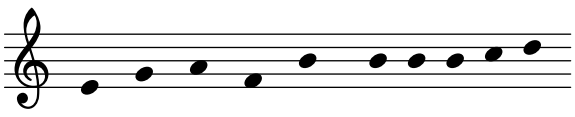
Dans l'exemple suivant, voyons comment nous commençons avec un greffon pour les têtes de notes, le graveur de têtes de note (`Note_heads_engraver`) :
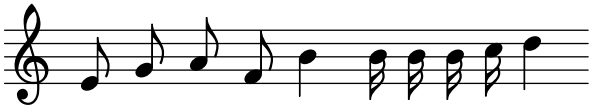


Ensuite, le graveur du symbole de portée (`Staff_symbol_engraver`) ajoute la portée



le graveur de clef (`Clef_engraver`) définit un point de référence pour la portée



et le graveur de hampes (`Stem_engraver`) ajoute les hampes :
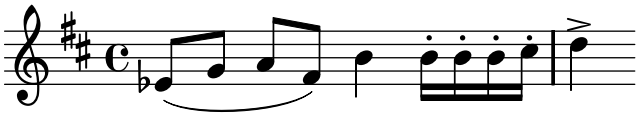


Le graveur de hampe est informé de chaque tête de note qui survient. Chaque fois qu'une tête de note — plusieurs pour un accord — est rencontrée, un objet hampe est créé et connecté à la
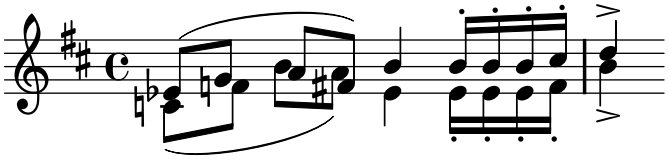
[1] traduction de l'anglais *plug-in*.

[2] `engraver`s en anglais.

tête de note. En ajoutant des graveurs pour les barres de ligature, les liaisons, les accents, les altérations accidentelles, les barres de mesure, la métrique, et les armures, nous obtenons un jeu de notation complet.



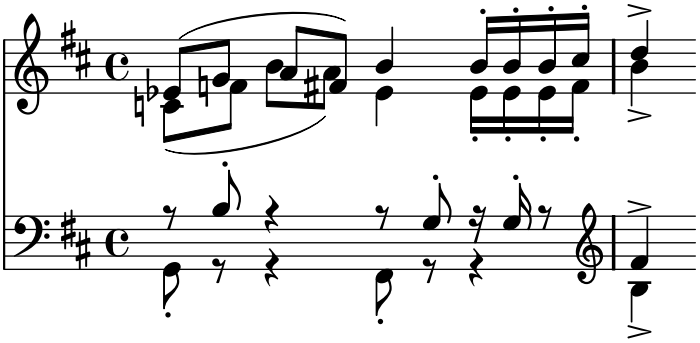
Ce système fonctionne bien pour de la musique monodique, mais qu'en est-il de la polyphonie ? En notation polyphonique, plusieurs voix peuvent partager une portée.



Dans cette situation, la portée et les altérations accidentelles sont partagées, mais les hampes, liaisons etc., sont spécifiques à chaque voix. Par conséquent, les graveurs doivent être groupés. Les graveurs des têtes de notes, hampes, liaisons etc., vont dans un groupe appelé « contexte de Voix »[3], alors que les graveurs des clés, altérations accidentelles, barres de mesure etc., vont dans un groupe appelé « contexte de Portée ». Dans le cas de la polyphonie, un seul contexte de Portée contient plusieurs contextes de Voix. De même, plusieurs contextes de Portée peuvent être inclus dans un seul contexte de Partition. Le contexte de Partition est le contexte de notation de plus haut niveau.

Voir aussi

Référence du programme: Section "Contexts" dans *Référence des propriétés internes*.



Représentation de la musique
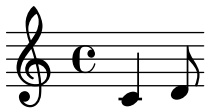
Idéalement, le format d'entrée pour n'importe quel système de formatage est une description abstraite du contenu. Dans ce cas-ci, ce serait la musique elle-même. Cela pose un formidable problème : comment pouvons-nous définir ce qu'est réellement la musique ? Plutôt que d'essayer de trouver une réponse, nous avons renversé la question. Nous écrivons un logiciel capable de produire de la musique écrite, et adaptons le format pour atteindre la plus grande concision possible. Quand le format ne peut plus être simplifié, il nous reste par définition le contenu lui-même. Notre logiciel sert de définition formelle d'un document de musique.

Pour LilyPond, la syntaxe est également l'interface utilisateur ; par conséquent il est facile de saisir

[3] 'Voice context' en anglais, 'Voice' commence par une majuscule comme tous les noms de contexte dans le programme LilyPond.

```
{
c'4 d'8
}
```

c'est-à-dire un do central noire et, juste au-dessus, un ré croche



Sur une échelle microscopique, une telle syntaxe est facile à utiliser. À plus grande échelle, la syntaxe a besoin aussi de structure. Comment serait-il possible autrement de rentrer des pièces complexes comme des symphonies ou des opéras ? La structure est formée par le concept d'expression musicale : en combinant de petits fragments de musique pour en former de plus grands, on peut exprimer de la musique plus complexe. Par exemple

```
f4
```
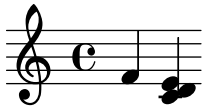


Des accord peuvent être construits avec << et >> autour des notes.
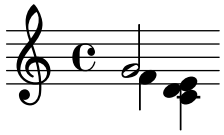
```
<<c4 d4 e4>>
```



On met cette expression dans une séquence en l'encadrant par des accolades `{ ... }`

```
{ f4 <<c4 d4 e4>> }
```



Ceci est également une expression, et peut donc encore une fois être combiné avec d'autres expressions simultanées (une blanche) en utilisant <<, \\, et >>

```
<< g2 \\ { f4 <<c4 d4 e4>> } >>
```
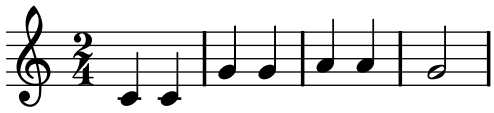


De telles structures récursives peuvent être spécifiées formellement et de manière ordonnée dans une grammaire indépendante de tout contexte. Le code d'analyse est aussi généré à partir de cette grammaire. Autrement dit, la syntaxe de LilyPond est définie clairement et sans ambiguité.

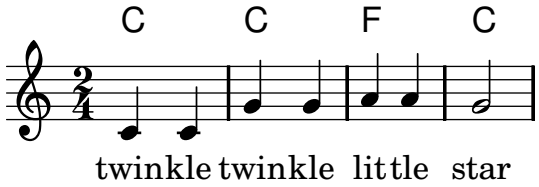
L'interface utilisateur et la syntaxe sont ce que les gens voient et manipulent le plus. Elles sont en partie une affaire de goût, et aussi sujettes à beaucoup de discussions. Même si ces discussions sur les goûts ont leur mérite, elles ne sont pas très productives. D'un point de vue plus large sur LilyPond, l'importance de la syntaxe est minime : il est facile d'inventer une syntaxe concise, alors qu'écrire un code de formatage décent est beaucoup plus difficile. Ceci est également illustré par le nombre de lignes de codes pour les composants respectifs : l'analyse et la représentation constituent moins de 10% du code source.

Exemples d'application

Nous avons conçu LilyPond comme une expérimentation visant à concentrer l'art de la gravure musicale dans un logiciel. Grâce à tout ce dur labeur, le programme peut maintenant être utilisé pour accomplir des travaux utiles. L'application la plus simple est d'imprimer des notes :



En ajoutant des noms d'accords et des paroles, nous obtenons une partition de chanson :

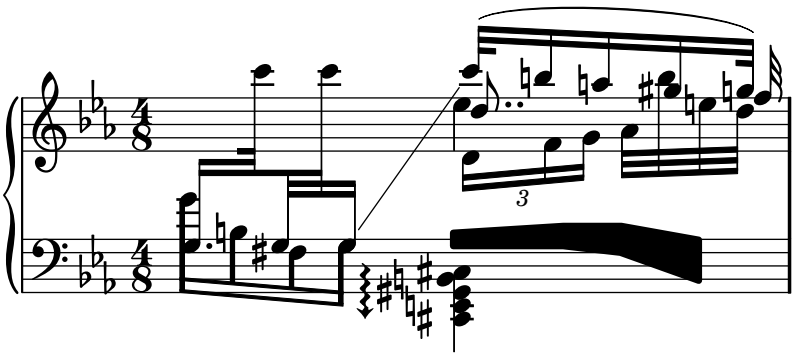


La notation polyphonique et la musique pour piano peuvent également être générées. L'exemple suivant associe quelques constructions plus exotiques :

Screech and boink

Random complex notation

Han-Wen Nienhuys



Les extraits exposés ici ont tous été écrits à la main, mais ce n'est pas une obligation. Puisque le moteur de formatage est en grande partie automatique, il peut servir de sortie pour d'autres programmes qui manipulent la musique. Par exemple, il peut être utilisé pour convertir des bases de données d'extraits musicaux en images pour des sites Internet et des présentations multimédias.

Ce manuel montre également une application : le format d'entrée est du texte, et peut donc facilement être intégré dans d'autres formats basés sur le texte comme LaTeX, HTML, ou dans le cas de ce manuel, Texinfo. À l'aide d'un programme spécial, les extraits de code peuvent être remplacés par des images de musiques dans les fichiers de sortie PDF ou HTML. Cela donne la possibilité de mélanger de la musique et du texte dans les documents.

1.2 À propos de la documentation

Cette partie présente les différents volumes de la documentation.

À propos du manuel d'initiation

Ce manuel explique comment débuter avec LilyPond, et expose de manière simple quelques concepts clés. Il est conseillé de lire ces chapitres de manière linéaire.

Dans 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.

- Chapitre 1 [Introduction], page 2 : le pourquoi du comment de LilyPond.
- Chapitre 2 [Tutoriel], page 12 : introduction en douceur à la typographie musicale. Les utilisateurs débutants sont invités à commencer par ce chapitre.
- Chapitre 3 [Concepts fondamentaux], page 41 : concepts généraux du format de fichier `ly` spécifique à LilyPond. Si vous n'êtes pas certain de l'endroit où placer une commande, lisez ce chapitre !
- Chapitre 4 [Retouche de partition], page 85 : introduction aux retouches de gravure avec LilyPond.
- Chapitre 5 [Travail sur des projets LilyPond], page 137 : utilisation pratique de LilyPond, conseils généraux, prévention et résolution des problèmes les plus courants. À lire avant de se lancer dans des travaux d'envergure !

Ce volume contient aussi des annexes que vous pouvez consulter au gré de vos besoins :

- Annexe A [Modèles], page 147 de pièces LilyPond. Copiez et collez un modèle dans un fichier, ajoutez les notes, et c'est prêt !
- Annexe B [Tutoriel Scheme], page 149 : courte introduction à Scheme, le langage de programmation utilisé dans les fonctions de musique. Ces quelques lignes vous aideront à construire des retouches avancées ; nombre d'utilisateurs ne touchent jamais à Scheme.

À propos du glossaire musicologique

Section "Glossaire musical" dans *Glossaire* : ce document explique en anglais des termes musicaux, et donne leur traduction dans diverses langues. Si vous n'êtes pas familier avec la notation et la terminologie musicales, il est conseillé de consulter le glossaire, notamment pour les parties non encore traduites de la documentation.

À propos du manuel de notation

Ce manuel détaille toutes les commandes LilyPond produisant une notation musicale. La lecture de cet ouvrage requiert une bonne compréhension des concepts exposés dans le manuel d'initiation.

- Section "Notation musicale générale" dans *Manuel de notation* : cette partie décrit la notation de base, qui sera utile dans la plupart des projets de partition. Les sujets sont groupés par type de notation.
- Section "Notation spécialisée" dans *Manuel de notation* : cette partie détaille des éléments de notation spécifiques à certains instruments ou styles. Les sujets sont groupés par type de notation.
- Section "Généralités en matière d'entrée et sortie" dans *Manuel de notation* : informations générales sur les fichiers source LilyPond et le contrôle des sorties.
- Section "Gestion de l'espace" dans *Manuel de notation* : différents aspects de l'espacement selon les axes et échelles, par exemple la sélection de la taille de papier, ou la gestion des sauts de page.

- Section “Modification des réglages prédéfinis” dans *Manuel de notation* : ce chapitre est une référence des différentes formes de retouches, qui permettent d'obtenir de Lilypond (presque) tout ce que vous désirez.
- Section “Interfaces pour les programmeurs” dans *Manuel de notation* : création de fonctions de musique à l'aide de Scheme.

Les annexes de ce manuel contiennent entre autres des tableaux de référence pratiques.

- Section “Bibliographie” dans *Manuel de notation* : choix de livres de référence, pour en savoir plus sur la notation et la gravure.
- Section “Tables du manuel de notation” dans *Manuel de notation* : tableaux montrant les noms d'accord, les instruments MIDI, les noms de couleur, et la police Feta.
- Section “Aide-mémoire” dans *Manuel de notation* : référence pratique des commandes LilyPond les plus courantes.
- Section “Index des commandes LilyPond” dans *Manuel de notation* : index de toutes les `\commandes` LilyPond.
- Section “Index de LilyPond” dans *Manuel de notation* : un index complet.

À propos du manuel d'utilisation

Ce manuel explique l'exécution des programmes et l'intégration de partitions LilyPond dans d'autres programmes.

- Section “Installation” dans *Manuel d'utilisation du programme* : installation — et éventuellement compilation — de LilyPond.
- Section “Environnement de travail” dans *Manuel d'utilisation du programme* : configuration de votre système pour une utilisation optimale de LilyPond, comprenant l'utilisation d'environnements adaptés pour certains éditeurs de tecte.
- Section “Exécution de LilyPond” dans *Manuel d'utilisation du programme* : exécution de LilyPond et de ses programmes auxiliaires. De plus, cette partie explique comment effectuer la mise à jour de fichiers source écrits avec d'anciennes versions de LilyPond.
- Section “LilyPond-book” dans *Manuel d'utilisation du programme* : création de documents intégrant des extraits musicaux, comme ce manuel.
- Section “Conversion à partir d'autres formats” dans *Manuel d'utilisation du programme* : utilisation des programmes de conversion. Ces programmes sont livrés avec le paquetage LilyPond, et convertissent divers formats de musique vers le format `.ly`.

À propos des morceaux choisis

Section “Exemples de code” dans *Exemples de code* : il s'agit d'une sélection de petits exemples montrant des trucs, astuces et fonctionnalités particulières de LilyPond, issus de LilyPond Snippet Repository (LSR). Tous ces exemples sont dans le domaine public.

Notez bien que cette annexe n'est en aucune manière un miroir ou même une partie du LSR. Dans la mesure où le LSR repose sur une version stable de LilyPond, les exemples illustrant des fonctionnalités introduites dans la dernière version de développement ne peuvent y figurer ; c'est pourquoi vous les trouverez dans le répertoire ‘`input/new/`’ des sources de LilyPond.

La liste des exemples correspondant à chacun des sous-chapitres du manuel de notation est accessible par des liens dans le paragraphe **Voir aussi**.

À propos des références du programme

Section “Top” dans *Référence des propriétés internes* : c'est un ensemble de pages HTML étroitement liées entre elles, qui documente les moindres petits détails de chaque classe, objet et fonction de LilyPond. Cette documentation est produite directement à partir des définitions de formatage utilisées.

Presque toutes les fonctions de formatage utilisées en interne sont directement disponibles pour l'utilisateur. Par exemple, toutes les variables qui contrôlent les épaisseurs, les distances etc., peuvent être modifiées dans les fichiers d'entrée. Il y a un grand nombre d'options de formatage, et elles sont toutes décrites dans ce document. Chaque section du manuel de notation a un paragraphe **Voir aussi**, qui renvoie à la documentation générée automatiquement. Dans la documentation au format HTML, ces paragraphes disposent de liens cliquables.

Autres sources de documentation

Pour finir, présentons d'autres précieuses sources de documentation.

- Nouveautés : ce document résume les changements importants et les nouvelles fonctionalités de LilyPond depuis la dernière version stable.
- Les archives de la liste lilypond-user : c'est un dépôt archivant les courriels qui ont été envoyés à la liste anglophone des utilisateurs. Beaucoup de questions sont apparues plusieurs fois sur la liste, il y a donc des chances que si vous avez une question, la réponse puisse être dans ces archives. Les archives de la liste francophone ne sont pas aussi bien fournies, mais vous pouvez toujours y chercher des conversations passées sur les traductions, et si vous avez de la chance une réponse à une question.
- Les archives de la liste lilypond-devel : les courriels envoyés à la liste des développeurs y sont archivés. Les sujets de discussion sont plus techniques ; si vous voulez vous renseigner sur l'histoire du développement ou si vous avez une question très technique, tentez votre chance en cherchant dans ces archives.
- Fragments de musique au cours du texte : dans tous les documents HTML qui incluent des fragments musicaux, le code LilyPond utilisé pour produire l'image est accessible par un clic sur l'image.
- L'emplacement des fichiers de documentation mentionnés ici peut varier d'un système à l'autre. De temps en temps, ce manuel fait référence aux fichiers d'exemple et d'initialisation. Tout au long de ce manuel, nous donnons les emplacements des fichiers d'entrée relativement au répertoire racine de l'archive source. Par exemple, '`input/test/bla.ly`' peut référer au fichier '`lilypond2.x.y/input/test/bla.ly`'. Dans les paquets binaires pour les plate-formes Unix, la documentation et les exemples se trouvent généralement sous '`/usr/share/doc/lilypond/`'. Les fichiers d'initialisation, par exemple '`scm/lily.scm`', ou '`ly/engraver-init.ly`', se trouvent généralement dans le répertoire '`/usr/share/lilypond/`'.

2 Tutoriel

Ce tutoriel commence par une introduction au langage musical utilisé par LilyPond, qui vous permettra de faire fonctionner le logiciel pour produire une partition. Après ce premier contact, nous verrons comment créer des partitions utilisant une notation musicale courante.

2.1 Premiers pas

Cette section présente les aspects élémentaires de l'utilisation de LilyPond.
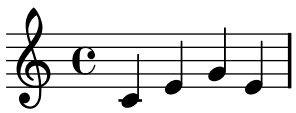
2.1.1 Compilation d'un fichier

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.

```
{
  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 2.1.4 [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.

Saisie de la musique et visualisation de la partition produite

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

Notez qu'il existe plusieurs éditeurs de texte disponibles avec un bon support de LilyPond ; consultez Section "LilyPond et les éditeurs de texte" dans *Manuel d'utilisation du programme*.

> **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.

MacOS X

Si vous double-cliquez sur `LilyPond.app`, un fichier d'exemple s'ouvrira. Sauvegardez-le, par exemple, sous '`test.ly`' sur votre bureau, puis traitez-le avec la commande de menu '`Compile > Typeset File`'. Le fichier PDF résultant sera alors affiché à l'écran.

À l'avenir, vous aurez certainement recours aux commandes « Nouveau » ou « Ouvrir ». Vous devez enregistrer votre fichier avant de lancer la gravure de la partition par LilyPond. Si une erreur apparaît pendant le traitement, vous la trouverez dans la fenêtre « log ».

Windows

Sous Windows, double-cliquez sur l'icône LilyPond qui se trouve sur le bureau, un fichier d'exemple s'ouvre dans un simple éditeur de texte. Enregistrez-le, par exemple en tant que '`test.ly`' sur le bureau, puis double-cliquez sur son icône (qui montre une note de musique) pour le traiter. Après quelques secondes, vous obtiendrez un fichier '`test.pdf`' sur le bureau, fichier que vous pourrez ouvrir pour voir la partition gravée. Une autre méthode pour lancer le traitement du fichier '`test.ly`' est de le glisser avec votre souris sur l'icône de LilyPond.

Pour modifier un fichier '`.ly`' existant, faites un clic droit dessus et sélectionnez « Éditer la source ». Pour partir d'un fichier vide, lancez l'éditeur en ouvrant un fichier existant et utilisez la commande « New » du menu « File ».

En double-cliquant sur le fichier, vous obtiendrez, en plus du fichier PDF, un fichier '`.log`' qui récapitule les opérations que LilyPond a effectuées sur votre fichier. Si une erreur survient, vous en trouverez les détails dans ce fichier.

UNIX

Créez un fichier texte '`test.ly`' qui contient

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

Pour traiter '`test.ly`', entrez la commande suivante dans un terminal :

```
lilypond test.ly
```

Vous verrez quelque chose ressemblant à

```
lilypond test.ly
GNU LilyPond 2.12.1
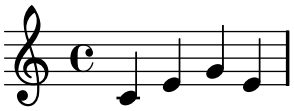
Traitement de « test.ly »
Analyse...
Interprétation en cours de la musique...
Pré-traitement des éléments graphiques...
Détermination du nombre optimal de pages...
Répartition de la musique sur une page...
Dessin des systèmes...
Sortie mise en page vers « test.ps »...
Conversion à « ./test.pdf »...
```

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

2.1.2 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'
}
```
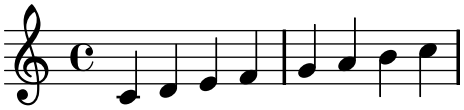


Ces valeurs automatiques simplifient la saisie du code source dans bien des cas ; nous verrons plus loin comment les indiquer explicitement.

Hauteurs

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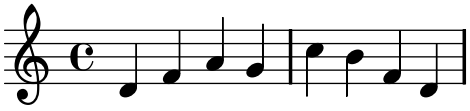
Le moyen le plus simple d’entrer des notes est d’utiliser le mode d’octaves relatives, ou mode `\relative`. Dans ce mode, l’octave de chaque note est sélectionnée automatiquement de façon à ce qu’elle soit la plus proche possible de la note précédente, c’est-à-dire de façon à ce que l’intervalle avec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

```
% définit le point de départ en référence au do central
\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` :
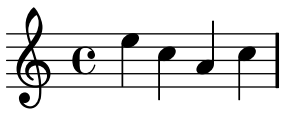
```
\relative c' {
  d f a g
  c b f d
}
```



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.
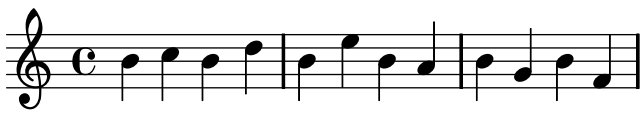
Dans l’exemple suivant, on remplace `c'` dans la commande `\relative c'` par `c''`, afin de calculer l’octave de la première note par rapport au do situé une octave au-dessus du do central :

```
% une octave au dessus du do central
\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.
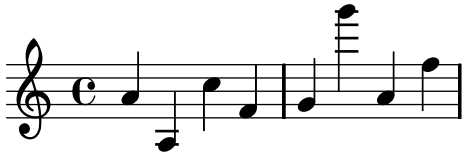
```
\relative c'' {
  b c  % do est à un cran de plus, il sera donc au-dessus
  b d  % ré est à 2 crans de plus ou 5 de moins, il sera donc au-dessus
  b e  % mi est à 3 crans de plus ou 4 de moins, il sera donc au-dessus
  b a  % la est à 6 crans de plus ou 1 de moins, il sera donc en dessous
  b g  % sol est à 5 crans de plus ou 2 de moins, il sera donc en dessous
  b f  % fa est à 4 crans de plus ou 3 de moins, il sera donc en dessous
}
```



Notez que le calcul des octaves relatives **ne dépend pas des altérations** des notes, dièses bémols 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'`.

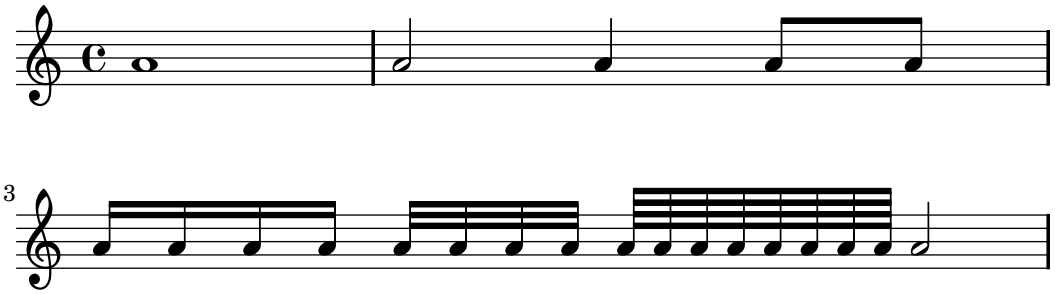
Durées et rythme

Glossaire musical : Section “barre de 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.
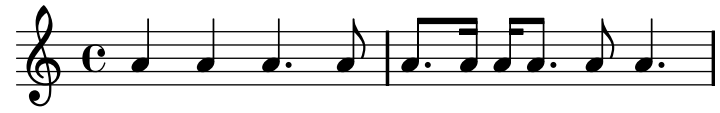
Si aucune durée n’est indiquée pour une note, la dernière durée entrée est utilisée. En l’absence d’indication de durée, la première note est une noire.

```
\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.
}
```

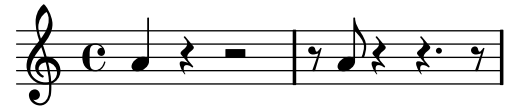


Silences

Glossaire musical : 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
}
```
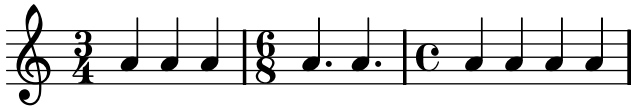


Métrique

Glossaire musical : 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
}
```
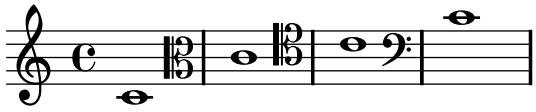


Clefs

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

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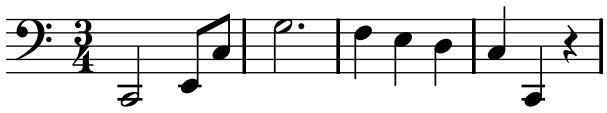
```
\relative c' {
  \clef treble
  c1
  \clef alto
  c1
  \clef tenor
  c1
  \clef bass
  c1
}
```



Tout ensemble

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

```
\relative c, {
  \time 3/4
  \clef bass
  c2 e8 c' g'2.
  f4 e d c4 c, r4
}
```



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 “Métrique” dans *Manuel de notation*, Section “Clefs” dans *Manuel de notation*.

2.1.3 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 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, ou de retours à la ligne. `{ c d e }` a le même sens que `{ c d e }` ou que

```
{
c                    d
 e }
```

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

```
{
  c d e
}
```

- **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 { }` 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 par l’ordinateur, 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
%}
```

2.1.4 Bien lire le manuel

Comme nous l'avons vu dans Section 2.1.3 [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.
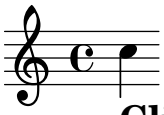
Pour reproduire les exemples, vous pouvez copier et coller le code affiché, mais **à condition** d'ajouter `\relative c'' { }` de la façon suivante :

```
\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'un 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.

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 :



Cliquez ici.

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 ».

Voir aussi

Vous trouverez plus de conseils pour construire des fichiers source dans Section 5.1 [Suggestions pour la saisie de fichiers LilyPond], page 137. Cependant, lors d'une première lecture il est préférable de terminer d'abord la lecture du tutoriel.

2.2 Notation sur une seule portée

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

2.2.1 Altérations et armure

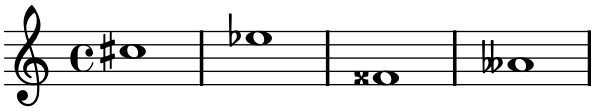
Altérations

Glossaire musical : 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 `\include "italiano.ly"` pour entrer les noms de notes 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 “Noms de note dans d’autres langues” dans *Manuel de notation*.

```
cis1 ees fisis, aeses
```
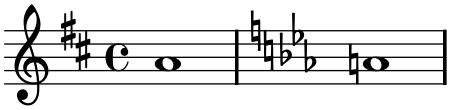


Armures

Glossaire musical : 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 musical : 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*.
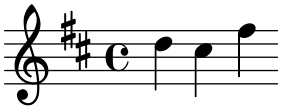
La combinaison de l’*armure* et des hauteurs de note — y compris les altérations — permet à LilyPond de déterminer dans quel cas imprimer des *altérations accidentelles*. L’armure n’affecte que les altérations *imprimées*, et non les hauteurs réelles ! Cette fonctionnalité est souvent source de confusion 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,
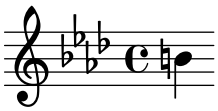
```
\key d \major
d cis fis
```



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
b
```



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 "Noms de note 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*.

Glossaire musical : Section "Noms de note" dans *Glossaire*.
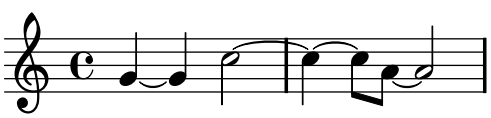
2.2.2 Liaisons

Liaisons de prolongation

Glossaire musical : Section "liaison de prolongation" dans *Glossaire*.

Pour créer une liaison de prolongation[1], on ajoute un tilde `~` à la première note liée.

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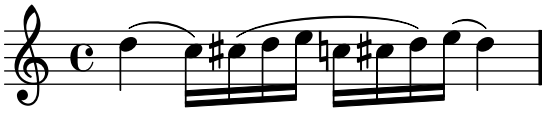


[1] parfois aussi appelée liaison de tenue

Liaisons d'articulation

Glossaire musical : Section "liaison d'articulation" 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 '`)`'.
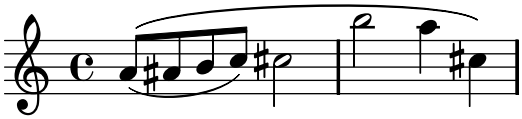
```
d4( c16) cis( d e c cis d) e( d4)
```



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.

```
a8(\( ais b c) cis2 b'2 a4 cis,\)
```
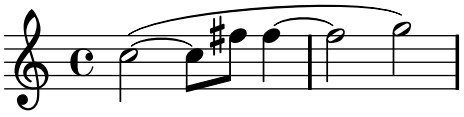


Attention aux types de liaison

Glossaire musical : Section "articulation" dans *Glossaire*, Section "liaison d'articulation" dans *Glossaire*, Section "liaison de prolongation" 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.

```
c2~( c8 fis fis4 ~ fis2 g2)
```



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*.
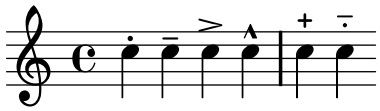
2.2.3 Articulations et nuances

Articulations

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

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

```
c-. c-- c-> c-^ c-+ c-_
```
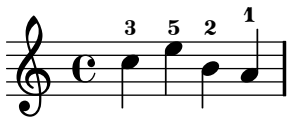


Doigtés

Glossaire musical : Section “doigté” dans *Glossaire*.
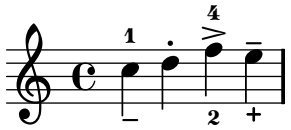
De même, des indications de doigté peuvent être ajoutées à une note en utilisant un tiret (‘-’) et le chiffre à écrire :

```
c-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.

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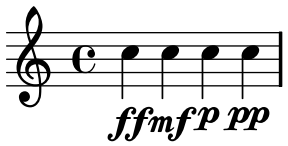


Nuances

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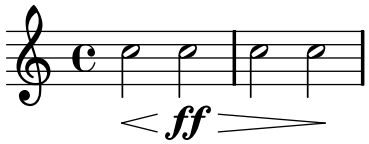
On obtient un signe de *nuance* en ajoutant à la note les lettres du signe, précédées d’un anti-slash ‘\’ :

```
c\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 `\!` :

```
c2\< c2\ff\> c2 c2\!
```



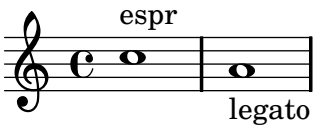
Voir aussi

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

2.2.4 Ajout de texte

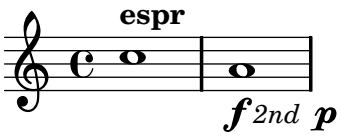
On peut ajouter du texte à une partition :

```
c1^"espr" a_"legato"
```



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

```
c1^\markup{ \bold espr}
a1_\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
}
```



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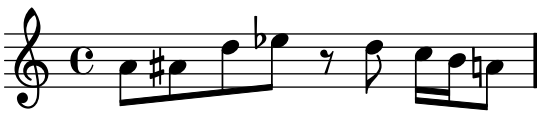
Manuel de notation : Section “Ajout de texte” dans *Manuel de notation*.

2.2.5 Barres de ligature automatiques et manuelles

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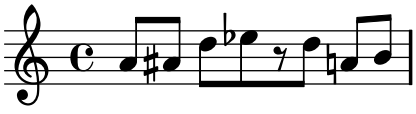
Toutes les barres de ligature sont dessinées automatiquement :

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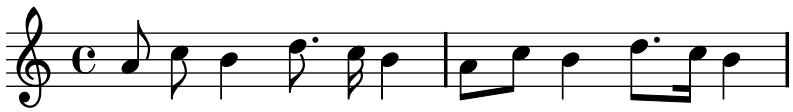


Lorsqu'on n'aime pas la manière dont les notes sont automatiquement groupées, il est possible de les ligaturer manuellement, en marquant la première note à attacher d'un crochet ouvrant `[` et la dernière d'un crochet fermant `]`.

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



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*.
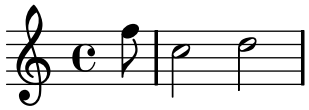
2.2.6 Commandes rythmiques avancées

Mesure incomplète

Glossaire musical : 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
```

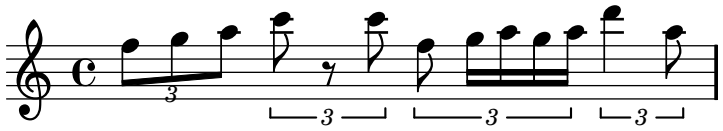


Nolets

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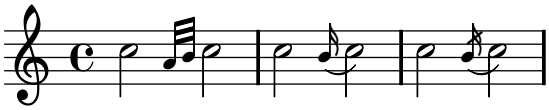


Notes d’ornement

Glossaire musical : Section “notes d’ornement” dans *Glossaire*, Section “acciaccature” dans *Glossaire*, Section “appoggiature” 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*.

2.3 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.

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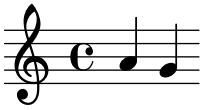
Dans les fichiers source LilyPond, la musique est représentée par ce qu’on appelle des *expressions musicales*. En soi, une seule note peut constituer une expression musicale :

```
a4
```
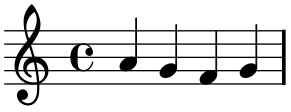


Mettre un groupe de notes entre accolades crée une nouvelle expression musicale, appelée *expression musicale composée*. En voici un exemple avec deux notes :

```
{ 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)
```

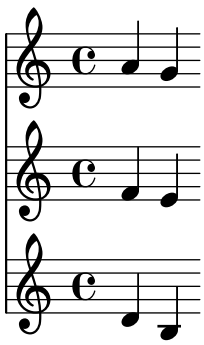
Ceci est une suite d'expressions, où chacune est contenue dans la suivante. Les expressions les plus simples sont les nombres, et de plus grandes expressions sont produites en combinant des expressions avec des opérateurs — comme '`+`', '`*`' et '`/`' — et des parenthèses. Tout comme les expressions 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 musical : 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'' {
  <<
    { a4 g }
    { f e }
    { d b }
  >>
}
```
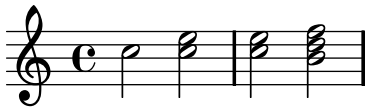


Notez que nous avons ici indenté chaque niveau du fichier d'entrée avec un nombre d'espaces différent. LilyPond se moque — ou presque — de l'espace qu'il peut y avoir ou non au début d'une ligne, mais un code bien indenté est bien plus lisible par des humains.

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.

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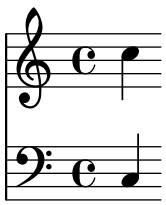


2.3.2 Plusieurs portées

Comme nous l'avons vu dans Section 2.3.1 [Les expressions musicales en clair], page 26, 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 c }
    \new Staff { \clef bass c,, }
  >>
}
```



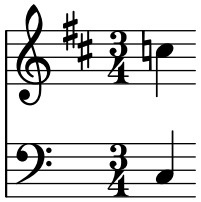
La commande `\new` introduit un « contexte de notation ». Un contexte de notation est un environnement dans lequel les événements musicaux — comme les notes ou les commandes `\clef` — sont interprétés. Pour des pièces simples, ces contextes sont créés automatiquement. Pour des pièces plus complexes, 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'accords.

En termes 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 $-(4 + 5)$ est une plus grande expression.

Les chiffres de métrique indiqués sur une portée affectent toutes les autres portées[2]. 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 c }
    \new Staff { \clef bass c,, }
  >>
}
```



2.3.3 Regroupements de portées
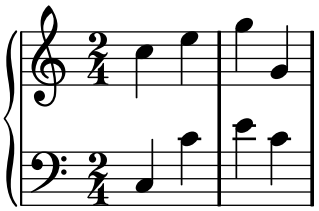
Glossaire musical : Section "accolade" dans *Glossaire*.

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 2.3.2 [Plusieurs portées], page 28, 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,, c' e c }
  >>
}
```



Voir aussi

Manuel de notation : Section "Instruments utilisant des portées multiples" dans *Manuel de notation*, Section "Gravure des portées" dans *Manuel de notation*.
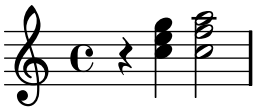
[2] Ce comportement peut être modifié si nécessaire, voir Section "Notation polymétrique" dans *Manuel de notation*

2.3.4 Combinaison de notes en accords

Glossaire musical : Section “accord” dans *Glossaire*.
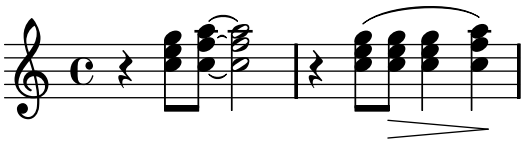
Nous avons vu précédemment comment combiner des notes simultanément, en les encadrant par des angles doubles `<<` et `>>`. Pour produire des accords simples, c'est-à-dire une superposition de notes de même durée, on encadre les hauteurs de notes par des angles simples `<` et `>`, et on écrit la durée juste après.

```
r4 <c e g>4 <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 angles.

```
r4 <c e g>8[ <c f a>]~ <c f a>2
r4 <c e g>8( <c e g>\> <c e g>4 <c f a>\!)
```
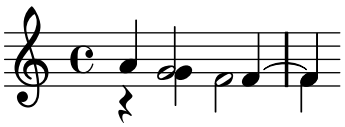


2.3.5 Polyphonie sur une portée

Quand différentes lignes mélodiques sont combinées sur une seule et même portée, elles sont imprimées comme des voix polyphoniques ; chaque voix a ses propre hampes[3], liaisons et ligatures, la voix supérieure ayant les hampes vers le haut, la voix inférieure vers le bas.

On réalise ce type de partition en entrant chaque voix comme une séquence, autrement dit avec `{...}`, puis en combinant simultanément les voix et en les séparant par `\\`.
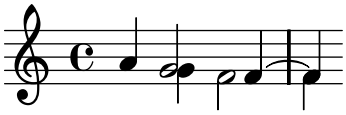
```
<<
  { a4 g2 f4~ f4 } \\
  { r4 g4 f2 f4 }
>>
```



Pour l'écriture de musique polyphonique, les silences invisibles s'avèrent bien pratiques : ce sont des silences qui ne s'impriment pas. Ils sont utiles pour remplir des voix qui, temporairement, ne jouent rien. Voici le même exemple que ci-dessus, avec un silence invisible `s` (pour *skip*) à la place d'un silence normal `r` :

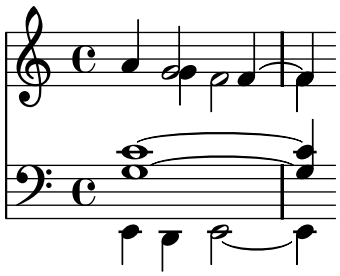
```
<<
  { a4 g2 f4~ f4 } \\
  { s4 g4 f2 f4 }
>>
```

[3] familièrement appelées queues de note.



Là encore, ces expressions peuvent s'imbriquer arbitrairement :

```
<<
  \new Staff <<
    { a4 g2 f4~ f4 } \\
    { s4 g4 f2 f4 }
  >>
  \new Staff <<
    \clef bass
    { <c g>1 ~ <c g>4 } \\
    { e,,4 d e2 ~ e4}
  >>
>>
```



Voir aussi

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

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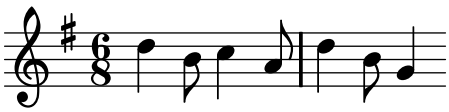
Cette section présente l'écriture vocale et les partitions de variété.

2.4.1 Écriture de chants simples

Glossaire musical : Section "paroles" dans *Glossaire*.

Prenons une mélodie toute simple, la comptine *Girls and boys come out to play.*
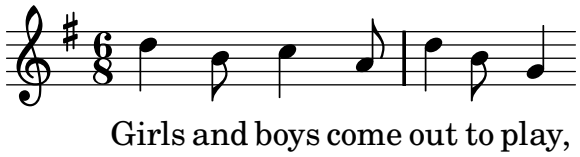
```
\relative c'' {
  \key g \major
  \time 6/8
  d4 b8 c4 a8 d4 b8 g4
}
```



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

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

```
  }
  \addlyrics {
    Girls and boys come out to play,
  }
>>
```



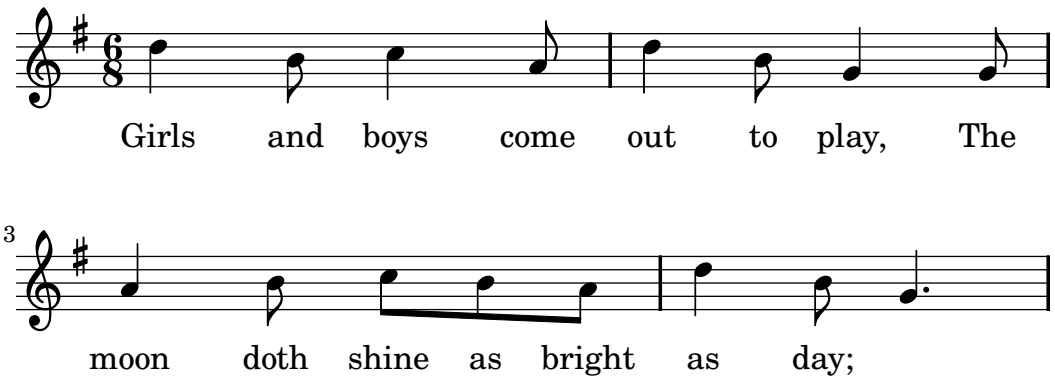
Remarquez les accolades embrassant la musique et celles embrassant les paroles, ainsi que les angles doubles encadrant toute la pièce ; ces derniers indiquent simplement que la musique et les paroles se produisent en même temps.

2.4.2 Alignement des paroles sur une mélodie

Glossaire musical : Section “mélisme” dans *Glossaire*, Section “ligne de prolongation” dans *Glossaire*.

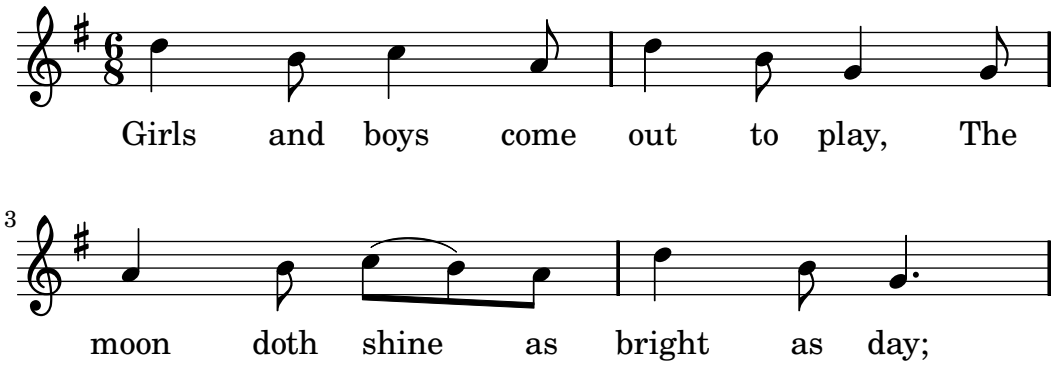
La ligne suivante de la comptine précédente est *The moon doth shine as bright as day.* Ajoutons-la au code.

```
<<
  \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;
  }
>>
```



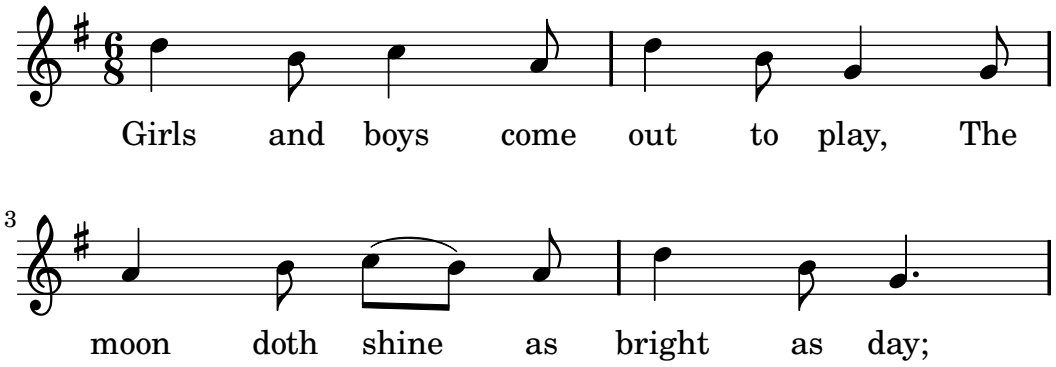
Remarquez que les paroles ajoutées ne s’alignent pas bien avec les notes. Le mot *shine* devrait être chanté sur deux notes au lieu d’une. On appelle ceci un *mélisme* : il s’agit d’une seule syllabe chantée sur plus d’une note. Il existe plusieurs façons d’étaler une sylabe sur plusieurs notes, la plus simple étant de lier les notes du mélisme. Pour les détails, consultez Section 2.2.2 [Liaisons], page 21.

```
<<
  \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;
  }
>>
```



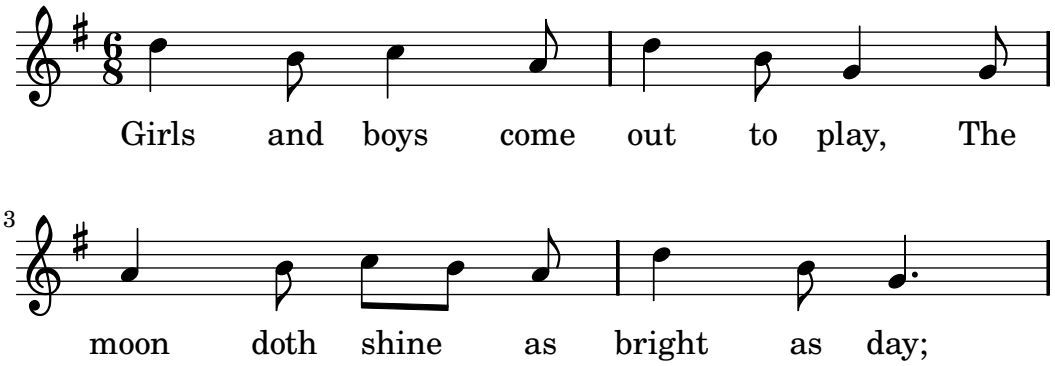
Les paroles sont maintenant correctement alignées, mais les liens de croche automatiques ne conviennent pas pour les notes au-dessus de *shine as*. On peut les corriger en ajoutant des liens de croche manuels, pour ceci consultez Section 2.2.5 [Barres de ligature automatiques et manuelles], page 24.

```
<<
  \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;
  }
>>
```
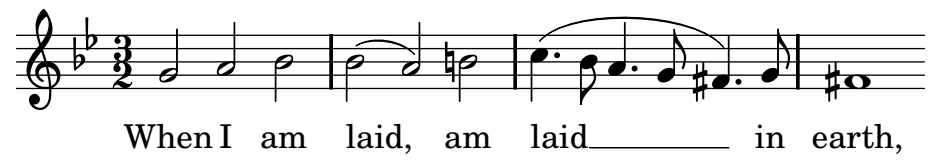


Au lieu d'utiliser une liaison, on peut indiquer le mélisme dans les paroles en insérant un caractère souligné _ pour chaque note du mélisme sauf la première.

```
<<
  \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;
  }
>>
```



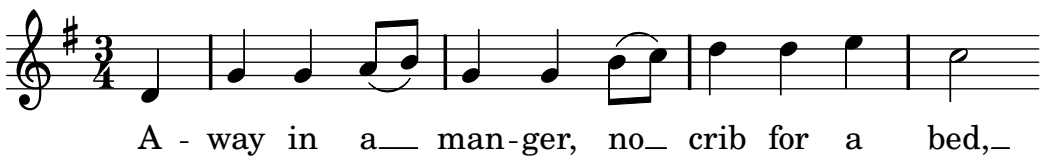
Si une syllabe s'étend sur un grand nombre de notes ou une note très longue, on représente souvent le mélisme par un *trait de prolongation*, qu'on entre avec __. L'exemple suivant montre les trois premières mesures de la plainte de Didon, extraite de *Didon et Énée* de Purcell.

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



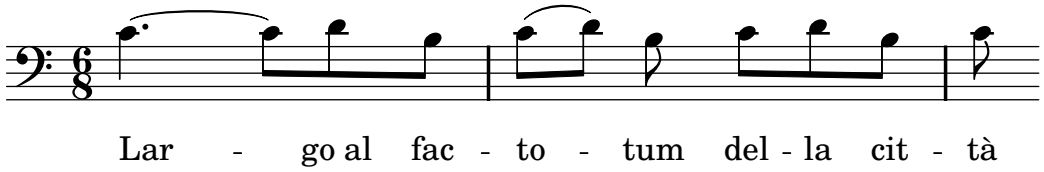
Aucun exemple jusqu'à présent n'a utilisé de mots de plus d'une syllabe. Dans des paroles, de tels mots sont écrits en syllabes séparées par des traits d'union. Avec LilyPond, on utilise deux tirets pour produire un trait d'union centré entre deux syllabes. L'exemple suivant montre tout ce que nous avons 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 g4
    b8( c) d4 d e4 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 c([ d]) b c d b c
  }
  \addlyrics {
    Lar -- go_al fac -- to -- tum del -- la cit -- tà
  }
>>
```



Voir aussi
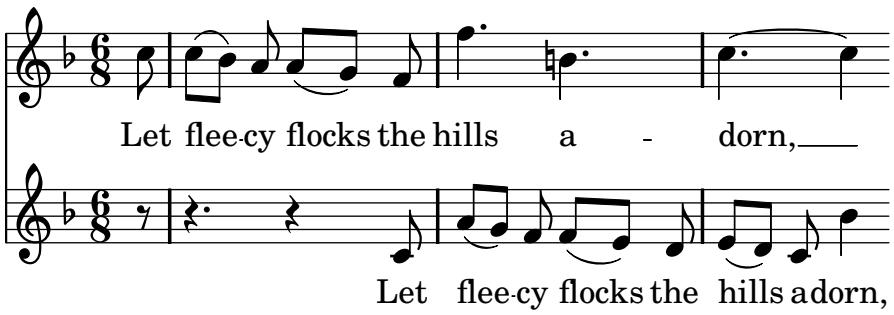
Manuel de notation : Section "Musique vocale" dans *Manuel de notation*.

2.4.3 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 c([ 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'([ g]) f f([ e]) d e([ 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 2.5.1 [Organisation du code source avec des variables], page 36.

Voir aussi

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

2.5 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.

2.5.1 Organisation du code source avec des variables
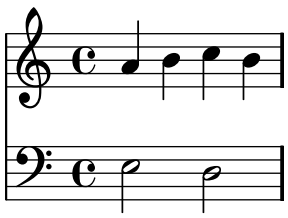
Lorsque l’on combine tous les éléments étudiés précédemment pour écrire des partitions plus longues, les expressions musicales prennent de l’ampleur et, dans le cas des pièces polyphoniques, deviennent profondément imbriquées, jusqu’au point où il devient difficile de se repérer dans le fichier source. 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.

```
violon = \new Staff {
  \relative c'' {
    a4 b c b
  }
}
cello = \new Staff {
  \relative c {
    \clef bass
    e2 d
  }
}
{
  <<
    \violon
    \cello
  >>
}
```
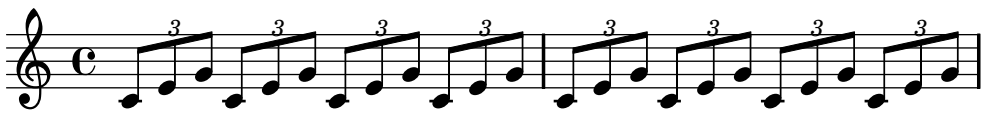


Le nom d'une variable ne doit comporter que des caractères alphabétiques non accentués, aucun nombre ni tiret ne sont autorisés.

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.

```
trioletA = \times 2/3 { c,8 e g }
mesureA = { \trioletA \trioletA \trioletA \trioletA }

\relative c'' {
 \mesureA \mesureA
}
```



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

```
width = 4.5\cm
name = "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 variable définies ci-dessus.

```
\paper {
  \aFivePaper
  line-width = \width
}
{ c4^\name }
```

2.5.2 Numéro de version

La déclaration `\version` stipule le numéro de la version de LilyPond pour laquelle le fichier a été écrit :

```
\version "2.12.1"
```

Par convention, on place cette instruction en début de fichier.

Cette instruction permet de faciliter les mises à jour futures de LilyPond. Les changements de syntaxe au fil des versions sont gérés avec un programme dédié, `convert-ly`, qui utilise la valeur de `\version` pour déterminer les règles de conversion à appliquer au fichier source. Pour plus d'informations, consultez Section "Mise à jour des fichiers avec convert-ly" dans *Manuel d'utilisation du programme*.

2.5.3 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 "2.12.1"
\header {
  title = "Symphonie"
  composer = "Moi"
  opus = "Op. 9"
}

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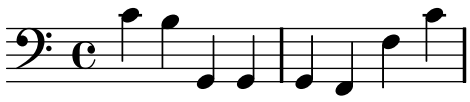
Quand LilyPond traite le fichier, le titre et le compositeur sont imprimés au début de la partition. Vous trouverez plus d'informations sur les titres à la section Section "Création de titres" dans *Manuel de notation*.

2.5.4 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.
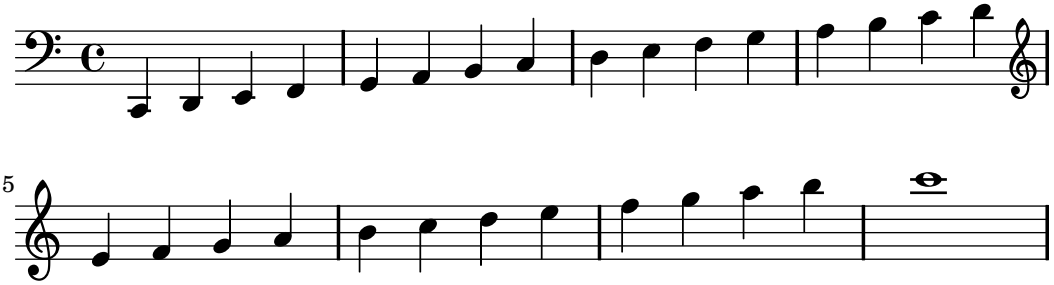
Si vous omettez la commande `\relative`, LilyPond considérera toutes les hauteurs comme des hauteurs absolues. Un `c'` désigne toujours le do central, un `b` se situe une seconde en dessous du do central, et un `g,` est situé sur la première ligne de la portée en clé de fa.

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



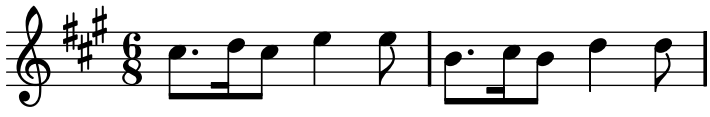
Voici une gamme sur 4 octaves :

```
{
  \clef bass
  c, d, e, f,
  g, a, b, c
  d e f g
  a b c' d'
  \clef treble
  e' f' g' a'
  b' c'' d'' e''
  f'' g'' a'' b''
  c'''1
}
```
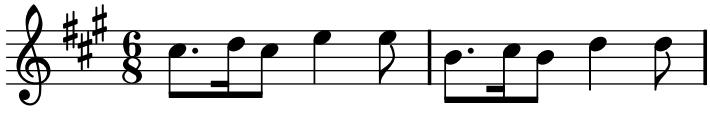


Comme vous pouvez le voir, il faut beaucoup d'apostrophes pour écrire de la musique dans un registre aigu, comme le montre cet extrait de Mozart.

```
{
  \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.

2.5.5 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 147. 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 74.

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 1.2 [À propos de la documentation], page 9. 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 "2.12.1"
\score {
  ...expression musicale composite...  % c'est là qu'est la musique !
  \header { }
  \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 devient un peu plus complexe.

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 63.

> **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.

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 ». 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 5.1.4 [Économie de saisie grâce aux identificateurs et fonctions], page 138.

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 2.3.1 [Les expressions musicales en clair], page 26, 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.

```
\score {
  {   % cette accolade marque le début de l'expression musicale
    \new StaffGroup <<
    ...insérez ici l'intégralité d'un opéra de Wagner...
    >>
  }   % cette accolade marque la fin de l'expression musicale
  \layout { }
}
```

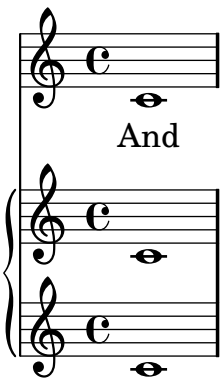
[1] Les noms de variables sont sensibles à la casse, et ne peuvent contenir ni chiffre, ni ponctuation, ni caractère accentué, ni espace.

Un opéra de Wagner multiplierait facilement la longueur de ce manuel par deux ou trois, alors contentons-nous d'une voix et d'un piano. On n'a plus besoin d'une partition d'orchestre — *i.e.* des portées regroupées en `StaffGroup` — donc laissons cela de côté. Par contre, nous voulons bien une voix et un piano.

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

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.

```
\score {
  <<
    \new Staff = "chanteur" <<
      \new Voice = "vocal" { c'1 }
      \addlyrics { And }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "superieur" { c'1 }
      \new Staff = "inferieur" { 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.

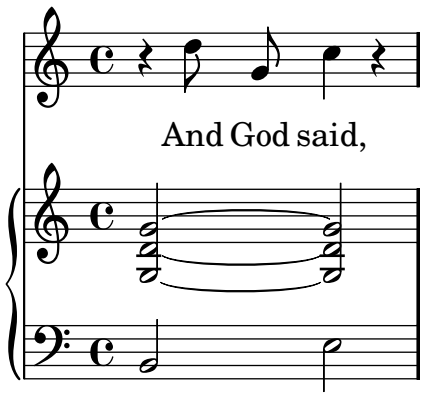
À ce point, on pourrait commencer à ajouter les notes. Dans les accolades qui suivent `\new Voice = chant`, on pourrait commencer à écrire

```
\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. Avec quelques notes de plus, nous pourrions avoir :

```
melodie = \relative c'' { r4 d8\noBeam g, c4 r }
texte   = \lyricmode { And God said, }
superieur  = \relative c'' { <g d g,>2~ <g d g,> }
inferieur  = \relative c { b2 e2 }

\score {
  <<
    \new Staff = "chanteur" <<
      \new Voice = "vocal" { \melodie }
      \addlyrics { \texte }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "superieur" { \superieur }
      \new Staff = "inferieur" {
        \clef "bass"
        \inferieur
      }
    >>
  >>
  \layout { }
}
```



Respectez bien la différence entre les notes – introduites par `\relative` –, et les paroles – introduites par `\lyricmode`. Cette distinction est primordiale afin que LilyPond puisse interpréter ce qui les suit comme étant respectivement de la musique ou du texte.

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 !
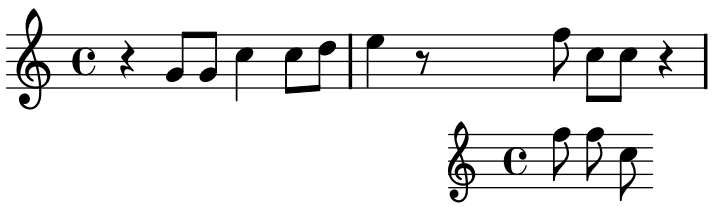
Voir aussi

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 :
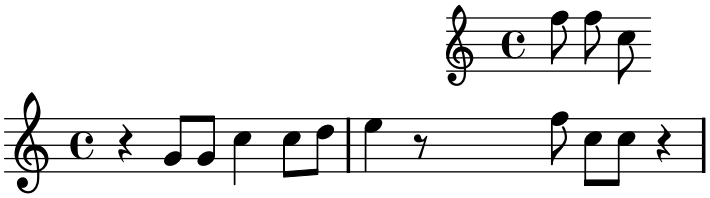
```
\new Staff {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f c c }
      \new Staff {
        f8 f c
      }
    >>
    r4 |
  }
}
```



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

Les ossia apparaissent souvent sans clef ni métrique, et dans une police plus petite. Ceci requiert des commandes dont nous n'avons pas encore parlé. Voir Section 4.3.2 [Taille des objets], page 100 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. Avant d'examiner ces règles, voici une liste des différents types de crochet :
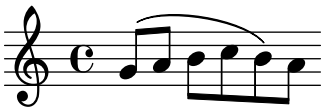
Type de crochet	**Fonction**
`{ .. }`	Délimite un segment de musique séquentielle
`< .. >`	Délimite les notes d'un accord
`<< .. >>`	Délimitent des sections simultanées
`( .. )`	Marque le début et la fin d'une liaison
`\( .. \)`	Marque le début et la fin d'une liaison de phrasé
`[ .. ]`	Marque le début 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.

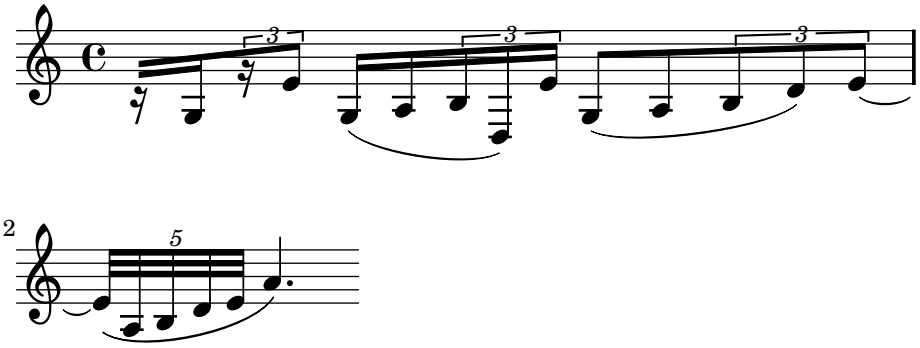
Ainsi, et bien que ce ne soit pas très musical, une liaison de phrasé peut débuter avant l'insertion d'une ligature manuelle et s'arrêter avant la fin de la ligature :

```
{ g8\( a b[ c b\) a] }
```



De manière générale, différents types de crochets, notamment s'ils indiquent des nolets, liaisons de prolongation ou notes d'ornements, 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[ g16 \times 2/3 {r16 e'8] }
  g16( a \times 2/3 {b d) e' }
  g8[( a \times 2/3 {b d') e'~]}
  \times 4/5 {e'32\( a b d' e'} a'4.\)
}
```



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) et qui se trouve être le concept fondamental de LilyPond.
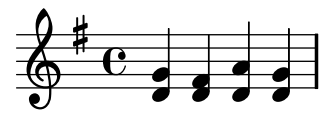
3.2.1 J'entends des Voix

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>
```
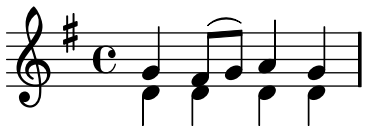


Nous exprimons ici chacun des accords par l'utilisation de chevrons gauche et droite simples, `< ... >`, puisque nous n'avons besoin que d'une seule voix. Supposons maintenant que le fa dièse soit une croche, suivie d'un sol croche – une note de passage vers le la ? Nous avons alors deux notes qui débutent 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 :
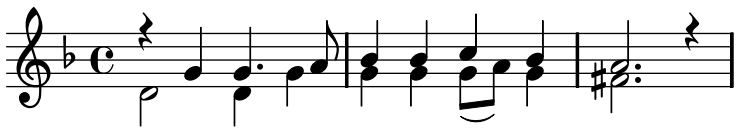
```
\key g \major
%    Voice "1"                   Voice "2"
<< { g4 fis8( g) a4 g }    \\ { d4 d d d }  >> |
```



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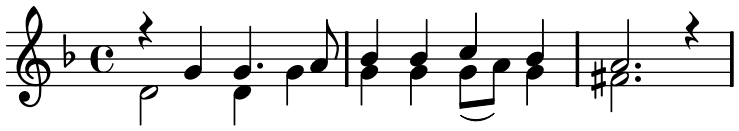
Autre exemple :

```
\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 :

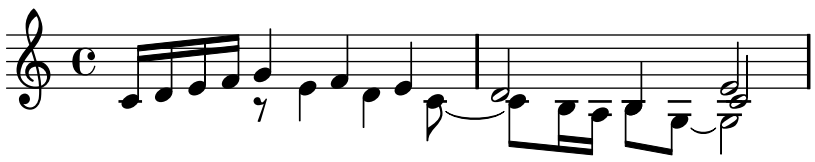
```
\key d \minor
<< {
  % Voice "1"
  r4 g g4. a8 |
  bes4 bes c bes |
  a2. r4 |
} \\ {
  % Voice "2"
  d,2 d4 g |
  g4 g g8( a) g4 |
  fis2. s4 |
} >>
```



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, la direction et l'orientation des liaisons, hampes, nuances, etc. est définie automatiquement.

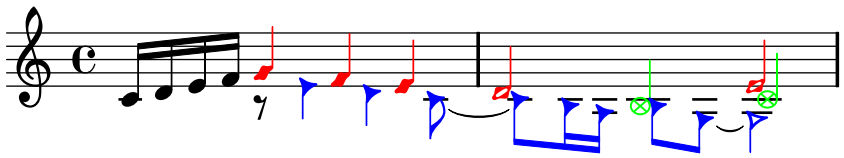
```
\new Staff \relative c' {
  % Voix principale
  c16 d e f
  %    Voice "1"     Voice "2"                  Voice "3"
  << { g4 f e } \\ { r8 e4 d c8 ~ } >> |
  << { d2 e2 }  \\ { c8 b16 a b8 g ~ g2 } \\ { s4 b4 c2 } >> |
}
```



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' {
  % Voix principale
  c16 d e f
  << % Mesure 1
    {
      \voiceOneStyle
      g4 f e
    }
  \\
    {
      \voiceTwoStyle
      r8 e4 d c8 ~
    }
  >>
  << % Mesure 2
     % Suite de la voix 1
    { d2 e2 }
  \\
     % Suite de la voix 2
    { c8 b16 a b8 g ~ g2 }
  \\
    {
      \voiceThreeStyle
      s4 b4 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 notes, 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 96 et Section 4.6.2 [Utilisation de variables dans les retouches], page 132.

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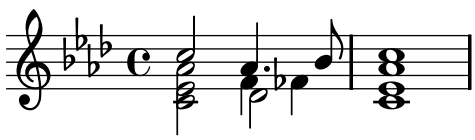
```
\relative c' { noteA << < noteB noteC > \\ noteD >> noteE }
```

`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 :

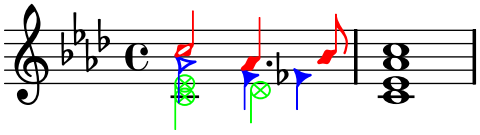
```
\relative c' { noteA ... }
<<
  \relative c'' { < noteB noteC > ... }
\\
  \relative g' { noteD ... }
>>
\relative c' { noteE ... }
```

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 prêtez 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.
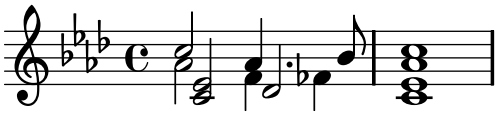


La direction des hampes sert souvent à indiquer dans la continuité deux lignes mélodiques simultanées. Ici, les hampes des notes les plus hautes vont vers le haut, et celles des notes plus basses vers le bas. C'est une première indication que nous avons eu recours à plus d'une voix.

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.
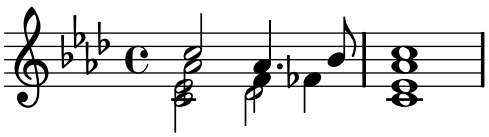


Essayons à présent de coder cette musique en partant de zéro. Comme nous le verrons, certaines difficultés vont se présenter. Partons de ce que nous avons appris : utilisons la construction `<< \\ >>` pour saisir la première mesure dans trois voix :

```
\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 :

```
\new Staff \relative c'' {
  \key aes \major
  << % Voix un
    { c2 aes4. bes8 }
  \\ % Voix deux
    { aes2 f4 fes   }
  \\ % Pas de Voix trois
  \\ % Voix quatre
    { <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 117).

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.
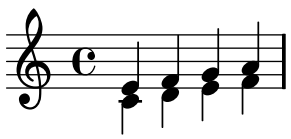
En particulier, la construction `<< \\ >>` que nous avons vue précédemment :

```
\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
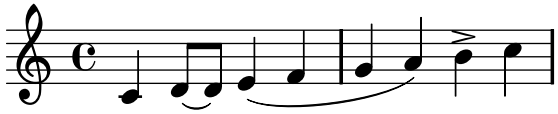


Les commandes `\voiceXXX` fixent la direction des hampes, des liaisons de prolongation et de phrasé, des articulations, des annotations, des points d'augmentation des notes pointées et des doigtés. `\voiceOne` et `\voiceThree` font pointer ces objets vers le haut, alors que `\voiceTwo` et `\voiceFour` 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.
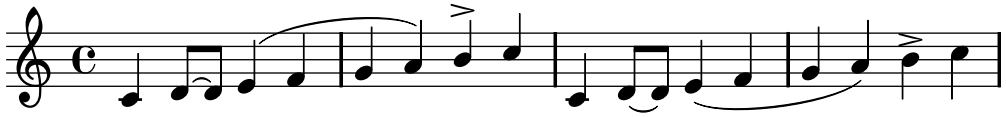
Voyons, à l'aide de ces exemples simples, les effets respectifs de `\oneVoice`, `\voiceOne` et `\voiceTwo` sur les annotations, liaisons de prolongation ou de phrasé, et sur les nuances.

```
\relative c'{
  % Comportement par défaut ou après \oneVoice
```
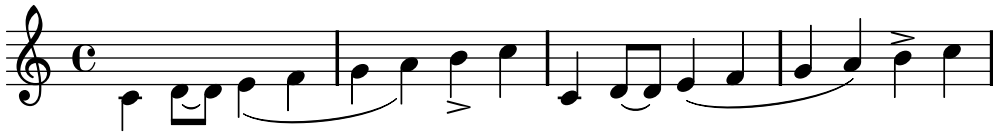
```
  c d8 ~ d e4 ( f g a ) b-> c
}
```



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



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

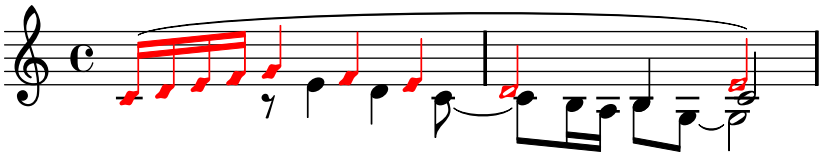


Voyons à présent trois différentes façons d'exprimer un passage polyhonique, à partir d'un exemple de la section précédente. Chacune d'elles aura ses avantages selon les circonstances.

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.

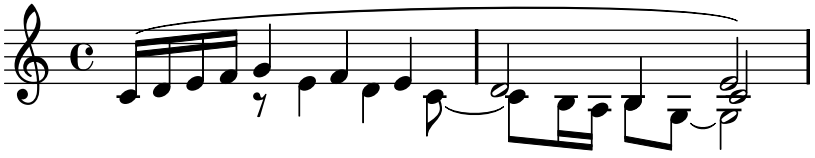
```
\new Staff \relative c' {
  \voiceOneStyle
  % Les notes qui suivent sont monophoniques
  c16^( d e f
  % Début d'une section de trois voix simultanées
  <<
    % Poursuite de la voix principale en parallèle
    { g4 f e | d2 e2) }
    % Initialisation de la seconde voix
    \new Voice {
      % Hampes et autres attributs iront vers le bas
      \voiceTwo
      r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2
    }
```

```
    % Initialisation de la troisième voix
    \new Voice {
      % Hampes et autres attributs iront vers le haut
      \voiceThree
      s2. | s4 b4 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 :

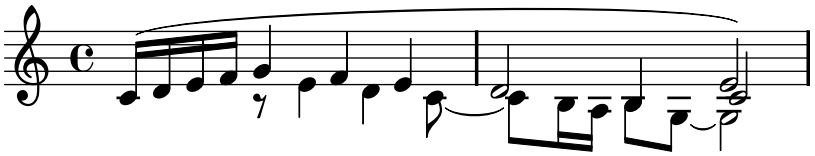
```
\new Staff \relative c' {
  c16^( d e f
  <<
    { g4 f e | d2 e2) }
    \new Voice {
      \voiceTwo
      r8 e4 d c8 ~ |
      <<
        {c8 b16 a b8 g ~ g2}
        \new Voice {
          \voiceThree
          s4 b4 c2
        }
      >>
    }
  >>
}
```



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' <<
  % Initialisation de la première voix
  \new Voice {
    \voiceOne
    c16^( d e f g4 f e | d2 e2) |
  }
  % Initialisation de la seconde voix
  \new Voice {
```

```
    % Hampes et autres attributs iront vers le bas
    \voiceTwo
    s4 r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2 |
  }
  % Initialisation de la troisième voix
  \new Voice {
    % Hampes et autres attributs iront vers le haut
    \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 122.

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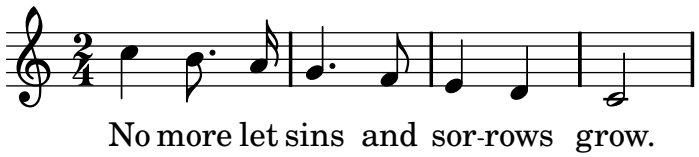
Manuel de notation : Section “Plusieurs voix” dans *Manuel de notation*.

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 = "un" \relative c'' {
    \autoBeamOff
    \time 2/4
    c4 b8. a16 g4. f8 e4 d c2
  }
  \new Lyrics \lyricsto "un" {
    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`.
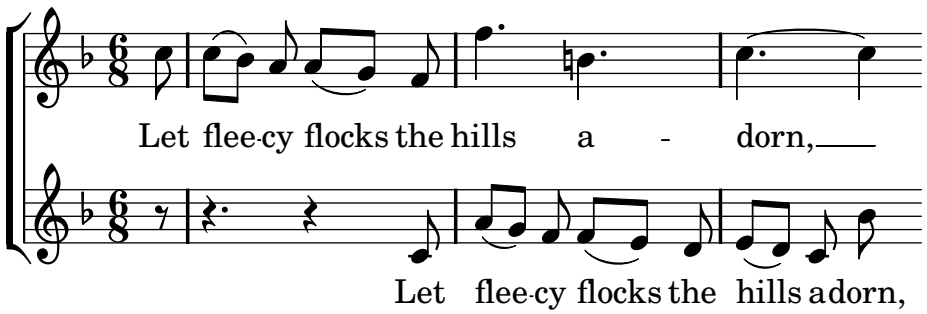
Si la ligature automatique que LilyPond applique par défaut est pleinement adaptée en matière de musique instrumentale, il n'en va pas de même dans le cas d'une musique associée à des paroles, et pour laquelle soit les ligatures sont carrément absentes, soit elles servent à indiquer un mélisme – plusieurs 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 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 = { \time 6/8 \partial 8 \key f \major}
SopUnMusique = \relative c'' {
  c8 | c([ bes)] a a([ g)] f | f'4. b, | c4.~ c4 }
SopDeuxMusique = \relative c' {
  r8 | r4. r4 c8 | a'([ g)] f f([ e)] d | e([ d)] c bes' }
SopUnParoles = \lyricmode {
  Let | flee -- cy flocks the | hills a -- dorn, __ }
SopDeuxParoles = \lyricmode {
  Let | flee -- cy flocks the | hills a -- dorn, }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "SopUn" {
        \global
        \SopUnMusique
      }
      \new Lyrics \lyricsto "SopUn" {
        \SopUnParoles
      }
    >>
    \new Staff <<
```

```
      \new Voice = "SopDeux" {
        \global
        \SopDeuxMusique
      }
      \new Lyrics \lyricsto "SopDeux" {
        \SopDeuxParoles
      }
    >>
  >>
}
```



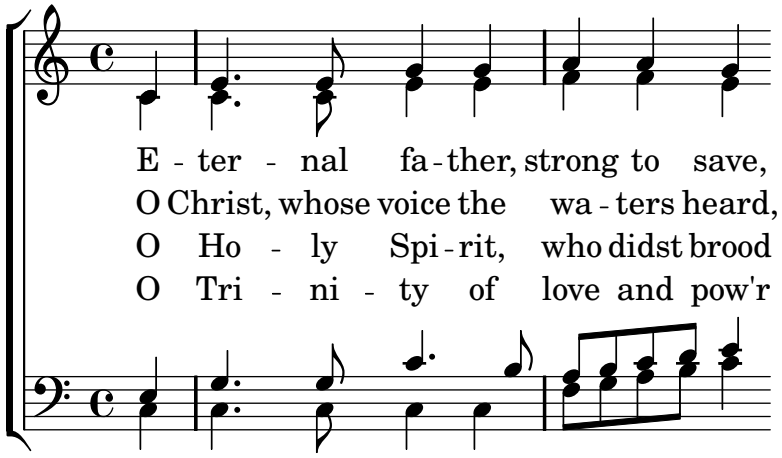
Voici donc la structure de base valable pour toute partition vocale. On peut y ajouter d'autres portées si besoin est, d'autres voix à chaque portée, plusieurs couplets aux paroles, et les variables contenant la musique peuvent même être stockées dans des fichiers indépendants dès lors que leur longueur 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 les notes et les paroles. Vous noterez aussi une variable particulière, que nous avons appelée 'MetriqueArmure', et qui contient plusieurs commandes que nous utiliserons dans les deux portées. Dans de nombreux autres exemples, elle s'appelle 'global'.

```
MetriqueArmure = { \time 4/4 \partial 4 \key c \major}
SopMusique   = \relative c' { c4 | e4. e8 g4  g  | a a g }
AltoMusique  = \relative c' { c4 | c4. c8 e4  e  | f f e }
TenorMusique = \relative c  { e4 | g4. g8 c4. b8 | a8 b c d e4 }
BasseMusique  = \relative c  { c4 | c4. c8 c4  c  | f8 g a b c4 }
CoupletUn   = \lyricmode {
  E -- | ter -- nal fa -- ther, | strong to save, }
CoupletDeux   = \lyricmode {
  O | Christ, whose voice the | wa -- ters heard, }
CoupletTrois = \lyricmode {
  O | Ho -- ly Spi -- rit, | who didst brood }
CoupletQuatre  = \lyricmode {
  O | Tri -- ni -- ty of | love and pow'r }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \clef "treble"
      \new Voice = "Sop"  { \voiceOne \MetriqueArmure \SopMusique }
      \new Voice = "Alto" { \voiceTwo \AltoMusique }
      \new Lyrics \lyricsto "Sop" { \CoupletUn   }
```

```
      \new Lyrics \lyricsto "Sop" { \CoupletDeux   }
      \new Lyrics \lyricsto "Sop" { \CoupletTrois }
      \new Lyrics \lyricsto "Sop" { \CoupletQuatre  }
    >>
    \new Staff <<
      \clef "bass"
      \new Voice = "Tenor" { \voiceOne \MetriqueArmure \TenorMusique }
      \new Voice = "Basse"  { \voiceTwo \BasseMusique  }
    >>
  >>
}
```

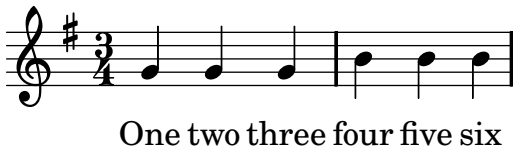


Nous allons terminer en voyant comment coder un couplet pour soliste suivi d'un refrain à deux voix sur deux portées. Les explications sont importantes, dans la mesure où les moyens mis en œuvre pour arriver à enchaîner le solo et la polyphonie dans une seule et même partition sont quelque peu tirés par les cheveux.

Commençons par ouvrir un bloc `score` qui contiendra un `ChoirStaff`, puisque nous aimerions voir un crochet au début du système choral. Nous devrions avoir, après `\new ChoirStaff`, un double chevron gauche pour synchroniser les portées ; mais comme nous reportons le parallélisme après le solo, nous utilisons des accolades – un double chevron ne serait cependant pas génant. À l'intérieur du `ChoirStaff`, nous voulons en premier la portée avec le couplet. Puisqu'elle englobe parallèlement des notes et des paroles, nous devons encadrer les `\new Voice` et `\new Lyrics` de doubles chevrons gauche/droite pour les faire démarrer de concert :

```
coupletnotes = \relative c'' {
  \clef "treble"
  \key g \major
  \time 3/4 g g g b b b
}
coupletparoles = \lyricmode {
  One two three four five six
}
\score {
  \new ChoirStaff {
    \new Staff <<
      \new Voice = "couplet" {
        \coupletnotes \break
      }
      \new Lyrics \lyricsto couplet {
```

```
        \coupletparoles
      }
    >>
  }
}
```



Voici la ligne du couplet réalisée.
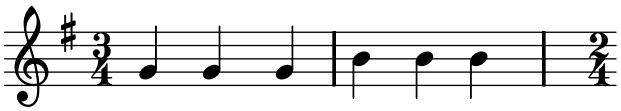
Nous poursuivons avec refrainA, sur la même portée, alors même qu'une deuxième portée s'amorce en parallèle pour contenir refrainB. Cette section parallèle doit s'enchaîner directement à la suite du `\break` de la voix contenant le couplet – il s'agit bien de la *même* voix. Voici cette section parallèle. On pourrait tout à fait ajouter encore d'autres portées ici, toujours de la même manière.

```
<<
  \refrainnotesA
  \new Lyrics \lyricsto verse {
    \refrainparolesA
  }
  \new Staff <<
    \new Voice = "refrainB" {
      \refrainnotesB
    }
    \new Lyrics \lyricsto "refrainB" {
      \refrainparolesB
    }
  >>
>>
```
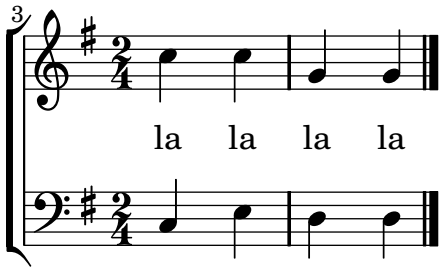
Et voici le résultat final, avec ses deux portées pour la partie chorale, et qui montre comment la section en parallèle s'enchaîne avec la voix du couplet :

```
coupletnotes = \relative c'' {
  \clef "treble"
  \key g \major
  \time 3/4 g g g b b b
}
refrainnotesA = \relative c'' {
  \time 2/4
  c c g g \bar "|."
}
refrainnotesB = \relative c {
  \clef "bass"
  \key g \major
  c e d d
}
coupletparoles = \lyricmode {
  One two three four five six
}
refrainparolesA = \lyricmode {
```

```
  la la la la
}
refrainparolesB = \lyricmode {
  dum dum dum dum
}
\score {
  \new ChoirStaff {
    \new Staff <<
      \new Voice = "couplet" {
        \coupletnotes \break
        <<
          \refrainnotesA
          \new Lyrics \lyricsto "couplet" {
            \refrainparolesA
          }
          \new Staff <<
            \new Voice = "refrainB" {
              \refrainnotesB
            }
            \new Lyrics \lyricsto "refrainB" {
              \refrainparolesB
            }
          >>
        >>
      }
      \new Lyrics \lyricsto "couplet" {
        \coupletparoles
      }
    >>
  }
}
```



One two three four five six



dum dum dum dum

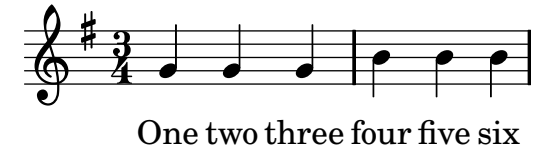
Bien que ce que nous venons de voir constitue un exercice intéressant et fort utile pour comprendre comment s'articulent des blocs séquentiels et simultanés, nous aurions aussi pu coder notre exemple sous la forme de deux blocs `\score` au sein d'un bloc `\book` implicite :

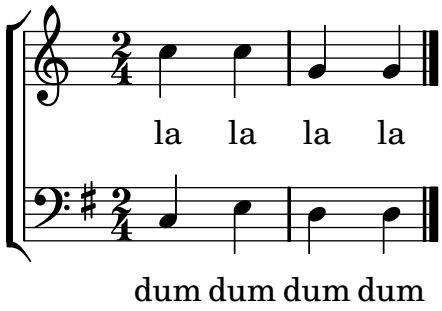
```
coupletnotes = \relative c'' {
```

```
  \clef "treble"
  \key g \major
  \time 3/4 g g g b b b
}
refrainnotesA = \relative c'' {
  \time 2/4
  c c g g \bar "|."
}
refrainnotesB = \relative c {
  \clef "bass"
  \key g \major
  c e d d
}
coupletparoles = \lyricmode {
  One two three four five six
}
refrainparolesA = \lyricmode {
  la la la la
}
refrainparolesB = \lyricmode {
  dum dum dum dum
}
\score {
  \new Staff <<
    \new Voice = "couplet" {
      \coupletnotes
    }
    \new Lyrics \lyricsto "couplet" {
      \coupletparoles
    }
  >>
}

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "refrainA" {
        \refrainnotesA
      }
      \new Lyrics \lyricsto "refrainA" {
        \refrainparolesA
      }
    >>
    \new Staff <<
      \new Voice = "refrainB" {
        \refrainnotesB
      }
      \new Lyrics \lyricsto "refrainB" {
        \refrainparolesB
      }
    >>
  >>
```

```
}
```





Voir aussi

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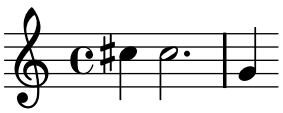
3.3 Contextes et graveurs

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 :

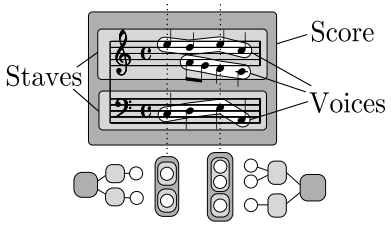
```
cis4 cis2. g4
```



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 tablatures d'instruments frettés, la basse chiffrée, etc.

Le nom de chacun des contextes est formé d'un ou plusieurs mots aux initiales en majuscule 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.

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 indispenssable 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 :
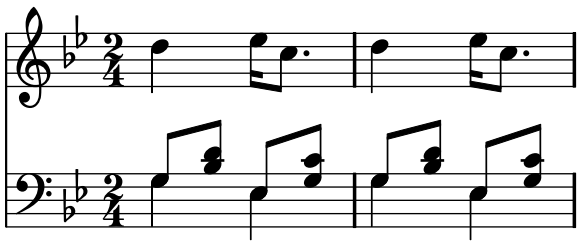
```
\new type expression-musicale
```

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.

Notez bien qu'il n'existe pas de commande `\new Score`, puisque le contexte premier que constitue `Score` est créé par `\score`.

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 largement commentée :

```
\score {  % début de l'unique expression musicale composée
  <<  % début d'une section de portées simultanées
    \time 2/4
    \new Staff {  % création de la portée MD
      \key g \minor
      \clef "treble"
      \new Voice {  % création d'une voix pour les notes de MD
        \relative c'' {  % début des notes de MD
          d4 ees16 c8. |
          d4 ees16 c8. |
        }  % fin des notes de MD
      }  % fin de la voix MD
    }  % fin de la portée MD
    \new Staff <<  % création de la portée MG ; nécessite deux voix simultanées
      \key g \minor
      \clef "bass"
      \new Voice {  % création de la voix un de MG
        \voiceOne
        \relative g {  % début des notes de la voix un de MG
          g8 <bes d> ees, <g c> |
          g8 <bes d> ees, <g c> |
        }  % fin des notes de la voix un de MG
      }  % fin de la voix un de MG
      \new Voice {  % création de la voix deux de MG
        \voiceTwo
        \relative g {  % début des notes de la voix deux de MG
          g4 ees |
          g4 ees |
        }  % fin des notes de la voix deux de MG
      }  % fin de la voix deux de MG
    >>  % fin de la portée MG
  >>  % fin de la section de portées simultanées
}  % fin de l'unique expression musicale composée
```



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 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. 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.

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
Clef_engraver	Grave les clefs
Completion_heads_engraver	Divise les notes qui dépassent de la mesure
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

Références 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.
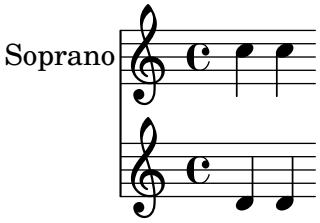
propriétéNommée	**Type**	**Fonction**	**Exemple de valeur**
extraNatural	Booléen	Si vrai, ajoute un bécarre avant une altération accidentelle	`#t`, `#f`
currentBarNumber	Entier	Détermine le numéro de la mesure en cours	`50`
doubleSlurs	Booléen	Si vrai, imprime les liaisons au-dessous **et** au-dessus 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 en 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"  % Mauvais !
  d4 d
 }
>>
```
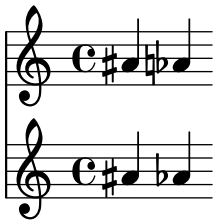


Dans la mesure où le nom de contexte par défaut est `Voice`, la deuxième commande `\set` a défini « Alto » comme propriété `instrumentName` du contexte de voix. Puisque LilyPond n'ira pas chercher une telle propriété dans la contexte `Voice`, celle-ci ne sera pas interprétée. Il ne s'agit pas d'une erreur, 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 le Manuel de références 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.
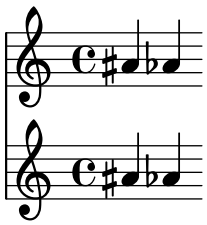
Voici comment supprimer les bécarres supplémentaires pour une portée :

```
<<
  \new Staff \relative c'' {
    ais4 aes
 }
  \new Staff \relative c'' {
    \set Staff.extraNatural = ##f
    ais4 aes
 }
>>
```



et pour toutes les portées :

```
<<
  \new Staff \relative c'' {
    ais4 aes
 }
  \new Staff \relative c'' {
    \set Score.extraNatural = ##f
    ais4 aes
 }
>>
```
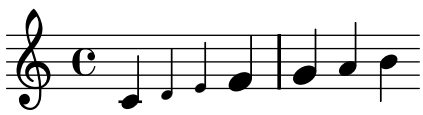


Autre exemple, si la propriété `clefOctavation` est déterminée au niveau du contexte `Score`, elle modifiera la valeur de l'octave en cours pour toutes les portées actives ; cette valeur sera considérée comme étant la nouvelle valeur par défaut pour toutes les portées à venir.

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.

```
c4
% pour obtenir des têtes de note plus petites
\set fontSize = #-4
d e
% pour obtenir des têtes de note plus grosses
\set fontSize = #2.5
f g
% retour à la taille par défaut
\unset fontSize
a b
```



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.
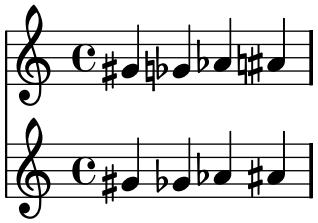
Définition des propriétés de contexte avec \with

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'' {
    gis ges aes ais
  }
  \new Staff \with { extraNatural = ##f }
  \relative c'' {
    gis 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 par défaut.
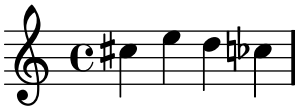
La propriété `fontSize` constitue une exception : lorsqu'elle est déterminée au sein d'un bloc `\with`, cela redéfinit la valeur par défaut de la taille de fonte. Une modification est possible par la commande `\set`, mais la commande `\unset fontSize` fera revenir à la nouvelle valeur par défaut.

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
    }
  }
}
```



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*.

Références 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
  \set fontSize = #-4  % pour obtenir des têtes de note plus petites
  d e
  \set fontSize = #2.5  % pour obtenir des têtes de note plus grosses
  f g
  \unset fontSize  % retour à la taille par défaut
  a b
}
```
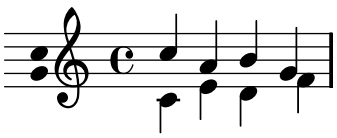


Vous pouvez aussi ajouter individuellement un graveur à un contexte. La commande se formule ainsi :

`\consists` *Nom_du_graveur*
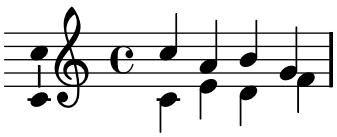
et se place dans un bloc `\with`. Certaines partitions vocales font apparaître un Section "ambitus" dans *Glossaire* au début de la portée, afin d'indiquer ses notes extrêmes. L'ambitus est généré par l'`Ambitus_engraver`, que l'on peut adjoindre à n'importe quel contexte. Si nous l'ajoutons au contexte `Voice`, seule la tessiture de cette voix sera calculée :

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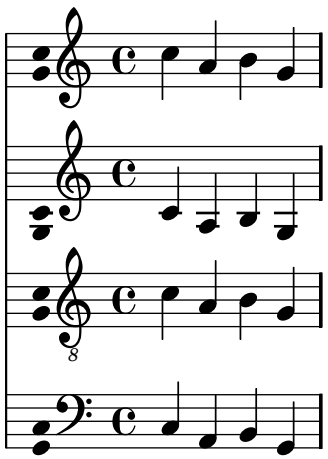


Modification de tous les contextes d'un même type

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'' { c a b g }
    >>
    \new Staff <<
      \relative c' { c a b g }
    >>
    \new Staff <<
      \clef "G_8"
      \relative c' { c a b g }
    >>
    \new Staff <<
      \clef "bass"
      \relative c { c 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*.

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 147, 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 "2.12.1"
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 "2.12.1"
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 "2.12.1"
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 "2.12.1"

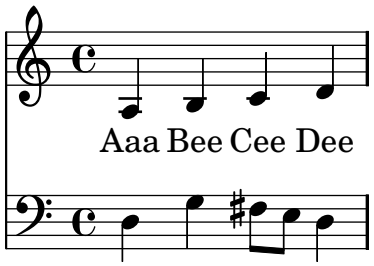
sopranoMusique = \relative c' {
  \clef treble
  \key c \major
  \time 4/4
  a4 b c d
}

sopranoParoles = \lyricmode {
  Aaa Bee Cee Dee
}

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
    >>
    \new Staff \violoncelleMusique
  >>
  \layout { }
  \midi { }
```

```
}
```

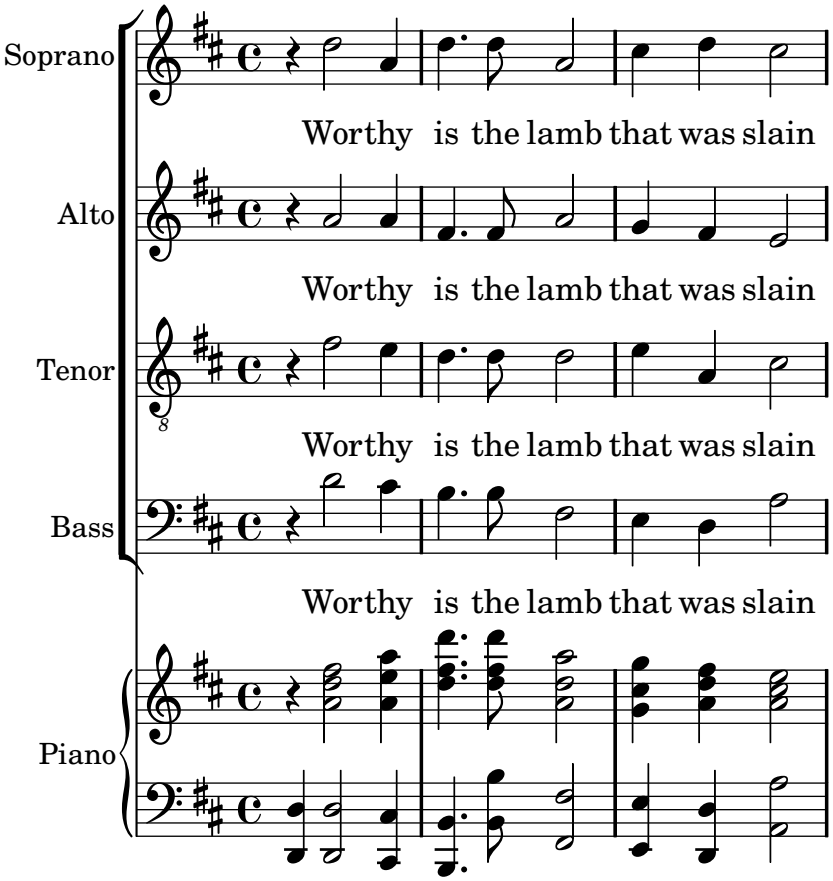


Voir aussi

Les patrons originaux sont disponibles à l'annexe « Modèles », voir Section A.1 [Portée unique], page 147.

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 147 – 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" { \tenorParoless }
  \new Staff = "basses" <<
    \set Staff.instrumentName = #"Bass"
    \new Voice = "basses" { \global \basseMusique }
  >>
  \new Lyrics \lyricsto "basses" { \basseParoles }
>>  % 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 and PianoStaff one above the other
  \new ChoirStaff <<
    \new Staff = "sopranos" <<
      \new Voice = "sopranos" { \global \sopranoMusique }
    >>
    \new Lyrics \lyricsto "sopranos" { \sopranoParoless }
    \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 \bassesMusique }
    >>
    \new Lyrics \lyricsto "basses" { \bassesParoles }
  >>  % end 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 :

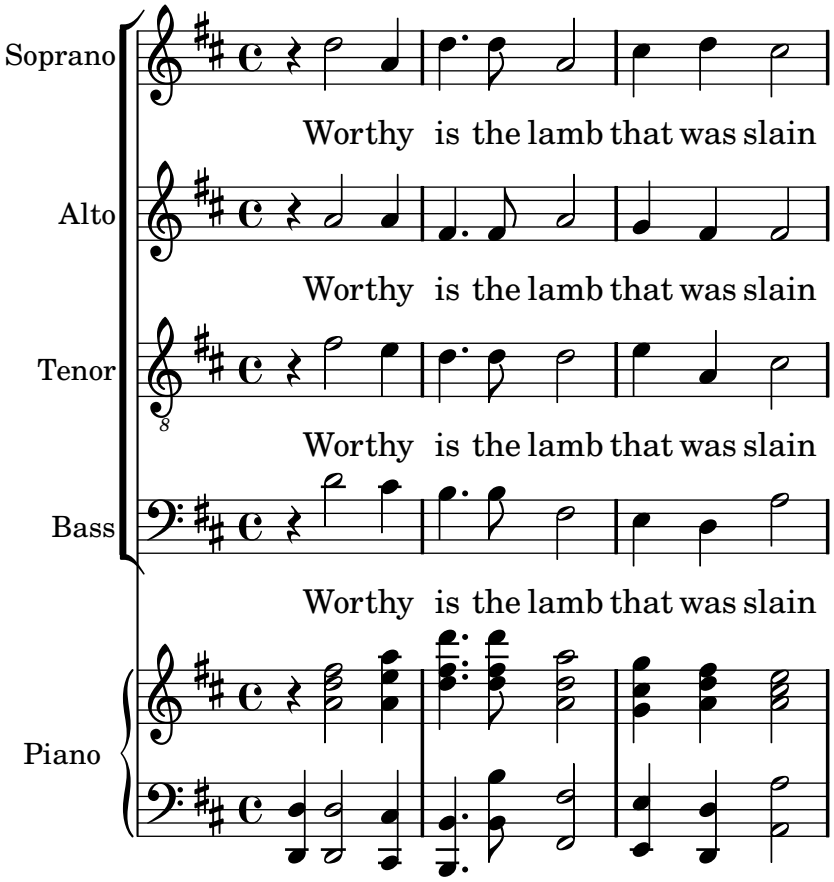
```
\version "2.12.1"

global = { \key d \major \time 4/4 }
sopranoMusique = \relative c'' {
  \clef "treble"
  r4 d2 a4 | d4. d8 a2 | cis4 d cis2 |
}
sopranoParoles = \lyricmode {
  Wor -- thy is the lamb that was slain
}
altoMusique = \relative a' {
  \clef "treble"
  r4 a2 a4 | fis4. fis8 a2 | g4 fis fis2 |
}
altoParoles = \sopranoParoles
tenorMusique = \relative c' {
  \clef "G_8"
  r4 fis2 e4 | d4. d8 d2 | e4 a, cis2 |
}
tenorParoles = \sopranoParoles
basseMusique = \relative c' {
  \clef "bass"
  r4 d2 cis4 | b4. b8 fis2 | e4 d a'2 |
}
basseParoles = \sopranoParoles
superieur = \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 |
}
inferieur = \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 {
  <<  % combinaison en parallèle du ChoirStaff et du PianoStaff
    \new ChoirStaff <<
      \new Staff = "soprani" <<
        \set Staff.instrumentName = #"Soprano"
        \new Voice = "soprani" { \global \sopranoMusique }
      >>
      \new Lyrics \lyricsto "soprani" { \sopranoParoles }
      \new Staff = "alti" <<
        \set Staff.instrumentName = #"Alto"
        \new Voice = "alti" { \global \altoMusique }
      >>
      \new Lyrics \lyricsto "alti" { \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 \basseMusique }
      >>
      \new Lyrics \lyricsto "basses" { \basseParoles }
    >>  % fin du ChoirStaff (système pour chœur)

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



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 "2.12.1"
\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
MetriqueArmure = { \time 4/4 \key c \minor }
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" <<
        \TimeKey  % définition de la métrique et de l'armure
```

```
        \clef "treble"
        \new Voice { \voiceOne \ManuelUnVoixUnMusique }
        \new Voice { \voiceTwo \ManuelUnVoixDeuxMusique }
      >>  % fin du contexte de la portée ManuelUn
      \new Staff = "ManuelDeux" <<
        \TimeKey
        \clef "bass"
        \new Voice { \ManuelDeuxMusique }
      >>  % fin du contexte de la portée ManuelDeux
    >>  % fin du contexte PianoStaff
    \new Staff = "PedalierOrgue" <<
      \TimeKey
      \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 "2.12.1"
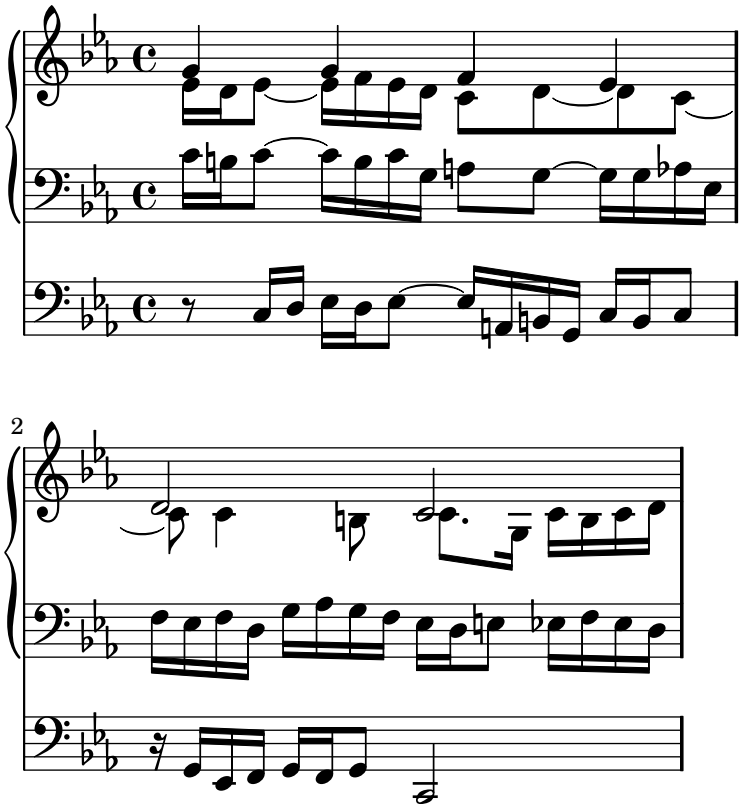
\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
MetriqueArmure = { \time 4/4 \key c \minor }
MusiqueManuelUnVoixUn = \relative g' {
  g4 g f ees | d2 c2 |
}
MusiqueManuelUnVoixDeux = \relative c' {
  ees16 d ees8~ ees16 f ees d c8 d~ d c~ |
  c c4 b8 c8. g16 c b c d |
}
MusiqueManuelDeux = \relative c' {
  c16 b c8~ c16 b c g a8 g~ g16 g aes ees |
  f ees f d g aes g f ees d e8~ ees16 f ees d |
}
MusiquePedalierOrgue = \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 {
  <<  % système pianistique et portée de pédalier simultanés
    \new PianoStaff <<
      \new Staff = "ManuelUn" <<
        \MetriqueArmure  % définition de la métrique et de l'armure
        \clef "treble"
        \new Voice { \voiceOne \MusiqueManuelUnVoixUn }
        \new Voice { \voiceTwo \MusiqueManuelUnVoixDeux }
      >>  % fin du contexte de portée ManuelUn
```

```
      \new Staff = "ManuelDeux" <<
        \MetriqueArmure
        \clef "bass"
        \new Voice { \MusiqueManuelDeux }
      >>  % fin du contexte de portée ManuelDeux
    >>  % fin du contexte PianoStaff
    \new Staff = "PedalierOrgue" <<
      \MetriqueArmure
      \clef "bass"
      \new Voice { \MusiquePedalierOrgue }
    >>  % fin du contexte de portée PedalierOrgue
  >>
}  % fin du contexte Score
```

Jesu, meine Freude

J S Bach



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 63, 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 d'« 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 63, de la façon de nommer les objets. Voici maintenant une liste de référence des types d'objets et de propriétés les plus courants, avec leur convention 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 comme ils se présentent.

Type d'objet/propriété	**Convention de désignation**	**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'objets sont modifiées par des commandes différentes, si bien qu'il est bon de savoir reconnaître le type d'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 67 et dans Section 3.3.5 [Ajout et suppression de graveurs], page 71, 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
```

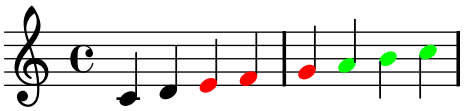
Elle attribue à la propriété appelée *propriété-rendu*, associée à l'objet *ObjetDeRendu*, appartenant lui-même au contexte *Contexte*, une valeur *valeur*.

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 :

```
c d
\override NoteHead #'color = #red
e f g
\override NoteHead #'color = #green
a b c
```



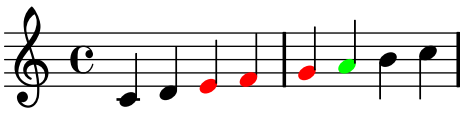
La commande \revert

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 :
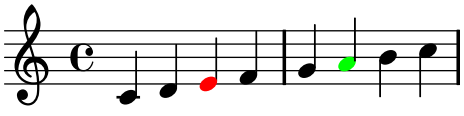
```
c d
\override NoteHead #'color = #red
e f g
\override NoteHead #'color = #green
a
\revert NoteHead #'color
b c
```



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 :

```
c d
\once \override NoteHead #'color = #red
e f g
\once \override NoteHead #'color = #green
a b c
```



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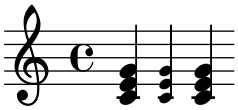
Il existe une autre forme de commande `\override`, `\overrideProperty`, qui est parfois utile. Nous la mentionnons ici par souci d'exhaustivité ; pour le détail, voir Section "Retouches complexes" dans *Manuel de notation*.

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` ne modifie 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>
<c e g>
```
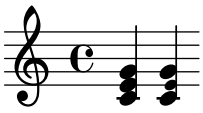


Nous voyons que `\override` modifie *toutes* les têtes de notes de l'accord, car toutes les notes de l'accord surviennent au même *instant musical* et que la fonction de `\once` est de faire porter la modification sur tous les objets du type spécifié qui surviennent au même instant musical, comme le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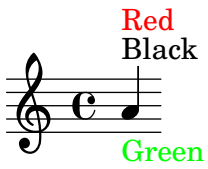


Vous noterez que la syntaxe de `\tweak` est différente de celle de la commande `\override`. Ni le contexte, ni l'objet n'ont besoin d'être spécifiés ; au contraire, cela produirait une erreur si on le faisait. Tous deux sont sous-entendus par l'élément suivant dans la chaîne de saisie. La syntaxe générale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

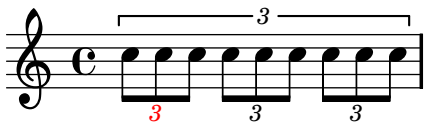
```
a ^Black
        -\tweak #'color #red ^Red
        -\tweak #'color #green _Green
```



Attention : la commande `\tweak` doit être précédée d'une marque d'articulation, comme si elle-même était une articulation.
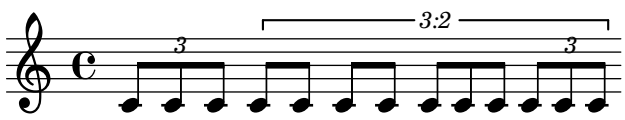
Quand plusieurs nolets sont imbriqués et commencent au même instant musical, c'est encore la commande `\tweak` qui est utilisée pour changer l'apparence de l'un d'entre eux. Dans l'exemple suivant, le long crochet de nolet et le premier des trois crochets courts commencent au même instant musical ; une 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[ c8 c8] }
        \times 2/3 { c8[ c8 c8] }
        \times 2/3 { c8[ c8 c8] }
}
```



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 {
        c[ c]
        c[ 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 tweak" dans *Manuel de notation*.

4.2 Le manuel des références internes
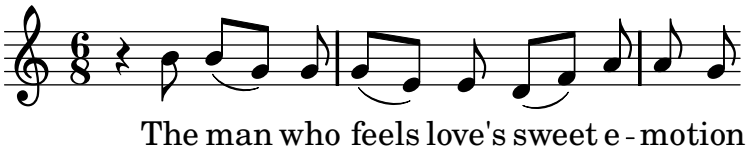
4.2.1 Propriétés des objets de rendu

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 :

```
{
  \time 6/8
  {
    r4 b8 b[( g]) g |
    g[( e]) e d[( f]) a |
    a 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 (number)
     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 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'explications 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.
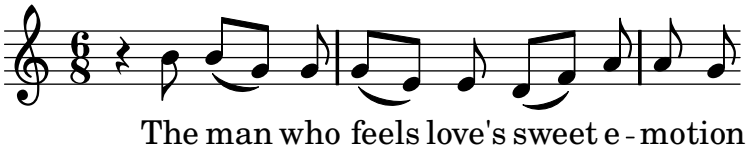
Pour construire la commande `\override`, il suffit donc de remplacer les valeurs que nous avons trouvées en guise de noms, en laissant de côté le contexte. Commençons par une valeur très élevée dans un premier temps, pour nous assurer que la commande fonctionne. Nous obtenons :

```
\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 :

```
{
  \time 6/8
  {
    % L'épaisseur de toutes les liaisons à venir passe de 1,2 à 5,0
    \override Slur #'thickness = #5.0
    r4 b8 b[( g]) g |
    g[( e]) e d[( f]) a |
    a 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.
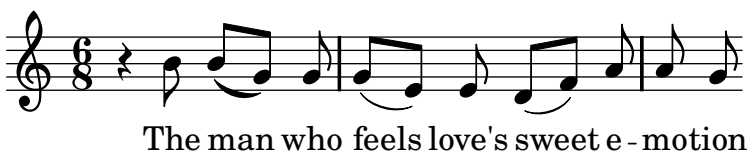
Détermination du contexte adéquat

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 :

```
{
  \time 6/8
  {
    r4 b8
    % Épaississement de la prochaine liaison seulement
    \once \override Slur #'thickness = #5.0
    b[( g]) g |
    g[( e]) e d[( f]) a |
    a g
  }
  \addlyrics {
    The man who feels love's sweet e -- mo -- tion
  }
}
```



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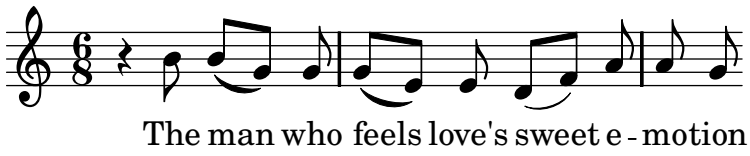
La commande `\once` peut aussi être utilisée devant la commande `\set`.

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 :
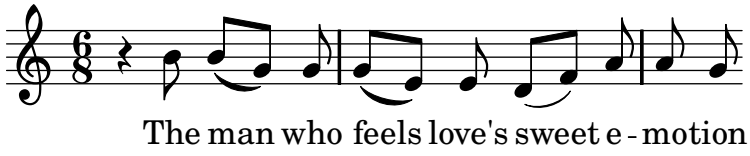
```
{
  \time 6/8
  {
    r4 b8
```

```
    % Épaississement de la prochaine liaison seulement
    \once \override Slur #'thickness = #5.0
    b[( g]) g |
    % Épaississement de la prochaine liaison seulement
    \once \override Slur #'thickness = #5.0
    g[( e]) e d[( f]) a |
    a 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 .

```
{
  \time 6/8
  {
    r4 b8
    % L'épaisseur de toutes les liaisons à venir passe de 1,2 à 5,0
    \override Slur #'thickness = #5.0
    b[( g]) g |
    g[( e])
    % Retour à l'épaisseur par défaut de 1,2 pour les prochaines liaisons
    \revert Slur #'thickness
    e d[( f]) a |
    a 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 de 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 Annexe B [Tutoriel Scheme], page 149.
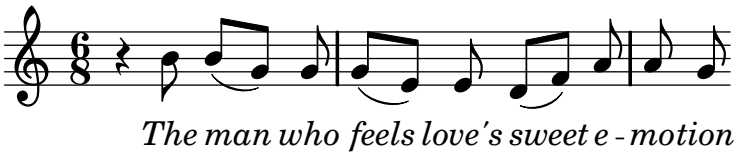
Ainsi, la commande `\override` pour mettre les paroles en italique est :

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

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

```
{
  \time 6/8
  {
    r4 b8 b[( g]) g |
    g[( e]) e d[( f]) a |
    a 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'initiation : Annexe B [Tutoriel Scheme], page 149.

4.2.3 Types de propriétés

Nous avons vu jusqu'à maintenant deux types de propriétés : `nombre` et `symbol`. 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 (anglais *Boolean*)	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`
Durée (*Moment* en anglais)	Une durée de note construite avec la fonction make-moment	`(ly:make-moment 1 4)`, `(ly:make-moment 3 8)`
Entier (*Integer* en anglais)	Un nombre entier positif	`3`, `1`
Inconnu (*Unknown* en anglais)	Un processus, ou `#f` pour empêcher toute action	`bend::print`, `ly:text-interface::print`, `#f`
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." }`
Nombre	Une valeur décimale positive ou négative	`3.5`, `-2.45`
Paire (de nombres)	Deux nombres séparées par 'espace . 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`
Vecteur	Une liste de trois éléments encadrés par des parenthèses et précédés par apostrophe-hash, `'#`	`'#(#t #t #f)`

Voir aussi

Manuel d'initiation : Annexe B [Tutoriel Scheme], page 149.

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 94. 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.
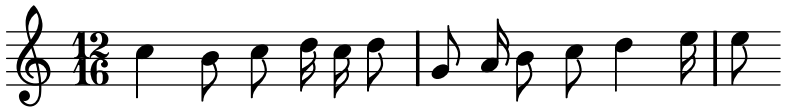
Nous avons vu, dans Section 4.2.1 [Propriétés des objets de rendu], page 90, comment trouver les informations sur les *grobs*. Nous procédons de la même manière et consultons la RPI pour connaître l'objet chargé d'imprimer les barres de mesure. En cliquant sur *Backend* puis sur *Tous les objets de rendu*, 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
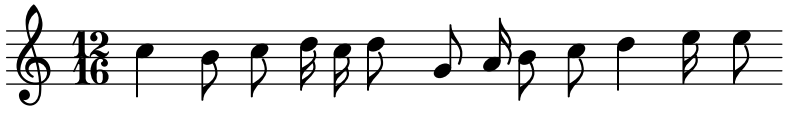
Cette propriété contrôle l'apparence des barres de mesure en précisant le type de symbole (glyphe) à imprimer. Comme pour de nombreuses autres propriétés, on peut lui indiquer de ne rien imprimer en lui attribuant la valeur `#f`. Essayons en laissant de côté, une fois encore, le contexte concerné (`Voice` en l'occurrence) :

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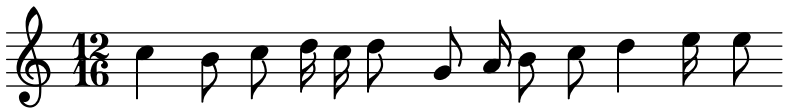


Cette fois, les barres de mesure ont disparu.

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, a16 b8 c d4 e16 |
  e8
}
```
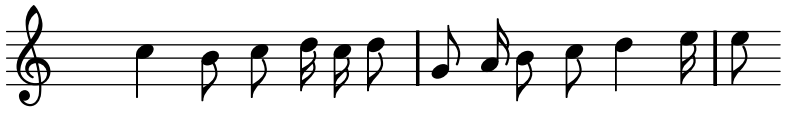


Comme on peut le constater, cette solution-là aussi supprime les barres de mesure.

transparence
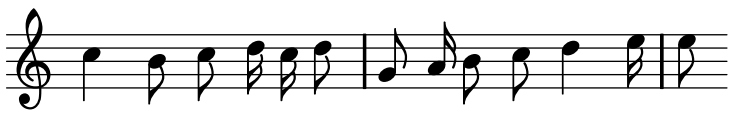
La RPI mentionne, à la page sur `grob-interface`, que la propriété `transparent` attend comme argument un boléen. Il faudrait donc mettre `#t` pour rendre l'objet transparent. Dans cet exemple, essayons de rendre transparente la métrique (*time signature* en anglais) plutôt que les barres de mesure. 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, 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 à 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, a16 b8 c d4 e16 |
  e8
}
```
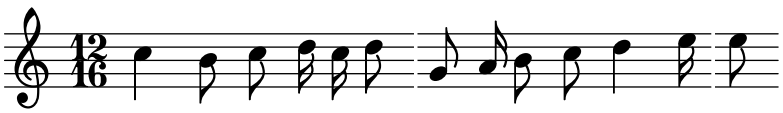


La différence est flagrante : le fait d'attribuer au stencil la valeur `#f` supprime totalement l'objet, tandis que le fait de le rendre `transparent` le laisse en place, mais de façon invisible.

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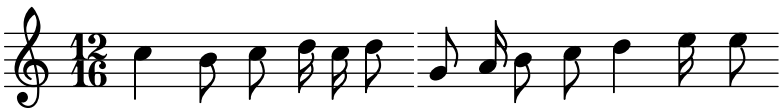
```
{
  \time 12/16
  \override Staff.BarLine #'color = #white
  c4 b8 c d16 c d8 |
  g, a16 b8 c d4 e16 |
  e8
}
```



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, a16 b8 c d4 e16 |
  e8
}
```
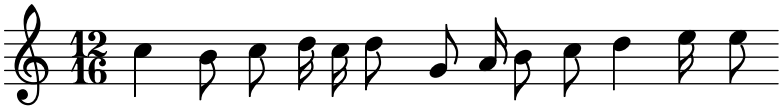


Vous noterez que, dans ce cas, la fonction `x11-color` admet un symbole comme argument ; il faut donc placer une apostrophe devant le symbole et insérer les deux à l'intérieur de parenthèses.

Il existe une troisième fonction, écrite pour convertir les valeurs RGB 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
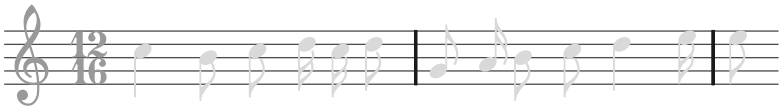
choisir la couleur rouge, la valeur serait `(rgb-color1 0 0)` ; pour le blanc, ce serait `(rgb-color 1 1 1)` :

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



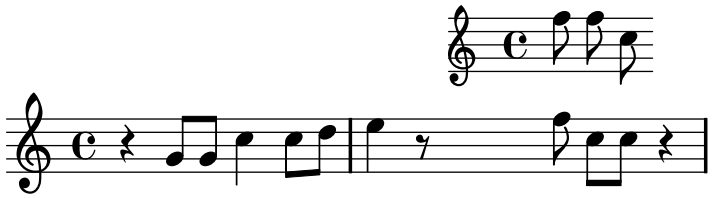
Vous remarquerez le contexte associé à chacun des objets. Une erreur sur ce point empêcherait la commande de fonctionner. Souvenez-vous que le contexte est celui dans lequel est placé le graveur approprié. Pour chaque graveur, on peut trouver son contexte par défaut en partant de l'objet lui-même, puis 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*.
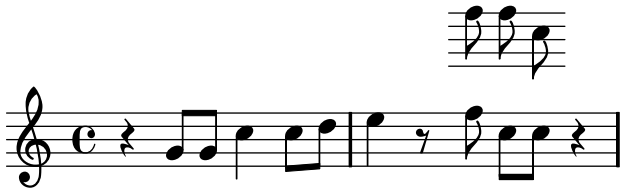
```
\new Staff ="main" {
       \relative g' {
         r4 g8 g c4 c8 d |
         e4 r8
         <<
           { f c c }
           \new Staff \with {
```

```
            alignAboveContext = #"main" }
          { f8 f c }
        >>
        r4 |
  }
}
```



Normalement, les ossia 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
    <<
      { f 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
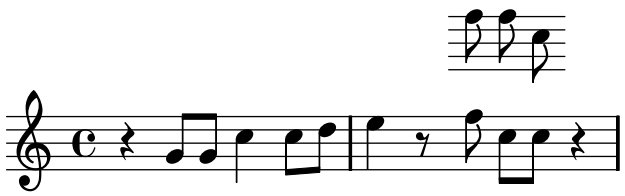
annulés ou désactivés par `\unset` ou`\revert`, les réglages reprennent les valeurs par défaut, c'est-à-dire celles qui ont été fixées dans la clause `\with`, ou, en l'absence de celle-ci, les valeurs par défaut normales.

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
    <<
      { f c c }
      \new Staff \with {
        alignAboveContext = #"main"
        % pas de clef pour cette portée
        \override Clef #'stencil = ##f
        % pas de métrique pour cette portée
        \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
```
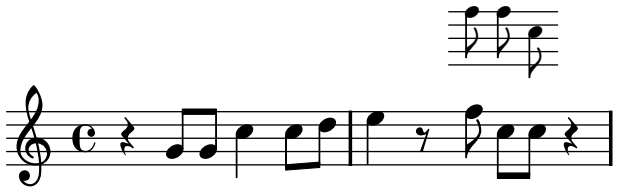
soit nous changeons la taille de toutes les polices à la fois grâce à la propriété `fontSize`, en utilisant `\set` ou en l'insérant dans une clause `\with` (mais alors sans le `\set`).

```
\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
    <<
      { f c c }
      \new Staff \with {
        alignAboveContext = #"main"
        \override Clef #'stencil = ##f
        \override TimeSignature #'stencil = ##f
        % Réduction d'environ 24 % de toutes les tailles de fonte
        fontSize = #-2
      }
        { f8 f c }
    >>
    r4 |
  }
}
```
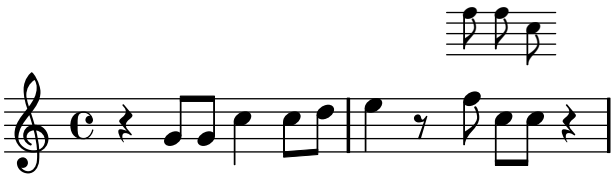


Ce n'est pas encore parfait. Les têtes de notes et les crochets sont plus petits mais, proportionnellement, les hampes sont trop longues et les lignes de portée trop espacées. Il faut donc les réduire dans les mêmes proportions que les polices de caractères. La prochaine sous-section montrera comment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

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f c c }
      \new Staff \with {
        alignAboveContext = #"main"
        \override Clef #'stencil = ##f
        \override TimeSignature #'stencil = ##f
        fontSize = #-2
        % Réduction de la taille des hampes et de l'interligne pour ne pas déborder
        \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 tailles et longueurs 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 90, 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).

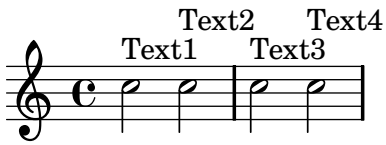
Les objets de la portée sont ceux qui sont placés sur la portée – les têtes de notes et les hampes, les altérations, etc. Leur position est généralement déterminée par la musique elle-même – ils sont placés verticalement sur des lignes spécifiques ou sont liés à d'autres objets placés de cette manière. 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 retouches 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.

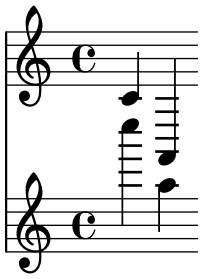
Dans l'exemple suivant, tous les *markup* ont la même priorité, dans la mesure où rien n'est indiqué explicitement. Vous remarquerez que `Text3` est également positionné près de la portée, juste en-dessous de `Text2`.

```
c2^"Text1"
c^"Text2"
c^"Text3"
c^"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' { c a, }
  }
  \new Staff {
    \relative c'''' { c a, }
  }
>>
```
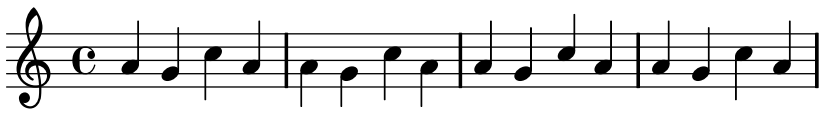


4.4.2 Objets inclus dans la portée

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 queues de notes. 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
a g c a
\override Stem #'direction = #UP
a g c a
\revert Stem #'direction
a 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
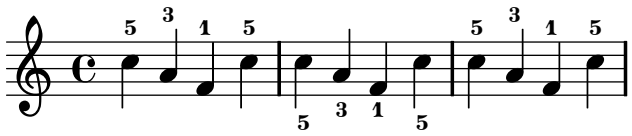
Attention : ces commandes prédéfinies **ne doivent pas** être précédées de `\once`. Pour limiter l'effet à une seule note, il faut soit utiliser la commande équivalente `\once \override`, soit utiliser la commande prédéfinie, suivie, après la note à modifier, de la commande `\xxxNeutral` correspondante.

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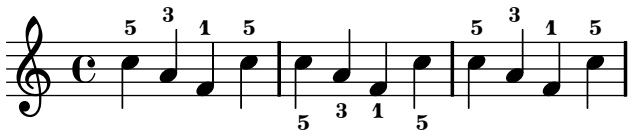
Tout d'abord, voici l'effet de `direction` sur le doigté lié à une note simple. La première mesure montre le comportement par défaut, et les deux suivantes montrent l'effet lorsqu'on indique `DOWN` et `UP` :

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



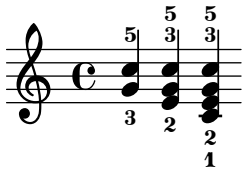
Le fait d'intervenir sur la propriété `direction` n'est sûrement pas la façon la plus simple de placer manuellement les doigtés au-dessus ou en dessous des notes ; mieux vaut utiliser `_` ou `^` devant le chiffre de doigté plutôt que `-`. Voici ce que donne l'exemple précédent avec cette méthode :

```
c-5 a-3 f-1 c'-5
c_5 a_3 f_1 c'_5
c^5 a^3 f^1 c'^5
```



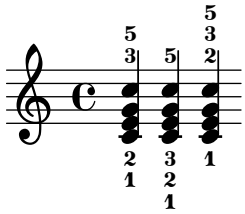
La propriété `direction` ne fonctionne pas pour les accords alors que les préfixes de direction, `_` et `^`, fonctionnent. Par défaut, le doigté est placé automatiquement à la fois au-dessus et au-dessous des notes d'un accord, comme ceci :

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



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



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`.

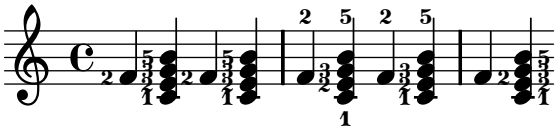
On peut attribuer à cette propriété une liste composée de une à trois valeurs. Celles-ci déterminent si l'indication de doigté doit être placée au-dessus (lorsque `up` apparaît dans la liste), au-dessous (lorsque `down` apparaît), à gauche (lorsque `left` apparaît) ou à droite (lorsque `right` apparaît). En revanche, si une valeur n'est pas sur 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 :** Ppour 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 :

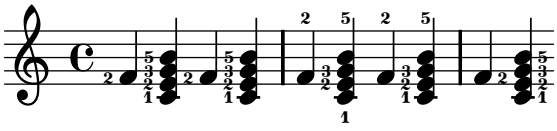
```
\set fingeringOrientations = #'(left)
<f-2>
< c-1  e-2 g-3 b-5 > 4
\set fingeringOrientations = #'(left)
<f-2>
< c-1  e-2 g-3 b-5 > 4
\set fingeringOrientations = #'(up left down)
<f-2>
< c-1  e-2 g-3 b-5 > 4
```

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



4.4.3 Objets hors de la portée

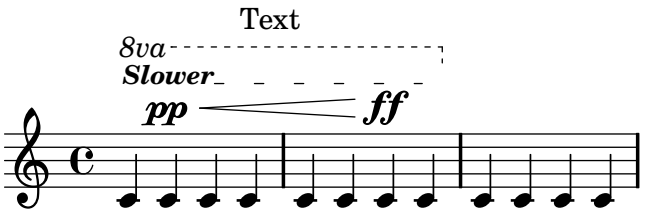
Les objets extérieurs à la portée sont placés automatiquement de façon à éviter les collisions. Les objets avec la plus petite valeur de la propriété `outside-staff-priority` sont placés au plus près de la portée, tandis que les autres sont écartés autant qu'il faut pour éviter les collisions. La `outside-staff-priority` 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`.

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.

```
% Réglage des détails du texte avec extension
\override TextSpanner #'bound-details #'left #'text
    = \markup { \small \bold Slower }
% Positionnement des nuances au-dessus de la portée
\dynamicUp
% Début du crochet d'octavation
\ottava #1
c' \startTextSpan
% Ajout d'une nuance textuelle
c\pp
% Ajout d'une nuance graphique
c\<
% Ajout d'un commentaire textuel
c^Text
c c
% Ajout d'une nuance textuelle
c\ff c \stopTextSpan
% Fin du crochet d'octavation
\ottava #0
c, 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 "Extensions de texte" dans *Manuel de notation*.

Il montre aussi comment créer des marques d'octaviation.

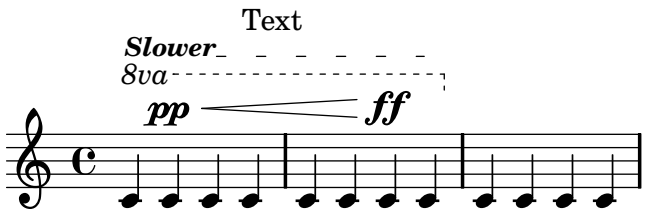
Vous aurez remarqué que les numéros de mesure, l'indication métronomique et les marques de répétition n'apparaissent pas. Par défaut, ils sont créés dans le contexte `Score` et leur `outside-staff-priority` est ignorée face aux objets qui sont créés dans le contexte `Staff`. Si vous voulez faire apparaître 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` oule `Mark_engraver`

du contexte `Score` et les placer en haut 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 que vous attendez, 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` :

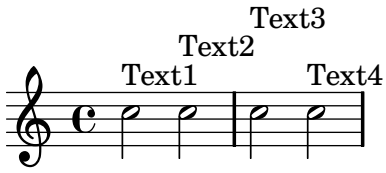
```
% Réglage des détails du texte avec extension
\override TextSpanner #'bound-details #'left #'text
    = \markup { \small \bold Slower }
% Positionnement des nuances au-dessus de la portée
\dynamicUp
%Positionnement du crochet d'octavation sous le texte avec extension
\once \override Staff.OttavaBracket #'outside-staff-priority = #340
% Début du crochet d'octavation
\ottava #1
c' \startTextSpan
% Ajout d'une nuance textuelle
c\pp
% Ajout d'une nuance graphique
c\<
% Ajout d'un commentaire textuel
c^Text
c c
% Ajout d'une nuance textuelle
c\ff c \stopTextSpan
% Fin du crochet d'octavation
\ottava #0
c, c c c
```



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 Comportement automatique, plus haut (voir Section 4.4.1 [Comportement automatique], page 105). 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"
c^"Text2"
```

```
\once \override TextScript #'outside-staff-priority = #500
c^"Text3"
c^"Text4"
```
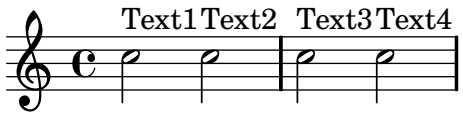


S'il est vrai que cela place « Text3 » au-dessus de « Text4 », ça le place aussi plus haut que « Text2 » tandis que « Text4 » dégringole. Ce n'est peut-être pas si bien que ça. En fait, ce que nous aimerions faire, c'est placer toutes les annotations à égale distance de la portée. Pour cela, nous avons 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 faut pour s'adapter au texte :

```
\textLengthOn  % Force les notes à s'espacer selon le texte
c2^"Text1"
c^"Text2"
c^"Text3"
c^"Text4"
```

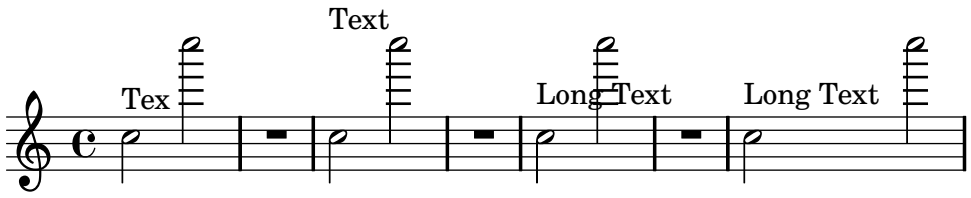


La commande qui permet de revenir au comportement par défaut est `\textLengthOff`. Rappelez-vous que `\once` ne fonctionne qu'avec `\override`, `\set`, `\revert` ou `\unset`, et donc ne peut pas être utilisé avec `\textLengthOn`.

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 notes.

```
% Ce morceau de texte est assez court pour ne pas risquer de collision
c2^"Tex"
c''2
R1
% Celui-ci est trop long pour tenir, il est déplacé vers le haut
c,,2^"Text"
c''2
R1
% Désactivation du processus d'évitement de collision
\once \override TextScript #'outside-staff-priority = ##f
c,,2^"Long Text   "
c''2
R1
```

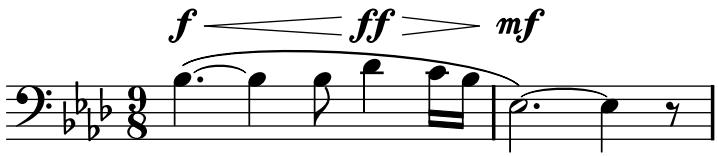
```
% Désactivation du processus d'évitement de collision
\once \override TextScript #'outside-staff-priority = ##f
\textLengthOn  % activation de textLengthOn
c,,2^"Long Text   "  % Les espaces de la fin sont pris en compte
c''2
```



Nuances

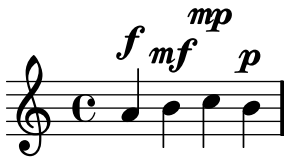
Les indications de nuances se placent normalement sous la portée mais on peut les placer au-dessus avec la commande `dynamicUp`. Elles se positionnent verticalement par rapport à la note à laquelle elles sont liées et se décalent vers le bas (ou le haut) en fonction des objets de la portée comme les 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
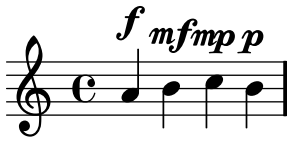
Tout d'abord, nous devons apprendre ce qui détermine la dimension des grobs. Tous les grobs portent en eux un point de référence qui est utilisé pour les positionner par rapport à leur objet parent. Ce point du grob est placé à une distance horizontale, `X-offset`, et à une distance verticale, `Y-offset`, 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'étendue verticale est fixée par une paire de nombre, `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 arriver 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)
```
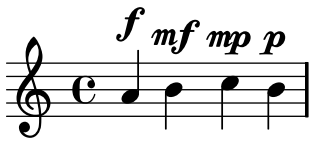
Voyons si ça marche sur notre exemple précédent :

```
\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
% Élargissement d'un espace
\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 nuances 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 106). Les queues de notes, 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 106.

   - `padding`, `left-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 `left-padding` et `right-padding`, et non plus `padding`. Ces propriétés appartiennent à l'objet `AccidentalPlacement`, qui, vous le remarquerez, prend place dans le contexte **Staff**. Dans le processus de composition, les têtes de notes sont composées en premier, puis les altérations, s'il y en a, sont ajoutées à gauche des têtes de notes suivant la propriété `right-padding` qui détermine l'espacement par rapport aux têtes de notes. 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, 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 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 une colonne de notes selon l'unité appropriée aux colonnes de notes, à savoir la largeur des têtes de notes 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 fusionnements 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 collisions 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 que ça perturbe quoi que ce soit.

`positions`
: Cette propriété est très utile pour ajuster manuellement l'inclinaison et la hauteur des barres de croches, 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 croches	`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`

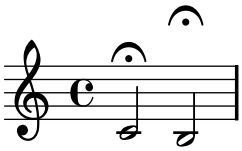
4.5.2 Correction des collisions d'objets

Voyons maintenant comment les propriétés décrites dans la section précédente peuvent nous aider à résoudre les collisions.

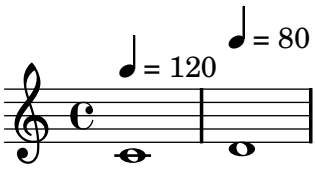
la propriété padding

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
```



```
% La commande suivante est sans résultat ; voir plus loin.
\override MetronomeMark #'padding = #3
\tempo 4=120
c1
% Celle-ci produit le résultat escompté
\override Score.MetronomeMark #'padding = #3
\tempo 4=80
d1
```
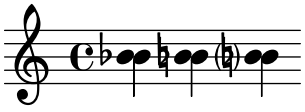


Vous remarquerez dans le second exemple à quel point il est important de préciser le nom du contexte qui contient l'objet. Puisque l'objet `MetronomeMark` appartient au contexte `Score`, le fait de modifier la propriété dans le contexte `Voice` passera inaperçu. Pour plus de détails, voir Section "Modification 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.

left-padding et 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'exemple ci-dessous montre une situation où elle est nécessaire. Imaginons que nous voulions faire apparaître un accord qui contienne à la fois un si bécarre et un si bémol. Pour qu'il n'y ait pas de doute, nous chercherions à précéder la note d'un signe bécarre et d'un signe bémol. Voici différentes tentatives pour y parvenir :
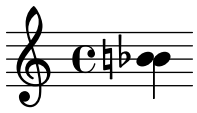
```
<b bes>
<b! bes>
<b? bes>
```



Pas une ne convient, et la seconde se solde même par une collision entre les deux signes.

Une solution pour arriver à nos fins consiste à retoucher le stencil des altérations grâce à un markup qui contient les symboles bécarre et bémol, dans l'ordre que nous souhaitons, comme ceci :

```
becarreplusbemol = \markup { \natural \flat }
\relative c'' {
  \once \override Accidental
    #'stencil = #ly:text-interface::print
  \once \override Accidental #'text = #becarreplusbemol
  \once \override Score.AccidentalPlacement #'right-padding = #1.5
  <b bes>
}
```
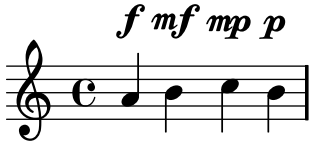


Cette méthode utilise, pour le stencil des altérations, une retouche qui ne sera par reprise par la suite. Le type de stencil est obligatoirement une procédure, qui consiste ici à imprimer le contenu de la propriété `text` de `Accidental`, déclaré comme étant un signe bécarre suivi d'un signe bémol. 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 nuances, comme dans cet exemple repris de la section précédente :

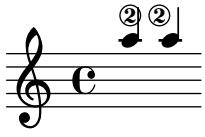
```
\dynamicUp
% Élargissement d'une unité
\override DynamicText #'extra-spacing-width = #'(-0.5 . 0.5)
% Alignement des nuances sur une ligne de base à 2 unités au-dessus de la portée
\override DynamicLineSpanner #'staff-padding = #2
a4\f b\mf c\mp b\p
```



la propriété self-alignment-X

L'exemple suivant montre comment résoudre une collision entre une indication de corde et une queue de note, en alignant le coin droit sur le point de référence de la note parente :
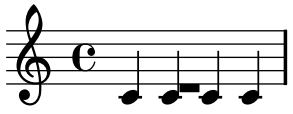
```
\voiceOne
< a \2 >
\once \override StringNumber #'self-alignment-X = #RIGHT
< a \2 >
```



la propriété staff-position

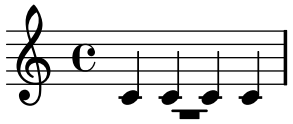
Dans une voix, un silence valant une mesure entière peut chevaucher les notes d'une autre voix. Vu que ces silences sont centrés entre les deux barres de mesure, il serait très compliqué de programmer LilyPond pour repérer ces risques de collisions, dans la mesure où, normalement, toutes les collisions entre notes ou entre notes et silences se produisent sur des notes et silences simultanés. Voici un exemple de collision de ce type :

```
<< {c 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`.

```
<<
  {c c c c}
\\
  \override MultiMeasureRest #'staff-position = #-8
  {R1}
>>
```



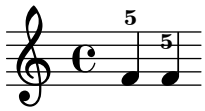
C'est mieux que d'utiliser, par exemple, `extra-offset`, car la ligne supplémentaire au-dessus du silence est insérée automatiquement.

la propriété extra-offset

La propriété `extra-offset` 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
f-5
\once \override Fingering
    #'extra-offset = #'(-0.3 . -1.8)
f-5
```



la propriété positions
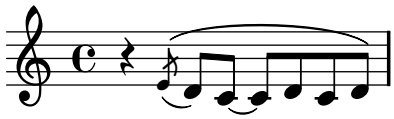
La propriété `positions` permet de contrôler manuellement la position et l'inclinaison des nolets, coulés, liaisons de phrasé et barres de croches. Voici un exemple avec une horrible liaison de phrasé – horrible pour avoir tenté de contourner la liaison de l'acciaccature.

```
r4  \acciaccatura e8\( d8 c ~c d c d\)
```
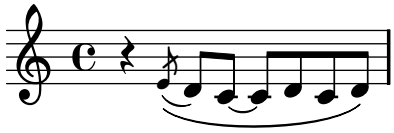


Nous pourrions tout simplement déplacer la liaison de phrasé au-dessus des notes, et ce serait la meilleure solution :

```
r4
\phrasingSlurUp
\acciaccatura e8\( d8 c ~c d c d\)
```



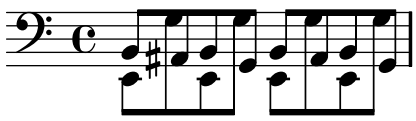
Mais si, pour une quelconque raison, cette solution n'était pas envisageable, l'autre solution consiste à déplacer légèrement vers le bas l'extrémité gauche de la liaison de phrasé, grâce à la propriété `positions`. Cela corrige en même temps la forme plutôt disgracieuse de la liaison.

```
r4
\once \override PhrasingSlur #'positions = #'(-4 . -3)
\acciaccatura
e8\( d8 c ~c d c d\)
```



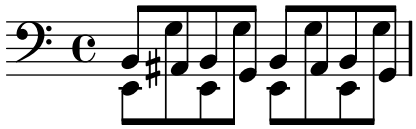
Voici un autre exemple, tiré du début de la partie de main gauche du Prélude de Chopin, Op. 28, No 2. Comme nous pouvons le constater, les barres de croches chevauchent les notes les plus hautes :

```
{
\clef "bass"
<< {b,8 ais, b, g,} \\ {e, g e, g} >>
<< {b,8 ais, b, g,} \\ {e, g e, g} >>
}
```



On peut y remédier en déplaçant manuellement vers le haut les deux extrémités des barres 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, g e, g}
  >>
  << {b,8 ais, b, g,} \\ {e, g e, g} >>
}
```
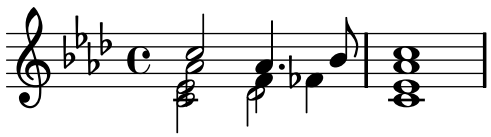


Vous remarquerez que la retouche continue à s'appliquer au second bloc de croches de la première voix mais qu'il ne s'applique à aucune barre de la deuxième voix.

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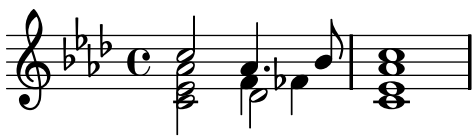
```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 } \\
    { aes2 f4 fes   } \\
    { \voiceFour
      <ees c>2
      des2
    }
  >> |
  <c ees aes c>1 |
}
```



Les deux plus basses notes du premier accord (c'est-à-dire celles de la troisième voix) ne devraient pas être décalées de la colonne 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 |
}
```

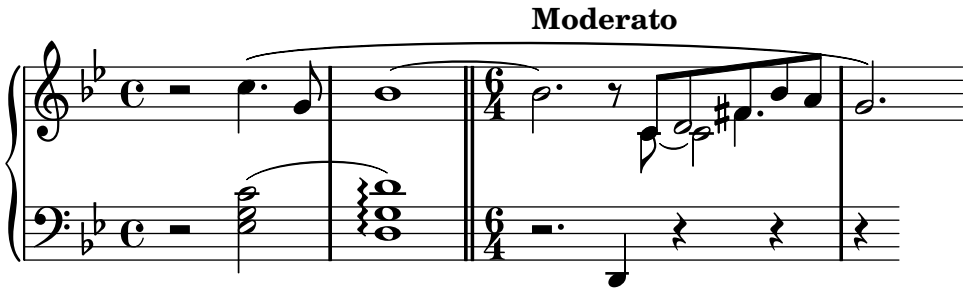


4.5.3 Exemple concret

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 nuances, de doigté ni de pédale.



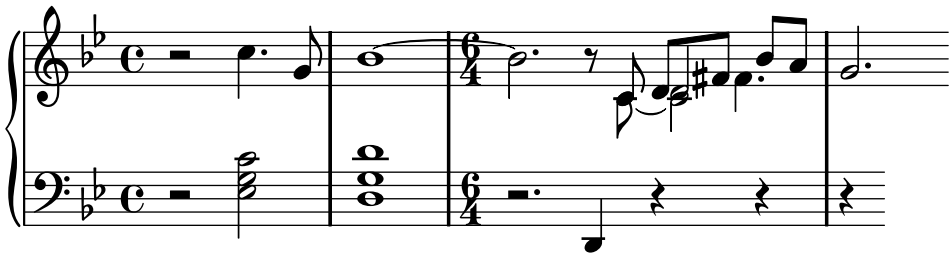
Nous constatons tout d'abord que, dans la troisième mesure, la main droite compte quatre voix. Ce sont les cinq croches avec une barre, le do avec liaison, le ré blanche qui se fond avec le ré croche, et le fa dièse noire pointée qui se fond lui aussi avec la croche de même hauteur. Tout le reste se 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 :

```
mdMusique = \relative c'' {
  r2 c4. g8 |
  bes1~ |
  \time 6/4
  bes2. r8
  % Début d'une section polyphonique de quatre voix
  <<
    {c,8 d fis bes a | }
  \\
    {c,8~ c2 | }
  \\
    {s8 d2 | }
  \\
    {s4 fis4. | }
  >>
  g2.
}

mgMusique = \relative c' {
  r2 <c g ees>2 |
  <d g, d>1 |
  r2. d,,4 r4 r |
  r4
}

\score {
```

```
  \new PianoStaff <<
    \new Staff = "MD"  <<
      \key g \minor
      \mdMusique
    >>
    \new Staff = "MG" <<
      \key g \minor
      \clef "bass"
      \mgMusique
    >>
  >>
}
```

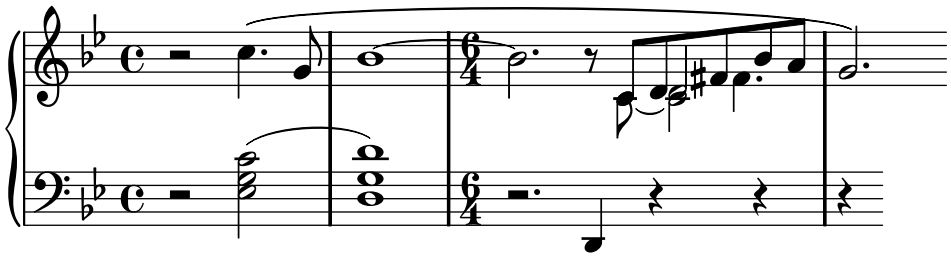


Toutes les notes sont correctes mais l'allure générale est loin d'être satisfaisante. La liaison se heurte à l'indication de mesure lors du changement de chiffrage, la barre des croches n'est pas bonne dans la troisième mesure, les notes ne sont pas fusionnées et il manque plusieurs éléments de notation. Commençons par le plus simple. Nous pouvons corriger la barre 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 :

```
mdMusique = \relative c'' {
  r2 c4.\( g8 |
  bes1~ |
  \time 6/4
  bes2. r8
  % Début d'une section polyphonique de quatre voix
  <<
    {c,8[ d fis bes a] | }
  \\
    {c,8~ c2 | }
  \\
    {s8 d2 | }
  \\
    {s4 fis4. | }
  >>
  g2.\)
}

mgMusique = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1) |
  r2. d,,4 r4 r |
  r4
}
```

```
\score {
  \new PianoStaff <<
    \new Staff = "MD"  <<
      \key g \minor
      \mdMusique
    >>
    \new Staff = "MG" <<
      \key g \minor
      \clef "bass"
      \mgMusique
    >>
  >>
}
```



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 115, 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
```

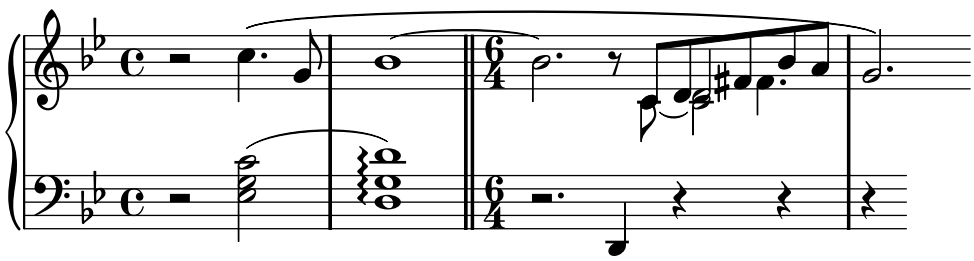
Cela s'adjoint à la deuxième mesure, pour donner :

```
mdMusique = \relative c'' {
  r2 c4.\( g8 |
  \once \override Tie #'staff-position = #3.5
  bes1~ |
  \bar "||"
  \time 6/4
  bes2. r8
  % Début d'une section polyphonique de quatre voix
  <<
    {c,8[ d fis bes a] | }
  \\
    {c,8~ c2 | }
  \\
    {s8 d2 | }
  \\
```

```
    {s4 fis4. | }
  >>
  g2.\)
}

mgMusique = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "MD"  <<
      \key g \minor
      \mdMusique
    >>
    \new Staff = "MG" <<
      \key g \minor
      \clef "bass"
      \mgMusique
    >>
  >>
}
```



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, dans Section "Résolution des collisions" dans *Manuel de notation*, aux commandes pour fusionner les notes, différentes en fonction du type de note et selon que celles-ci 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
```

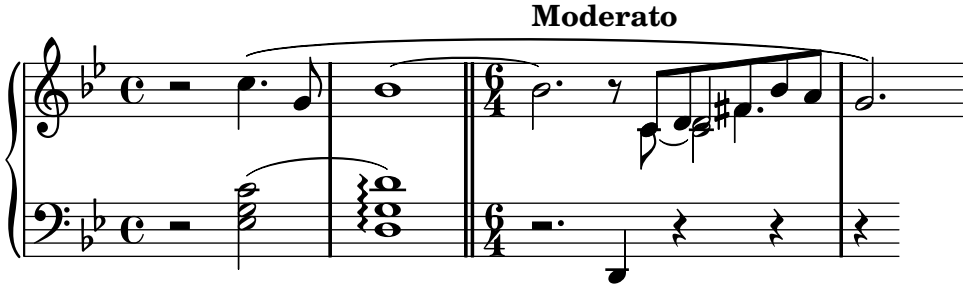
à la fin, ce qui donne :

```
mdMusique = \relative c'' {
  r2 c4.\( g8 |
```

```
  \once \override Tie #'staff-position = #3.5
  bes1~ |
  \bar "||"
  \time 6/4
  bes2.^\markup {\bold "Moderato"} r8
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  % Début d'une section polyphonique de quatre voix
  <<
    {c,8[ d fis bes a] | }
  \\
    {c,8~ c2 | }
  \\
    {s8 d2 | }
  \\
    {s4 fis4. | }
  >>
  \mergeDifferentlyHeadedOff
  \mergeDifferentlyDottedOff
  g2.\)
}

mgMusique = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "MD"  <<
      \key g \minor
      \mdMusique
    >>
    \new Staff = "MG" <<
      \key g \minor
      \clef "bass"
      \mgMusique
    >>
  >>
}
```



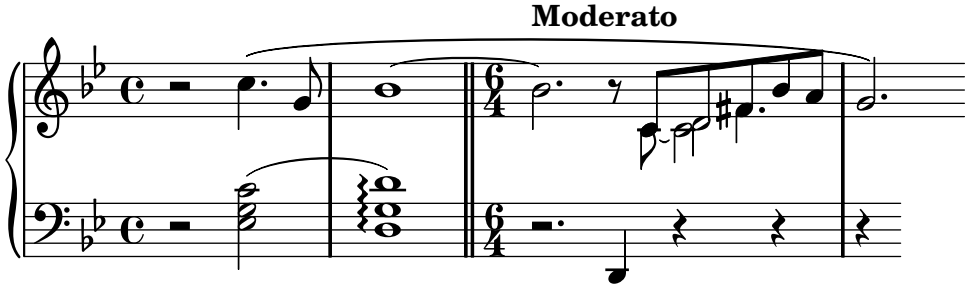
Ces retouches ont permis de fusionner les deux fa dièse mais pas les deux ré. Pourquoi ? La réponse se trouve dans la même section du Manuel de notation : les notes à fusionner doivent avoir des queues dans des directions opposées et deux notes ne peuvent pas être fusionnées s'il y a une troisième note dans la même colonne. Ici, les deux ré ont leur queue orientée vers le haut et il y a une troisième note, do. Nous savons changer l'orientation de la queue 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 :

```
mdMusique = \relative c'' {
  r2 c4.\( g8 |
  \once \override Tie #'staff-position = #3.5
  bes1~ |
  \bar "||"
  \time 6/4
  bes2.^\markup {\bold "Moderato"} r8
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  % Début d'une section polyphonique de quatre voix
  <<
    {c,8[ d fis bes a] | }
  \\
    % Déplace le do2 pour que la fusion puisse fonctionner
    {c,8~ \shiftOnn c2 | }
  \\
    % La hampe du ré2 ira vers le bas pour permettre la fusion
    {s8 \stemDown d2 | }
  \\
    {s4 fis4. | }
  >>
  \mergeDifferentlyHeadedOff
  \mergeDifferentlyDottedOff
  g2.\)
}

mgMusique = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "MD"  <<
      \key g \minor
      \mdMusique
    >>
    \new Staff = "MG" <<
```

```
      \key g \minor
      \clef "bass"
      \mgMusique
    >>
  >>
}
```



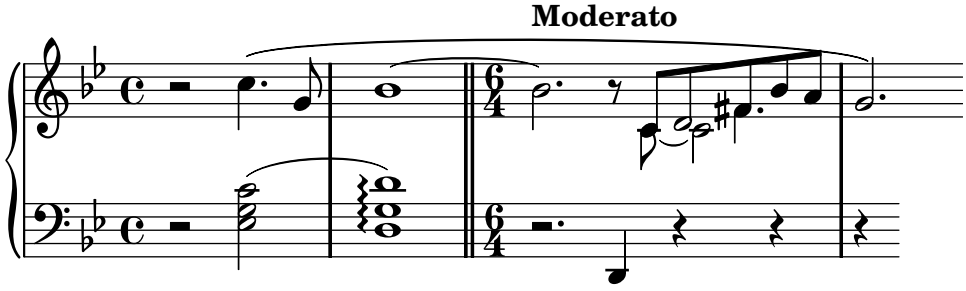
Pas loin. Il ne reste plus que deux problèmes : les ré une fois fusionnés ne devraient plus avoir de queue vers le bas, et le do serait mieux à la droite des ré. Nous savons remédier à ces deux problèmes grâce aux retouches précédentes : nous allons rendre la queue transparente et déplacer le do avec la propriété `force-hshift`. Et voici le résultat final :

```
mdMusique = \relative c'' {
  r2
  c4.\( g8 |
  \once \override Tie #'staff-position = #3.5
  bes1~ |
  \bar "||"
  \time 6/4
  bes2.^\markup {\bold "Moderato"} r8
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  <<
    {c,8[ d fis bes a] | }
  \\
    % Repositionnement du do2 à droite de la note fusionnée
    {c,8~ \once \override NoteColumn #'force-hshift = #1.0
    % Déplace le do2 pour que la fusion puisse fonctionner
    \shiftOnn c2}
  \\
    % La hampe du ré2 ira vers le bas pour permettre la fusion
    {s8 \stemDown \once \override Stem #'transparent = ##t d2}
  \\
    {s4 fis4.}
  >>
  \mergeDifferentlyHeadedOff
  \mergeDifferentlyDottedOff
  g2.\)
}

mgMusique = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
```

```
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "MD"  <<
      \key g \minor
      \mdMusique
    >>
    \new Staff = "MG" <<
      \key g \minor
      \clef "bass"
      \mgMusique
    >>
  >>
}
```
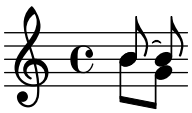


4.6 Autres retouches

4.6.1 Autres utilisations des retouches
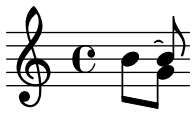
Liaison entre plusieurs voix

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
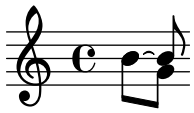


et à rendre transparente la première queue de note de cette voix ; on a alors l'impression que la liaison couvre les deux voix.

```
<<
  {
    \once \override Stem #'transparent = ##t
    b8~ b8\noBeam
  }
\\
  { b[ g8] }
>>
```



Pour être sûr que la queue 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] }
>>
```

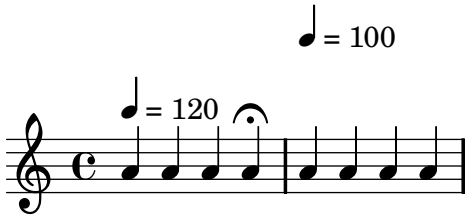


Rendu MIDI d'un point d'orgue
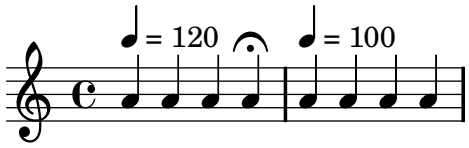
En ce qui concerne les objets extérieurs à la portée, quand on veut les faire disparaître de la partition imprimée , il est généralement préférable de modifier leur propriété `stencil` plutôt que leur propriété `transparent`. Le fait d'attribuer à la propriété `stencil` d'un objet la valeur `#f` supprimera 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'' {
    % Indication de tempo visible
    \tempo 4=120
    a4 a a
    \once \override Score.MetronomeMark #'transparent = ##t
    % Indication de tempo invisible ; utilisée pour le MIDI
    \tempo 4=80
    a\fermata
    % Nouveau tempo pour la section suivante
    \tempo 4=100
    a a a a
  }
  \layout { }
  \midi { }
}
```



```
\score {
  \relative c'' {
    % Indication de tempo visible
    \tempo 4=120
    a4 a a
    \once \override Score.MetronomeMark #'stencil = ##f
    % Indication de tempo invisible ; utilisée pour le MIDI
    \tempo 4=80
    a\fermata
    % Nouveau tempo pour la section suivante
    \tempo 4=100
    a a a a
  }
  \layout { }
  \midi { }
}
```



Les deux méthodes permettent l'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.

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 italiques. 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

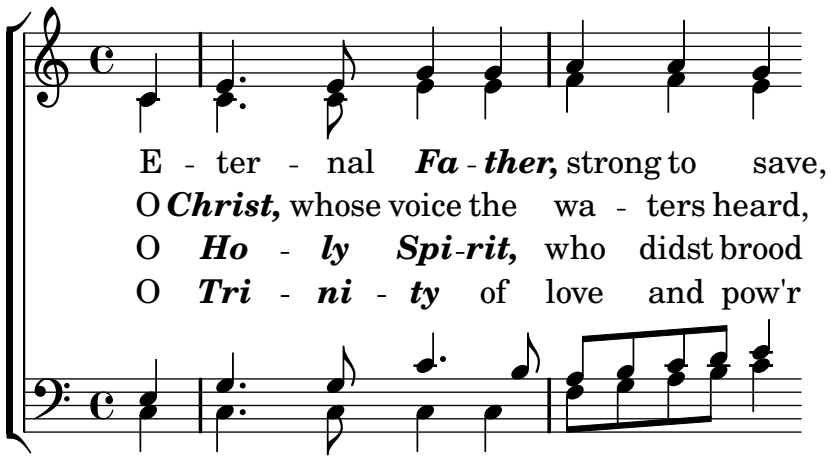
\revert Lyrics . LyricText #'font-shape
\revert Lyrics . LyricText #'font-series
```

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 suit – quoique on choisirait sans doute pour les variables des noms plus courts pour simplifier la frappe :

```
emphase = {
  \override Lyrics . LyricText #'font-shape = #'italic
  \override Lyrics . LyricText #'font-series = #'bold
}
normal = {
  \revert Lyrics . LyricText #'font-shape
  \revert Lyrics . LyricText #'font-series
}
```

```
global = { \time 4/4 \partial 4 \key c \major}
sopranoMusique   = \relative c' { c4 | e4. e8 g4  g  | a a g }
AltoMusique  = \relative c' { c4 | c4. c8 e4  e  | f f e }
TenorMusique = \relative c  { e4 | g4. g8 c4. b8 | a8 b c d e4 }
BasseMusique  = \relative c  { c4 | c4. c8 c4  c  | f8 g a b c4 }
CoupletUn   = \lyrics { E -- | ter -- nal \emphase Fa -- ther, \normal | strong to save
CoupletDeux   = \lyricmode { O | \emphase Christ, \normal whose voice the | wa -- ters
CoupletTrois = \lyricmode { O | \emphase Ho -- ly Spi -- rit, \normal | who didst brooo
CoupletQuatre  = \lyricmode { O | \emphase Tri -- ni -- ty \normal of | love and pow'r

\score {
  \new ChoirStaff <<
    \new Staff <<
      \clef "treble"
      \new Voice = "Soprano"  { \voiceOne \global \sopranoMusique }
      \new Voice = "Alto" { \voiceTwo \AltoMusique }
      \new Lyrics \lyricsto "Soprano" { \CoupletUn   }
      \new Lyrics \lyricsto "Soprano" { \CoupletDeux   }
      \new Lyrics \lyricsto "Soprano" { \CoupletTrois }
      \new Lyrics \lyricsto "Soprano" { \CoupletQuatre  }
    >>
    \new Staff <<
      \clef "bass"
      \new Voice = "Tenor" { \voiceOne \TenorMusique }
      \new Voice = "Basse"  { \voiceTwo \BasseMusique  }
    >>
  >>
}
```



4.6.3 Autres sources de documentation

La Référence du programme 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 sous 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 Annexe B [Tutoriel Scheme], page 149).

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

4.6.4 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 serrer 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 bloc Score `\with`, plutôt qu’à l’intérieur du fragment musical, comme ceci :

```
\new Score \with {
  % 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
} {
   ..
}
```

4.6.5 Retouches avancées avec Scheme

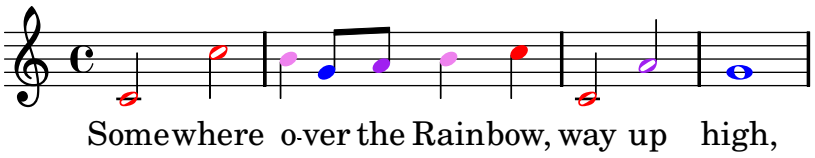
Nous avons vu à quel point le résultat obtenu avec LilyPond peut être largement personnalisé à l’aide de commandes comme `\override` et `\tweak`. Et pourtant l’utilisation de Scheme ouvre des possibilités encore plus grandes. Le code écrit dans le langage de programmation Scheme peut être intégré directement 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 Annexe B [Tutoriel Scheme], page 149.

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' {
  % Arranger pour obtenir une couleur par la procédure color-notehead
  \override NoteHead #'color = #color-notehead
  c2 c' |
  b4 g8 a b4 c |
  c,2 a' |
  g1 |
}
\addlyrics {
  Some -- where o -- ver the Rain -- bow, way up high,
}
```



Vous trouverez dans Section B.1 [Scheme et les retouches], page 149 d'autres exemples d'utilisation de ces interfaces programmables.

5 Travail sur des projets LilyPond

Cette section explique comment résoudre ou éviter certains problèmes courants. Si vous avez de l'expérience en programmation, beaucoup de ces astuces peuvent vous paraître évidentes, mais vous ne perdrez tout de même pas votre temps à lire ce chapitre.

5.1 Suggestions pour la saisie de fichiers LilyPond

Maintenant vous êtes prêt à travailler sur de plus gros fichiers LilyPond — des pièces entières, et plus seulement les petits exemples du tutoriel. Mais comment devriez-vous vous y prendre ?

Tant que LilyPond parvient à comprendre vos fichiers et produit le résultat que vous souhaitez, peu importe la manière dont le code est organisé. Néanmoins, quelques critères doivent être pris en compte lorsque l'on écrit un fichier LilyPond.

- Si vous faites une erreur, la structure même du fichier LilyPond peut permettre de la localiser plus ou moins facilement.
- Et si vous souhaitez partager vos fichiers avec quelqu'un d'autre, ou si vous souhaitez modifier vos propres fichiers dans quelques années ? Si certains fichiers LilyPond sont compréhensibles au premier coup d'oeil, d'autres vous feront vous arracher les cheveux pendant une heure.
- Et si vous souhaitez mettre à jour votre fichier pour l'utiliser avec une version plus récente de LilyPond ? La syntaxe du langage d'entrée change parfois lorsque LilyPond s'améliore. La plupart des changements peuvent être appliqués automatiquement avec `convert-ly`, mais quelques-uns peuvent requérir une intervention manuelle. Vos fichiers LilyPond peuvent être structurés de manière à faciliter leur mise à jour.

5.1.1 Suggestions générales

Voici quelques conseils qui peuvent vous éviter certains problèmes ou en résoudre d'autres.

- **Ajoutez le numéro de version dans chaque fichier**. Notez que chaque fichier modèle contient une ligne `\version "2.11.32"`. Nous vous conseillons fortement d'inclure cette ligne, même pour de petits fichiers. Par expérience, il est très difficile de se rappeler quelle version de LilyPond on utilisait quelques années auparavant. L'utilitaire `convert-ly` demande que vous spécifiiez la version de LilyPond vous utilisiez alors.
- **Ajoutez des contrôles**: Section "Vérifications d'octave" dans *Manuel de notation*, et Section "Vérification des limites et numéros de mesure" dans *Manuel de notation*. Si vous avez ajouté des contrôles de loin en loin, et que vous faites une erreur, vous pourrez la retrouver plus rapidement. « De loin en loin », qu'est-ce à dire ? Cela dépend de la complexité de la musique. Pour de la musique très simple, peut-être une ou deux fois. Pour de la musique très complexe, peut-être à chaque mesure.
- **Une mesure par ligne de texte**. Si la musique en elle-même ou le résultat que vous désirez contient quelque chose de compliqué, il est souvent bon de n'écrire qu'une seule mesure par ligne. Économiser de la place en tassant huit mesures par ligne, ça ne vaut pas vraiment le coup si l'on doît corriger vos fichiers.
- **Ajoutez des commentaires**. Utilisez soit des numéros de mesure (assez souvent), soit des références au contenu musical — « second thème des violons », « quatrième variation », etc. Vous pouvez ne pas avoir besoin des commentaires lorsque vous écrivez une pièce pour la première fois, mais si vous souhaitez y revenir deux ou trois ans plus tard pour changer quelque chose, ou si vous donnez le fichier source à un ami, ce sera beaucoup plus difficile de déterminer vos intentions ou la manière dont votre fichier est structuré si vous n'y avez pas adjoint de commentaires.
- **Indentez les accolades**. Beaucoup de problèmes viennent d'un défaut de parité entre `{` et `}`.

- **Séparez les affinages de mise en forme** de la musique elle-même. Voyez Section 5.1.4 [Économie de saisie grâce aux identificateurs et fonctions], page 138 et Section 5.1.5 [Feuilles de style], page 140.

5.1.2 Gravure de musique existante

Si vous saisissez de la musique à partir d'une partition existante, c'est-à-dire de la musique déjà écrite,

- n'entrez qu'un seul système de la partition originale à la fois — mais toujours une seule mesure par ligne de texte —, et vérifiez chaque système lorsqu'il est terminé. Vous pouvez utiliser la commande `showLastLength` pour accélérer la compilation — voir Section "Ignorer des passages de la partition" dans *Manuel de notation* ;
- définissez `mBreak = {\break }` et insérez `\mBreak` dans le fichier d'entrée pour obtenir des sauts de ligne identiques à la partition originale. Cela facilite la comparaison entre la partition originale et la partition de LilyPond. Lorsque vous avez fini de relire votre musique, vous pouvez définir `mBreak = { }` pour enlever tous ces sauts de ligne, et laisser LilyPond placer les sauts de ligne selon son propre algorithme.

5.1.3 Projets d'envergure

Lorsque l'on travaille sur un gros projet, il devient vital de structurer clairement ses fichiers LilyPond.

- **Utilisez un identificateur pour chaque voix**, avec un minimum de structure dans la définition. La structure de la section `\score` est la plus susceptible de changer, notamment dans une nouvelle version de LilyPond, alors que la définition du `violon` l'est beaucoup moins.

```
violin = \relative c'' {
g4 c'8. e16
}
...
\score {
 \new GrandStaff {
   \new Staff {
     \violin
   }
 }
}
```

- **Séparez les retouches** des définitions de musique. Ce conseil a été vu dans Section 5.1.1 [Suggestions générales], page 137, mais pour les projets d'importance c'est absolument vital. Nous pouvons avoir besoin de changer la définition de `fthenp`, mais dans ce cas nous n'aurons besoin de le faire qu'une seule fois, et nous pourrons encore éviter de modifier quoi que ce soit à l'intérieur de la définition du `violon`.
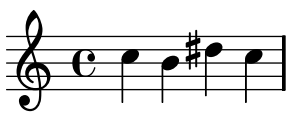
```
fthenp = _\markup{
 \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
g4\fthenp c'8. e16
}
```

5.1.4 Économie de saisie grâce aux identificateurs et fonctions

Jusqu'à maintenant, vous avez vu ce type de code :
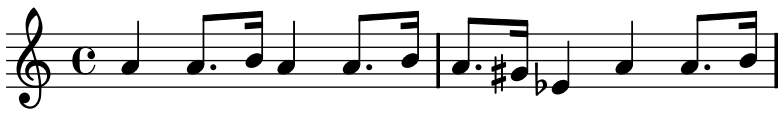
```
corNotes = \relative c'' { c4 b dis c }
\score {
  {
```

```
    \corNotes
  }
}
```
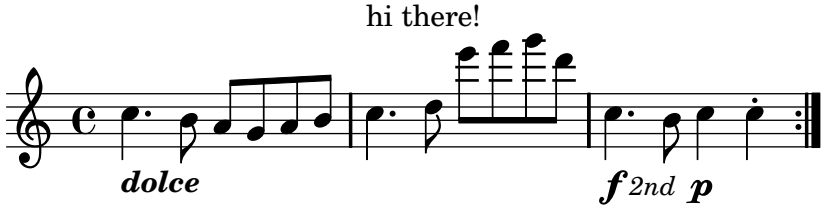


Vous comprendrez combien cela peut être utile pour écrire de la musique minimaliste :

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



Cependant, 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 }
decallageTexte = { \once \override TextScript #'padding = #5.0 }
fpuisp=_\markup{ \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violon = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \decallageTexte
    c4.^"hi there!" d8 e' f g d |
    c,4.\fpuisp b8 c4 c-. |
  }
}
\score {
  {
    \violon
  }
\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.
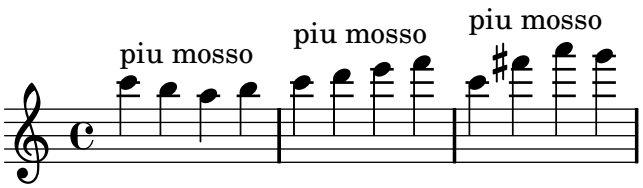
```
violin = \relative c'' {
  \repeat volta 2 {
    c4._\markup{ \italic \bold dolce } b8 a8 g a b |
    \once \override TextScript #'padding = #5.0
    c4.^"hi there!" d8 e' f g d |
    c,4.\markup{ \dynamic f \italic \small { 2nd }
      \hspace #0.1 \dynamic p } b8 c4 c-. |
  }
}
```

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.

```
decallageTexte =
#(define-music-function (parser location padding) (number?)
  #{
    \once \override TextScript #'padding = #$padding
  #})

\relative c''' {
  c4^"piu mosso" b a b
  \decallageTexte #1.8
  c4^"piu mosso" d e f
  \decallageTexte #2.6
  c4^"piu mosso" fis a g
}
```



Utiliser les identificateurs est aussi un bon moyen pour vous épargner du travail si la syntaxe de LilyPond change un jour — voir Section 5.2.1 [Mise à jour d'anciens fichiers], page 144. Si vous avez une seule définition, par exemple `\dolce`, pour tous vos fichiers (voir Section 5.1.5 [Feuilles de style], page 140), 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`.

5.1.5 Feuilles de style
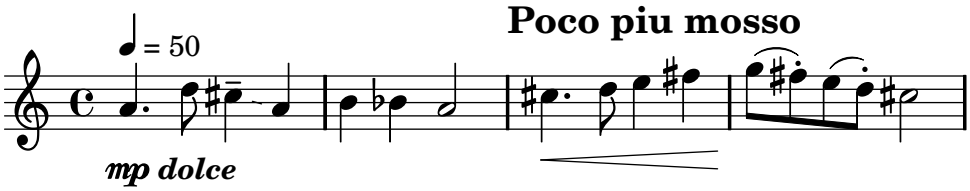
La sortie que produit LilyPond peut être largement modifiée — voir Chapitre 4 [Retouche de partition], page 85 pour plus de détails. Mais que faire si vous avez beaucoup de fichiers auxquels vous souhaitez appliquer vos retouches ? Ou si vous souhaitez simplement séparer les retouches de la musique elle-même ? 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 4.6.5 [Retouches avancées avec Scheme], page 135.

```
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))
tempoMarque = #(define-music-function (parser location markp) (string?)
```

```
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \tempoMarque "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```
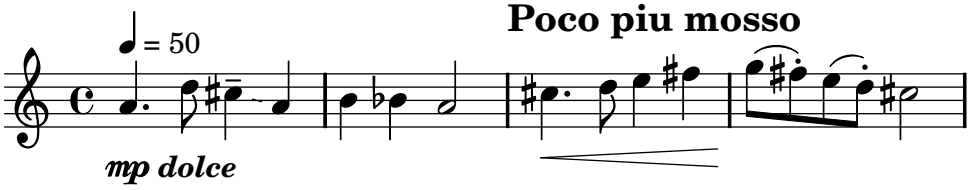


Il y a quelques problèmes de chevauchement ; nous allons arranger cela en utilisant les techniques de Section 4.5.1 [Déplacement d'objets], page 115. On peut aussi faire quelque chose pour les définitions de `mpdolce` et `tempoMark`. Elles produisent le résultat que nous désirons, mais nous pourrions aussi vouloir les utiliser dans une autre pièce. Il suffirait de les copier et les 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.ly"
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})
```

Maintenant, modifions notre musique (enregistrez ce fichier sous '`"musique.ly"`').

```
\include "definitions.ly"

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \once \override Score.RehearsalMark #'padding = #2.0
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```
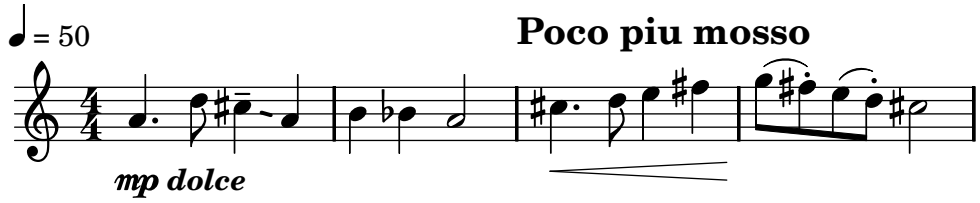


C'est mieux, mais effectuons encore quelques retouches. Le glissando est peu visible, c'est pourquoi nous allons l'épaissir et le rapprocher des têtes de notes. Déplaçons l'indication métronomique au-dessus de la clef, au lieu de la laisser au-dessus de la première note. Et pour finir, mon professeur 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.ly`' par ceci :

```
%%%  definitions.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})
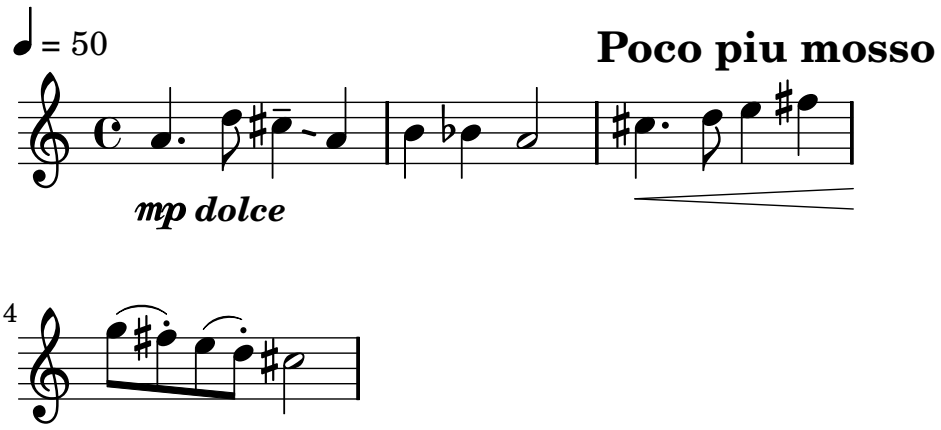
\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



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.ly`' dans le fichier '`publication-web.ly`', 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.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})
```

```
#(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.ly"` par `\include "publication-web.ly"` dans notre fichier de musique.

Il est possible, bien sûr, de rendre cela encore plus pratique. Nous pourrions créer un fichier '`definitions.ly`' qui ne contiendrait que les définitions de `mpdolce` et de `tempoMark`, un fichier '`publication-web.ly`' qui ne contiendrait que la section `layout` décrite ci-dessus et un fichier '`universite.ly`' 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.ly"

%%%  Décommentez seulement une de ces deux lignes !
\include "publication-web.ly"
%\include "universite.ly"
```

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.ly"` qui contient :

```
%%%   global.ly
\version "2.12.1"
#(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"
```

5.2 Quand ça ne fonctionne pas

5.2.1 Mise à jour d'anciens fichiers

La syntaxe de LilyPond change de temps en temps. Ces changements de syntaxe du langage d'entrée accompagnent les améliorations du logiciel. Ces changements sont parfois destinés à rendre les fichiers plus faciles à lire et à écrire, ou permettent d'intégrer de nouvelles fonctionnalités.

LilyPond est fourni avec un utilitaire qui facilite cette mise à jour : `convert-ly`. Pour savoir comment utiliser ce programme, voir Section "Mise à jour des fichiers avec convert-ly" dans *Manuel d'utilisation du programme*.

Malheureusement, `convert-ly` ne peut pas réaliser toutes les modifications. Il s'occupe des changements qui ne requièrent qu'une simple substitution de texte — comme `raggedright` devenant `ragged-right` —, les autres étant trop compliqués à effectuer. Les changements de syntaxe qui ne sont pas gérés par `convert-ly` sont énumérés dans Section "Mise à jour des fichiers avec convert-ly" dans *Manuel d'utilisation du programme*.

Par exemple, dans les versions 2.4 et antérieures de LilyPond, les accents et les lettres non anglaises étaient entrées en utilisant LaTeX — par exemple, '`No\"el`'. À partir de la version 2.6, le caratère 'ë' doit être entré directement dans le fichier LilyPond comme caractère UTF-8. `convert-ly` ne peut pas changer tous les caractères LaTeX en caractères UTF-8 ; vous devez mettre à jour vos vieux fichiers LilyPond manuellement.

5.2.2 Résolution de problèmes – tout remettre à plat

Tôt ou tard, vous écrirez un fichier que LilyPond ne peut pas compiler. Les messages que LilyPond affiche peuvent vous aider à trouver l'erreur, mais dans beaucoup de cas vous aurez besoin de faire quelques recherches pour déterminer la source du problème.

Pour ce faire, les outils les plus puissants sont le commentaire de fin de ligne, indiqué par `%`, et le commentaire multilignes (ou bloc de commentaire), indiqué par `%{ ... %}`. Si vous ne pouvez localiser le problème, commencez par mettre en commentaire de grandes parties de votre fichier d'entrée. Après avoir mis en commentaire une section, essayez de compiler à nouveau. Si cela fonctionne, c'est que le problème se situe dans cette partie du fichier. Si cela ne fonctionne pas, continuez à mettre en commentaire d'autres sections, jusqu'à ce que vous ayez quelque chose qui compile.

Dans un cas extrême, vous pourriez en arriver à

```
\score {
  <<
    % \melodie
    % \harmonie
    % \basse
  >>
  \layout{}
}
```

c'est-à-dire un fichier sans aucune musique.

Si cela arrive, ne vous découragez pas. Décommentez un peu, la partie de basse par exemple, et voyez si ça fonctionne. Si ce n'est pas le cas, placez en commentaire toute la partie de basse, mais laissez `\basse` décommenté dans le bloc `\score`.

```
basse = \relative c' {
%{
  c4 c c c
  d d d d
```

```
%}
}
```

Maintenant commencez à décommenter petit à petit le partie de `basse` jusqu'à ce que vous localisiez la ligne qui pose problème.

Une autre technique de déboguage très utile est la construction de Section 5.2.3 [Exemples minimalistes], page 145.

5.2.3 Exemples minimalistes

Un exemple minimal est un exemple de code aussi court que possible. De tels exemples sont bien plus compréhensibles que des exemples longs. Les exemples minimaux sont utilisés pour

- les rapports de bogue,
- les demandes d'aide sur les listes de diffusion,
- un ajout à LilyPond Snippet Repository.

Pour construire un exemple minimal, la règle est très simple : enlevez tout ce qui n'est pas nécessaire. Il est préférable de commenter les lignes non nécessaires plutôt que de les effacer : ainsi, si vous vous apercevez que certaines étaient *réellement* nécessaires, vous pouvez les décommenter au lieu de les resaisir.

Il y a deux exceptions à cette règle du strict nécessaire :

- incluez le numéro de `\version` en début de fichier
- si possible, utilisez `\paper{ ragged-right=##t }` au début de votre exemple.

Tout l'intérêt d'un exemple minimal réside dans sa facilité de lecture :

- évitez d'utiliser des notes, armures ou métriques compliquées, à moins que vous ne vouliez montrer quelque chose en rapport avec celles-ci,
- n'utilisez pas de commandes `\override` sauf si elles font l'intérêt de l'exemple.

5.3 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 :
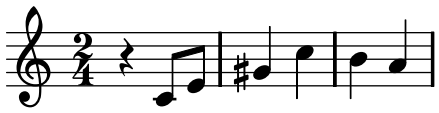
```
\include "musique-Cor.ly"
\header {
  instrument = "Cor en Fa"
}

{
 \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 :
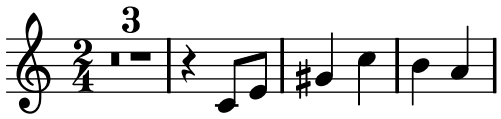


Dans les pièces d’ensemble, il arrive souvent qu’une voix ne joue pas pendant plusieurs mesures. Un silence spécial, appelé silence multi-mesures, l’indique alors. On l’obtient par un ‘`R`’ majuscule, suivi d’une durée : `1` pour une pause, `2` pour une demi-pause, etc. Cette durée peut être multipliée 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 multi-mesures 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 multi-mesures et cette option, on obtient le résultat suivant :

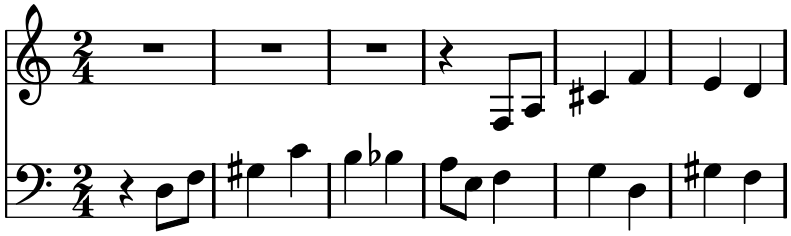


Le conducteur rassemble toute la musique. Si l’on suppose que l’autre voix de notre duo se trouve dans le fichier ‘`musique-Basson.ly`’ en tant que variable `notesBasson`, on établira un conducteur avec

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

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

ce qui équivaut à



Des informations plus détaillées sur la mise en place de conducteurs et de parties séparées se trouvent dans le manuel : voir Section “Écriture de parties séparées” dans *Manuel de notation*.

Les variables (‘propriétés’) réglables sont abordées en détail dans Section “La commande de fixation (set)” dans *Manuel de notation*.

Annexe A Modèles

Cette annexe du manuel 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

A.1.2 Notes et paroles

A.1.3 Notes et accords

A.1.4 Notes, paroles et accords

A.2 Modèles pour claviers

A.2.1 Piano seul

A.2.2 Chant et accompagnement

A.2.3 Piano et paroles entre les portées

A.2.4 Piano et nuances entre les portées

A.3 Quatuor à cordes

A.3.1 Quatuor à cordes

A.3.2 Parties pour quatuor à cordes

A.4 Ensemble vocal

A.4.1 Partition pour chœur à quatre voix mixtes

A.4.2 Partition pour chœur SATB avec réduction pour piano

A.4.3 Partition pour chœur SATB avec alignement des contextes

A.5 Exemples de notation ancienne

A.5.1 Transcription de musique mensurale

A.5.2 Transcription du grégorien

A.6 Symboles de jazz

A.7 Squelettes pour lilypond-book

A.7.1 LaTeX

A.7.2 Texinfo

A.7.3 xelatex

Annexe B Tutoriel Scheme

B.1 Scheme et les retouches

Annexe C Licence GNU de documentation libre

Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA

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 written 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 that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The 'Document', below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as 'you'.

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. (For example, 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.

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 'Transparent' copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and 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 has been designed to thwart or discourage subsequent modification by readers is not Transparent. 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 designed for human modification. Opaque formats include PostScript, PDF, 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 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.

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 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 publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. 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 less than five).

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', and 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. In any section entitled 'Acknowledgments' or 'Dedications', preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments 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 as 'Endorsements' or to conflict in title with any Invariant Section.

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.

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 'Acknowledgments', 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, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an 'aggregate', and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they 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 quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around 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 provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

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.

SUPPLÉMENT : comment utiliser cette licence pour vos 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 1.1
  or any later version published by the Free Software Foundation;
  with the Invariant Sections being list their titles, with the
  Front-Cover Texts being list, and with the Back-Cover Texts being list.
  A copy of the license is included in the section entitled ‘GNU
Free Documentation License’
.
```

If you have no Invariant Sections, write ‘with no Invariant Sections’ instead of saying which ones are invariant. If you have no Front-Cover Texts, write ‘no Front-Cover Texts’ instead of ‘Front-Cover Texts being *list*’; likewise for Back-Cover Texts.

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 D Index de LilyPond

!

! 23

%

% 18
%{ ... %} 18

,

' 14

(

(...) 21

,

, ... 14

.

... . 18

<

< 23, 30
< ... > 30
<< 27, 30
<< ... >> 27
<< ... \\ ... >> 30
<< \\ >> 49

>

> 23, 30
>> 27, 30

[

[..... 24
[...] 24

]

] 24

^

^ 23

_

_ 23

\

\! 23
\(... \) 21
\< 23
\> 23
\\ 30, 49
\acciaccatura 25
\addlyrics 31
\addlyrics – exemple 90
\addlyrics, exemple 94
\appoggiatura 25
\autoBeamOff 24, 57
\autoBeamOn 24
\book 41, 61
\clef 17
\consists 71
\f 23
\ff 23
\grace 25
\header 38, 42
\key 20
\layout 42, 72
\lyricmode 57
\lyricsto 56
\major 20
\markup 24
\mf 23
\midi 42
\minor 20
\mp 23
\new 28, 64
\new ChoirStaff 57
\new Lyrics 56
\new Staff 28
\new Voice 53
\once 87, 92
\oneVoice 53
\override 86
\overrideProperty 88
\p 23
\partial 25
\pp 23
\relative 14
\remove 71
\revert 87, 92
\score 41, 43
\set 67
\set, exemple d'utilisation 108
\shiftOff 56
\shiftOn 56
\shiftOnn 56
\shiftOnnn 56
\startTextSpan 110
\stopTextSpan 110
\textLengthOff 112
\textLengthOn 112
\time 17
\times 25
\tweak 88
\tweak, exemple 88, 89
\unset 67
\version 38
\voiceFour 53
\voiceFourStyle 51

\voiceNeutralStyle 51
\voiceOne 53
\voiceOneStyle 51
\voiceThree 53
\voiceThreeStyle 51
\voiceTwo 53
\voiceTwoStyle 51
\with 70
\with, exemple 100, 101, 102, 103, 104

~

~ 21

A

absolu, mode 38
accents 22
acciaccatura 25
acciaccature 25
Accidental, example of overriding 118
AccidentalPlacement, example of overriding 118
accolade 29
accolades 18
accord 30
accords et durée 30
accords, notes simultanées 30
addlyrics 31
Aide-mémoire 10
ajout de texte 24
Ajout de texte 24
alignAboveContext, exemple de propriété .. 100, 101, 102, 103, 104
alignement des paroles 32
aligning objects on a baseline 119
altération 14, 20
Altérations 21
altérations à l'armure 20
Altérations accidentelles automatiques 21
altérations et mode relatif 14
ambitus 72
ambitus, graveur 71
anacrouse 25
annexes 9
apostrophe 14
appoggiature 25
armure 20
Armure 21
armure, altérations à l' 20
armure, définition de l' 20
articulation 22
articulation, liaisons d' 21
Articulations et ornements 24
assignation de variables 36
autoBeamOff 24
autoBeamOn 24
automatisée, gravure 3

B

b 20
BarLine, example of overriding 97
BarLine, exemple de dérogation 97, 99, 100
barre de ligature 16, 24
Barres de ligature automatiques 25
Barres de ligature manuelles 25
bb 20
Beam, example of overriding 121
beams, controlling manually 120
bécarre 20
bémol 20
bémol, double 20
Bibliographie 10
bien lire le manuel 19
blanche 16
Blanchiment des objets 99
bloc d'en-tête 38
bloc de commentaire 18
book, exemple d'utilisation 61
book, livre, ouvrage 41
bound-details, exemple de propriété 110, 111
break-visibility property, example 97
break-visibility, propriété 97

C

calques (layers) 48
caractère souligné (paroles) 32
caractères autorisés dans les variables 36
casse, prise en compte de 18
casse, prise en compte de la 12
center 106
chansons 31
chœur, système pour 29
ChoirStaff 29
ChordNames 28
clavier, portée pour 29
clé 17
clef 17
Clef, exemple de dérogation .. 100, 101, 102, 103, 104
Clefs 18
cliquables, exemples 19
color property, setting to Scheme procedure 135
color, exemple de propriété 99, 100
color, propriété 99
combinaison d'expressions en parallèle 27
commentaire 18
commentaire de fin de ligne 18
commentaire-bloc 18
compilation 12
compulser le manuel 19
conseils de construction des fichiers 19
construction des fichiers, conseils 19
contexte 28
contexte de notation 28
contexte de voix 48
contexte, détermination du 92
contexte, identification correcte du 92
contexte, noms de 65
contexte, propriétés 67
contexte, spécification en mode lyrique 95
contextes 6
Contextes, création de 64
contextes, les différents 63
Contexts 6, 68, 71
controlling tuplets, slurs, phrasing slurs, and beams manually 120
conventions de nommage des objets 86

conventions de nommage des propriétés ... 86
Conversion à partir d'autres formats ... 10
convert-ly ... 38
couleur X11 ... 99
couleur, exemple de propriété ... 88, 89
couleur,exemple de propriété ... 87
couleurs rgb ... 99
couplet et refrain ... 59
Création de fichiers MIDI ... 42
Création de titres ... 38
crescendo ... 23
crochet de nolet ... 89
crochets, imbrication ... 47

D

d ... 20
dd ... 20
décalage, commandes ... 56
decrescendo ... 23
défaut, retour aux propriétés par ... 92
dérogation pour une seule fois ... 92
dérogation, exemple ... 90
dièse ... 20
dièse, double ... 20
dimensionnement des grobs ... 114
direction, exemple de propriété ... 89, 106, 107
distance ... 103
do central ... 14
documentation du fonctionnement interne ... 10
doigté ... 23
doigtés ... 23
Doigtés ... 24
doigtés, accords ... 107
doigtés, exemple ... 107, 108
doigtés, exemple de dérogation ... 107
doigtés, positionnement ... 107
double bémol ... 20
double dièse ... 20
down ... 106
durée ... 16
durée, liaisons de ... 21
durées ... 16
DynamicLineSpanner, example of overriding ... 119
DynamicText, example of overriding ... 114, 119

E

écartement des lignes, modification ... 103
Écriture de parties séparées ... 146
Écriture des hauteurs de note ... 18
Écriture des silences ... 18
Écriture du rythme ... 18
éditeurs de texte ... 12
empilement de notes ... 56
en-tête ... 42
en-têtes ... 38
engravers ... 66
Engravers and Performers ... 67
Environnement de travail ... 10
épaisseur ... 103
épaisseur des caractères ... 2
Épaisseur, exemple de propriété ... 92, 93
équilibre ... 2
es ... 20
eses ... 20
espacement optique ... 2
espacement régulier ... 3
espaces multiples, insensibilité ... 18
étendre lilypond ... 10
étiquette ... 24
examples simples ... 8
Exécution de LilyPond ... 10
exemple, premier ... 12
exemple, SATB ... 77
exemples cliquables ... 19
Exemples de code ... 9
expression ... 27
expression musicale ... 26
Expression musicale composite ... 43
expressions ... 18
expressions parallèles ... 27
expressions parallèles et hauteur relative ... 27
extenseur ... 85
Extensions de texte ... 110
extra-offset property ... 117
extra-offset property, example ... 120
extra-spacing-width ... 114
extra-spacing-width property ... 116
extra-spacing-width property, example ... 114, 119
extraits de code ... 10

F

fa, clef de ... 17
FDL, GNU Free Documentation License ... 150
fermata, implementing in MIDI ... 131
fichier PDF ... 12
fichiers, conseils de construction ... 19
fichiers, mise à jour de ... 38
Fingering, example of overriding ... 120
fingeringOrientations, exemple de propriété ... 108
fixing overlapping notation ... 117
font-series property, example ... 132
font-shape property, example ... 132
font-shape, exemple de propriété ... 94
font-size, exemple de propriété ... 88
fonte ... 2
fontSize , exemple de propriété ... 104
fontSize, exemple de propriété ... 103
force-hshift property ... 116
force-hshift property, example ... 122, 129
format d'entrée ... 41
formatage d'une partition ... 4
formatage, règles de ... 4
future mise à jour ... 38

G

gamme ... 14
Généralités en matière d'entrée et sortie ... 9
Gestion de l'espace ... 9
Glossaire musical ... 9
grace ... 25
GrandStaff ... 29
graver plusieurs voix ... 6
graveur ... 5
graveurs ... 66

graveurs, ajout ... 71
graveurs, suppression ... 71
gravure ... 2, 5
gravure automatisée ... 3
`Gravure des portées` ... 29
greffon ... 5
grob ... 85
grob, dimensionnement ... 114
grobs, positioning ... 120
grobs, propriétés ... 90

H

hampe, modification de longueur ... 103
hampes en bas ... 52
hampes en haut ... 52
`hauteur` ... 14, 20
hauteur relative et expressions parallèles ... 27
hauteur relative et musique simultanée ... 27
hauteurs ... 14
hauteurs, valeurs absolues ... 38
`header` ... 38
header ... 42
hiding objects ... 130

I

identificateurs ... 36, 43, 138
`Ignorer des passages de la partition` ... 138
imbrication d'expressions musicales ... 55
imbrication de constructions simultanées ... 55
implicit contexts ... 41
index ... 10
`Index de LilyPond` ... 10
`Index des commandes LilyPond` ... 10
indications métronomiques, modification du positionnement ... 110
insensibilité aux espaces multiples ... 18
`Installation` ... 10
`Instruments utilisant des portées multiples` ... 29
intégration de LilyPond avec d'autres programmes ... 10
interface ... 85, 94
`Interfaces pour les programmeurs` ... 10
interfaces, propriétés des ... 94
`intervalle` ... 14
invisible objects ... 130
invisible, silence ... 30
`is` ... 20
`isis` ... 20
italic, exemple ... 94

J

jargon ... 9

K

`key` ... 20

L

`La commande de fixation (set)` ... 71, 146
`La commande tweak` ... 89
langage ... 9
langage de programmation Scheme ... 4
langue ... 9
langues étrangères ... 9
layout ... 42
left-padding property ... 115, 118
legato ... 22
levée ... 25
`Levées` ... 26
`liaison d'articulation` ... 22
`liaison de prolongation` ... 21, 22
Liaison, exemple de dérogation ... 92, 93
liaisons d'articulation ... 21
`Liaisons d'articulation` ... 22
liaisons d'articulation et de prolongation, différences ... 22
liaisons de phrasé ... 22
`Liaisons de phrasé` ... 22
liaisons de prolongation ... 21
`Liaisons de prolongation` ... 22
liaisons de tenue ... 21
ligatures automatiques ... 24
ligatures et paroles ... 57
ligatures explicites ... 24
ligatures manuelles ... 24
ligne d'extension ... 32
`ligne de prolongation` ... 32
lihature ... 24
`LilyPond et les éditeurs de texte` ... 12
LilyPond et MacOS X ... 12
LilyPond et Unix ... 12
LilyPond et Windows ... 12
LilyPond Snippet Repository ... 10
`LilyPond-book` ... 10
lilypond-internals ... 10
lire la partition ... 12
`Liste des couleurs` ... 99
livre ... 41
longueur ... 103
LSR ... 10
`Lyrics` ... 28
Lyrics, création d'un contexte ... 56
LyricText, example of overriding ... 132
LyricText, exemple de dérogation ... 94

M

MacOS X et LilyPond ... 12
macros ... 36
magstep ... 103
magstep, exemple d'utilisation de la fonction ... 104
`majeur` ... 20
`major` ... 20
manually controlling tuplets, slurs, phrasing slurs, and beams ... 120
Manuel d'initiation ... 9
Manuel de notation ... 9
Manuel des références internes ... 90
manuel, lecture ... 19
`markup` ... 24
markup text, allowing collisions ... 112
markup, exemple ... 105
marques de repère, modification du positionnement ... 110

`mélisme` ... 32
mesure incomplète ... 25
méthodes de retouche ... 86
`métrique` ... 17
`Métrique` ... 18
MetronomeMark, example of overriding ... 118, 131, 132
midi ... 42
`mineur` ... 20
`minor` ... 20
mise à jour ... 38
mise à jour de fichiers ... 38
`Mise à jour des fichiers avec convert-ly` ... 38, 144
mise en forme ... 42
`Mise en forme de la partition` ... 42
mode absolu ... 38
mode lyrique, spécification de contexte en ... 95
mode relatif ... 14
mode relatif et altérations ... 14
modèles, création ... 81
modèles, modification des ... 74
modification de la taille des objets ... 100
`Modification de propriétés` ... 118
`Modification des greffons de contexte` ... 73
modification des propriétés d'un contexte ... 67
`Modification des réglages par défaut d'un contexte` ... 71, 73
`Modification des réglages prédéfinis` ... 10
modification du positionnement des indications métronomiques ... 110
modification du positionnement des marques de repère ... 110
modification du positionnement des numéros de mesure ... 110
modifier le positionnement des nuances ... 113
MultiMeasureRest, example of overriding ... 120
multiples portées ... 28
musique concurrente ... 48
musique et typographie ... 2
musique simultanée ... 48
musique simultanée et hauteur relative ... 27
`Musique vocale` ... 35, 36, 63

N

neutral ... 106
`new` ... 28
`new Staff` ... 28
`noire` ... 16
nolet, crochet ... 89
nolets ... 25
`Nolets` ... 26
nolets imbriqués ... 89
nommage des contextes ... 65
`Noms de note` ... 21
noms de note absolus ... 38
`Noms de note dans d'autres langues` ... 20, 21
notation des silences ... 16
`Notation musicale générale` ... 9, 40
`Notation polymétrique` ... 29
notation simple ... 13
`Notation spécialisée` ... 9
notation, contexte ... 28
note column ... 56
`note pointée` ... 16
note, durée ... 16
NoteColumn, example of overriding ... 122, 129
NoteHead, example of overriding ... 135
NoteHead, exemple de dérogation ... 87, 88, 100
`notes d'ornement` ... 25
`Notes d'ornement` ... 26
`Notes simultanées` ... 31
notes, nom des ... 38
notes, répartition selon le texte ... 112
nouveaux contextes ... 64
`nuances` ... 23
`Nuances` ... 24
nuances, modifier le positionnement ... 113
numéro de version ... 38
numéros de mesure, modification du positionnement ... 110

O

object collision within a staff ... 119
objects, aligning on a baseline ... 119
objects, hiding ... 130
objects, invisible ... 130
objects, making invisible ... 130
objects, outside-staff ... 105
objects, positioning ... 120
objects, removing ... 130
objects, within-staff ... 105
objet ... 85
objet de rendu ... 85
objet, propriétés ... 85
objets de rendu, propriétés ... 90
objets, conventions de nommage des ... 86
objets, modification de taille ... 100
objets, taille ... 100
`octave` ... 14
`once` ... 87, 92
ornementation ... 25
`ossia` ... 46, 100
ossias ... 46
ottava bracket ... 110
outside-staff objects ... 105
outside-staff-priority, exemple de propriété ... 111, 112
overlapping notation ... 117
`override` ... 86
override, commande ... 86
override, exemple ... 90
override, syntaxe ... 86
`overrideProperty` ... 88
overrideProperty, commande ... 88

P

padding ... 115, 117
padding property ... 115
padding property, example ... 117, 118
parallèles, expressions ... 27
`paroles` ... 31
paroles et ligatures ... 57
paroles et portées mutiples ... 35
paroles, affectation à une voix ... 56
paroles, alignement des ... 32

paroles, mot de plusieurs syllabes ... 32
partial ... 25
partition ... 41, 43
partition, formatage ... 4
partition, lire ... 12
PDF ... 12
phrasé ... 22
phrasé, liaisons de ... 22
phrasing slurs, controlling manually ... 120
PhrasingSlur, example of overriding ... 121
piano, portée pour ... 29
PianoStaff ... 29
Plusieurs partitions dans un même ouvrage ... 43
Plusieurs voix ... 53, 56
police ... 2
polyphonie ... 6
polyphonie ... 27
polyphonie ... 30, 48
portée double ... 29
portée pour piano ... 29
portée, positionnement ... 46
Portées d'ossia ... 47
portées multiples ... 27, 28
portées multiples et paroles ... 35
portées, regroupement de ... 29
portées, temporaires ... 46
positioning grobs ... 120
positioning objects ... 120
positions property ... 117
positions property, example ... 121
premier exemple ... 12
prise en compte de la casse ... 12, 18
prolongation, liaisons de ... 21
propriété, types de ... 95
propriétés ... 10
propriétés d'un contexte, définition avec \with ... 70
propriétés d'un contexte, modification ... 67
propriétés des interfaces ... 94
propriétés des objets ... 85
propriétés des objets de rendu ... 90
propriétés des objets graphiques (grobs) ... 90
propriétés, conventions de nommage des ... 86

R

recueil, exemple d'utilisation ... 61
Référence des propriétés internes ... 9
Références internes ... 90
Références internes, exemple d'utilisation ... 90
références, tables de ... 9
regroupement de portées ... 29
régulier, espacement ... 3
régulier, rythme ... 3
relative ... 14
removing objects ... 130
rendu, objets de ... 85
Résolution des collisions ... 126
retoucher ... 10
Retouches complexes ... 88
retouches, méthodologie ... 86
retour ... 92
revert ... 87, 92
revert, commande ... 87
rgb, couleur ... 99
rgb-color ... 99
rhythmes ... 16
right-padding property ... 115, 118
right-padding property, example ... 118
ronde ... 16

S

SATB, squelette ... 77
Scheme ... 10
Score ... 28
score, partition ... 41
scores, multiples ... 42
Script, example of overriding ... 117
self-alignment-X property ... 116
self-alignment-X property, example ... 119
sensibilité à la casse ... 12, 18
shift, commandes ... 56
silence ... 16
silence invisible ... 30
silences ... 16
simple, notation ... 13
Slur, exemple de dérogation ... 91, 92, 93
slurs, controlling manually ... 120
snippets ... 10
sol, clef de ... 17
spacing notes ... 55
spanner ... 85
spécification des durées ... 16
squelettes ... 19
staccato ... 22
Staff ... 28
staff-padding property ... 115
staff-padding property, example ... 119
staff-position property ... 116
staff-position property, example ... 120, 128
staff-space, exemple de propriété ... 104
StaffSymbol, exemple de dérogation ... 100, 104
startTextSpan ... 110
stem down ... 52
stem up ... 52
Stem, example of overriding ... 129, 130
Stem, exemple de dérogation ... 100, 106
stencil property, example ... 118, 132
stencil property, use of ... 131
stencil, exemple de propriété ... 97, 98, 101, 104
stencil, propriété ... 97
stopTextSpan ... 110
StringNumber, example of overriding ... 119
structure d'hymne ... 58
Structure d'une partition ... 46
structure d'une partition vocale ... 57
structure de ficher ... 41
Structure de fichier ... 41, 43
structures recursives ... 6
symboles musicaux ... 2
syntaxe ... 6
système ... 29
système pour chœur ... 29

T

tables de références ... 9
Tables du manuel de notation ... 10

taille d'objets 100
taille, modification 103
tenue, liaisons de 21
terminologie 9
text property, example 118
text spanner 110
texte, ajout de 24
texte, exemple de propriété 89
`textLengthOff` 112
`textLengthOn` 112
TextScript, exemple de dérogation 111, 112
TextSpanner, exemple de dérogation 110, 111
thickness, exemple de propriété 91, 92, 93
Tie, example of overriding 128
`time` 17
`times` 25
TimeSignature, exemple de dérogation 98, 100, 101, 103, 104
TimeSignature, exemple dérogation 102
titre 38
`Top` 9, 10
`Tout savoir sur les contextes` 64
trait d'union (paroles) 32
transparence 98
transparence, example de propriété 89
transparent property, example 129, 131
transparent property, use of 130
transparent propery, example 130
transparent, exemple de propriété 98
transparent, propriété 98
`transposition` 20
`triolet` 25
triolet, crochet 89
triolets 25
triolets imbriqués 89
`Tunable context properties` 68, 71
tuplet beams, controlling manually 120
tuplet-number, example de fonction 89
`TupletBracket` 89
TupletNumber, exemple de dérogation 89
`tweak` 88
tweak, commande 88
tweaks, using variables for 132
tying notes across voices 130
typographie 3, 5
typographie musicale 2

U

Unix et LilyPond 12
up 106
usage unique, dérogation à 92
using variables for tweaks 132
ut, clef d' 17
utilisation de variables 36
Utilisation des programmes 9, 10

V

`valeur d'une note` 25
variables 10, 36, 43, 138
variables, caractères autorisés dans les 36
variables, definition 36
variables, using for tweaks 132
variables, utilisation de 36
`Vérification des limites et numéros de mesure` 137
`Vérifications d'octave` 137
`version` 38
version, numéro de 38
versionage 38
virgule 14
`Voice` 28, 53
Voice, contexte 48
Voice, création de contextes 53
voix multiples 48
voix multiples sur une portée 30
voix temporaires 55
voix, imbrication 55

W

Windows et LilyPond 12
within-staff objects 105

X

X11, couleurs 99
`x11-color` 99
x11-color function, example of using 135
x11-color, exemple d'utilisation 100