LilyPond

Le système de gravure musicale

Manuel d'initiation

L'équipe de développement de LilyPond

Pour LilyPond version 2.11.58

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 des partitions ... 86

4.1 Retouches élémentaires ... 86
4.1.1 Introduction aux retouches ... 86
4.1.2 Objets et interfaces ... 86
4.1.3 Conventions de nom des objets et propriétés ... 86
4.1.4 Méthodes de retouche ... 86
4.2 Le manuel des références internes ... 86
4.2.1 Propriétés des objets de rendu ... 86
4.2.2 Propriétés listées par interface ... 86
4.2.3 Types de propriétés ... 86
4.3 Apparence des objets ... 86
4.3.1 Visibilité et couleur des objets ... 86
4.3.2 Taille des objets ... 86
4.3.3 Longueur et épaisseur des objets ... 86
4.4 Positionnement des objets ... 86
4.4.1 Comportement automatique ... 86
4.4.2 Objets inclus dans la portée ... 86
4.4.3 Objets hors de la portée ... 86
4.5 Collisions d'objets ... 86
4.5.1 Déplacement d'objets ... 86
4.5.2 Correction des collisions d'objets ... 89
4.5.3 Exemple concret ... 90
4.6 Retouches courantes ... 90
4.7 Autres retouches ... 92
4.7.1 Autres utilisations des retouches ... 92
4.7.2 Utilisation de variables dans les retouches ... 92
4.7.3 Autres sources de documentation ... 92
4.7.4 Options ralentissant le traitement ... 92
4.7.5 Retouches avancées avec Scheme ... 92

5 Travail sur des projets LilyPond 94
5.1 Suggestions de saisie des fichiers LilyPond 94
5.1.1 Suggestions générales 94
5.1.2 Gravure de musique existante 95
5.1.3 Projets d'envergure 95
5.1.4 Économies de saisie grâce aux identificateurs et fonctions 95
5.1.5 Feuilles de style 97
5.2 Quand ça ne fonctionne pas 101
5.2.1 Mise à jour d'anciens fichiers 101
5.2.2 Résolution de problèmes — tout remettre à plat 101
5.2.3 Exemples minimaux 102
5.3 Conducteurs et parties 102

Annexe A Modèles 104
A.1 Portée unique 104
A.1.1 Notes seules 104
A.1.2 Notes et paroles 104
A.1.3 Notes et accords 104
A.1.4 Notes, paroles et accords 104
A.2 Modèles pour claviers 104
A.2.1 Piano seul 104
A.2.2 Chant et accompagnement 104
A.2.3 Piano et paroles entre les portées 104
A.2.4 Piano et nuances entre les portées 104
A.3 Quatuor à cordes 104
A.3.1 Quatuor à cordes 104
A.3.2 Parties pour quatuor à cordes 104
A.4 Ensemble vocal 104
A.4.1 Partition pour chœur à quatre voix mixtes 104
A.4.2 Partition pour chœur SATB avec réduction pour piano 104
A.4.3 Partition pour chœur SATB avec alignement des contextes 104
A.5 Exemples de notation ancienne 104
A.5.1 Transcription de musique mensurale 104
A.5.2 Transcription du grégorien 104
A.6 Symboles de jazz 104
A.7 Squelettes pour lilypond-book 104
A.7.1 LaTeX 104
A.7.2 Texinfo 105
A.7.3 xelatex 105

Annexe B Tutoriel Scheme 106
B.1 Scheme et les retouches 106

Annexe C Licence GNU de documentation libre 107

Annexe D Index de LilyPond 113

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 en jouer semble naturel, c'est un vrai bonheur. Et si d'autres jours on ne peut tout simplement rien en tirer, on continue quand même à travailler, jour après jour.

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 est sans doute que notre logiciel est véritablement utile aux gens. En flânant sur Internet, nous trouvons beaucoup de gens qui se servent de LilyPond, et réalisent des partitions très impressionnantes : 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

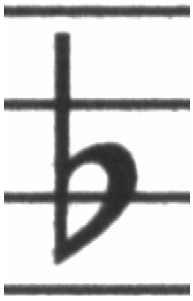
Cette partie présente les objectifs de LilyPond ainsi que son architecture.

Gravure

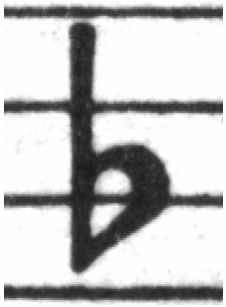
L'art de la typographie musicale se nomme la *gravure*. Ce terme est issu du processus traditionnel d'impression musicale. Il y a seulement quelques dizaines d'années, on faisait les partitions en coupant et en embossant une plaque de zinc ou d'étain en image miroir. Cette plaque était ensuite encrée, 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'éditeur 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.

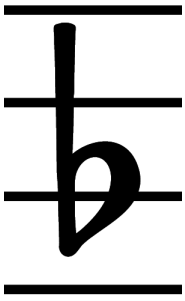
Les images ci-dessous illustrent la différence entre la gravure traditionelle et l'impression typique par ordinateur, et la troisème image montre comment LilyPond mime l'aspect traditionnel. L'image de gauche est une numérisation d'un symbole bémol d'une édition publiée en 2000. Celle du centre montre 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)

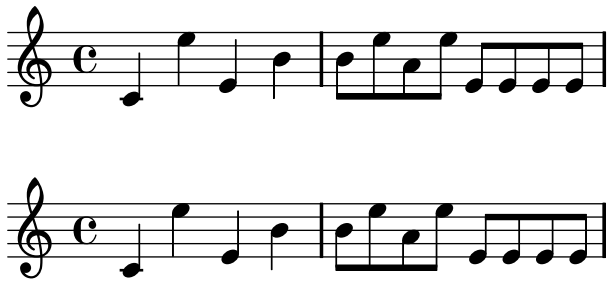


Bärenreiter (1950)



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 n'ont généralement pas. LilyPond représente notre effort pour apporter 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 ?

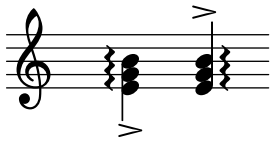
La réponse est : nous ne le pouvons pas. La typographie se base sur le jugement visuel humain, donc les humains ne peuvent pas être complètement remplacés. Si LilyPond arrive à résoudre la plupart des situations correctement, ce sera déjà une grande avancée sur les logiciels existants. Les autres situations peuvent être résolues 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 :

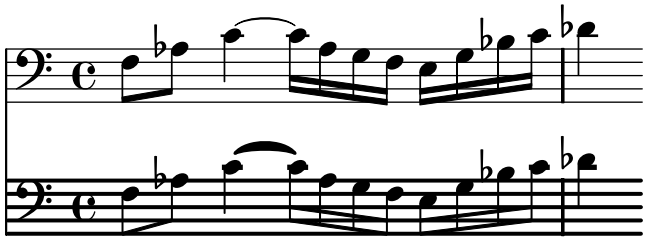
- Quand Lilypond fait des erreurs, les utilisateurs ont besoin de contredire les décisions de formatage. Les utilisateurs doivent donc avoir accès au moteur de formatage. Par conséquent, les règles et les propriétés ne peuvent pas être fixées par nous au moment de la compilation, mais doivent être accessibles 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 language 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 language 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.

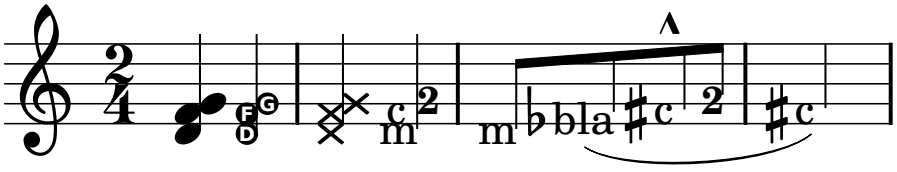
Les variables Scheme contrôlent les décisions de mise en page. Par exemple, beaucoup d'objets graphiques ont une variable de direction qui encode le choix entre haut et bas (ou gauche et droite). Vous pouvez voir ici deux accords, avec des accents, et des arpèges. Dans le premier accord, les objets graphiques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

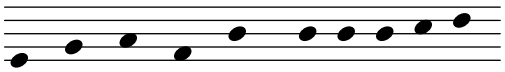
Le processus de formatage décide où placer les symboles. Cependant, cela ne peut être fait qu'à partir du moment où il a été décidé *quels* symboles doivent être imprimés, c'est-à-dire quelle notation utiliser.

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'a 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.

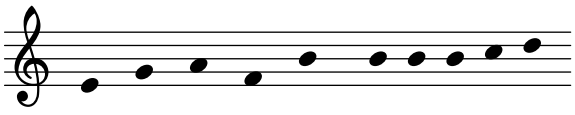
Comment pouvons nous appréhender un tel monstre à plusieurs têtes, et le confiner dans l'espace réduit d'un programme informatique ? Notre solution consiste à diviser le problème de la notation — par opposition à la gravure, ou typographie — en morceaux digestes et programmables : chaque type de symbole 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`) :

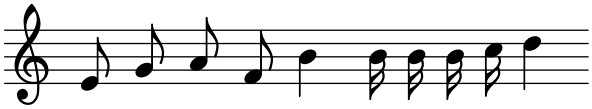
Ensuite, le graveur du symbole de portée (`Staff_symbol_engraver`) ajoute la portée



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



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

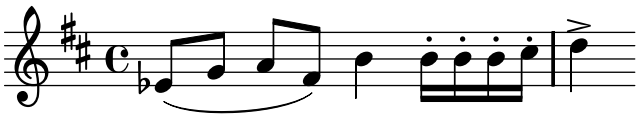


Le graveur de hampe est notifié de chaque tête de note qui survient. Chaque fois qu'une tête de note — plusieurs pour un accord — est rencontrée, un objet hampe est créé et connecté à la

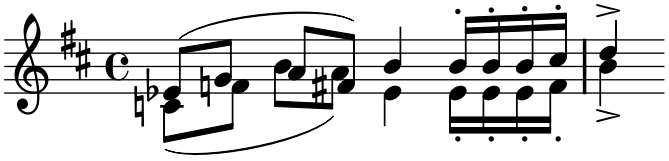
[1] traduction de l'anglais *plug-in*.

[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.



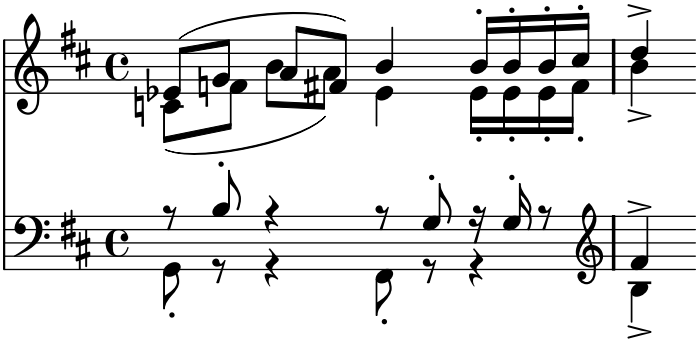
Ce système fonctionne bien pour de la musique monodique, mais qu'en est-il de la polyphonie ? En notation polyphonique, plusieurs voix peuvent partager une portée.



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

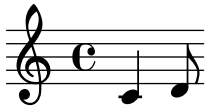
Idéalement, le format d'entrée pour n'importe quel système de formatage est une description abstraite du contenu. Dans ce cas-ci, ce serait la musique elle-même. Cela pose un formidable problème : comment pouvons-nous définir ce que la musique est réellement ? Plutôt que d'essayer de trouver une réponse, 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.

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

La syntaxe est également l'interface utilisateur pour LilyPond, par conséquent il est facile de saisir

```
{
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. A 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

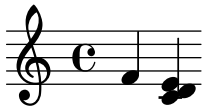
```
f4
```

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

```
<<c4 d4 e4>>
```

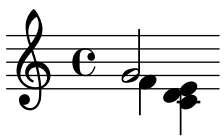
Cette expression est mise dans une séquence grace à l'encadrement par des accolades `{ ... }`

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



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

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



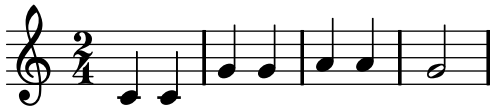
De telles strucutres récursives peuvent être spécifiées formellement et de manière ordonnée dans une grammaire indépendante de tout contexte. Le code d'analyse est aussi générée à partir de cette grammaire. Autrement dit, la syntaxe de LilyPond est définie clairement et sans ambiguité.

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

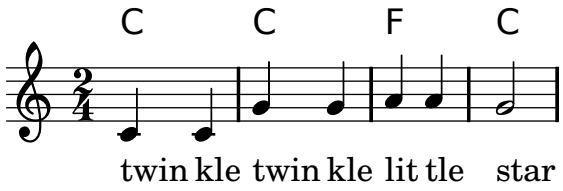
discussions sur les goûts ont leur mérite, elles ne sont pas très productives. D'un point de vue plus large sur LilyPond, l'importance de la syntaxe est minime : il est facile d'inventer une syntaxe concise, alors qu'écrire un code de formatage décent est beaucoup plus difficile. Ceci est également illustré 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 :

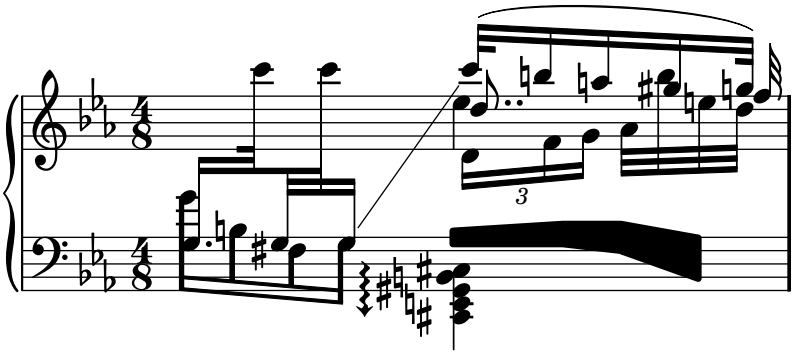


La notation polyphonique et la musique pour piano peuvent également être générées. L'exemple suivant associe quelques constructions plus exotiques :

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 des partitions], page 86 : introduction aux retouches de gravure avec LilyPond.
- Chapitre 5 [Travail sur des projets LilyPond], page 94 : 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 104 de pièces LilyPond. Copiez et collez un modèle dans un fichier, ajoutez les notes, et c'est prêt !
- Annexe B [Tutoriel Scheme], page 106 : 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 n'ont jamais touché 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 "General input and output" 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

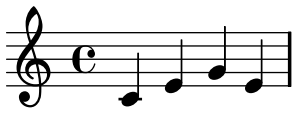
Cette section présente les aspects élémentaires de l'utilisation de LilyPond.

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'
}
```

Le compilation de cette partition 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 faire 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

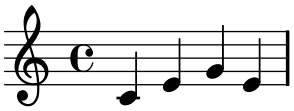
```
lilypond test.ly
GNU LilyPond 2.11.58

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 clé, un chiffre de mesure et du rythme.

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



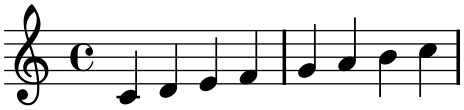
Ces valeurs automatiques simplifient la saisie du code source dans bien des cas ; nous verrons plus loin comment les indiquer explicitement.

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 accidentelle” dans *Glossaire*.

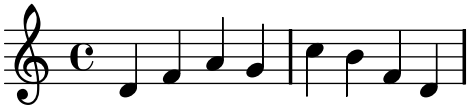
Le moyen le plus simple d’entrer des notes est d’utiliser le mode d’octaves relatives, ou mode `\relative`. Dans ce mode, l’octave de chaque note est sélectionnée automatiquement de façon à ce qu’elle soit la plus proche possible de la note précédente, c’est-à-dire de façon à ce que l’intervalle avec 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 clé de sol et la clé 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` :

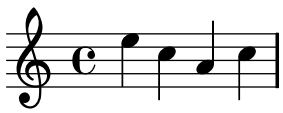
```
\relative c' {
  d f a g
  c b f d
}
```



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.

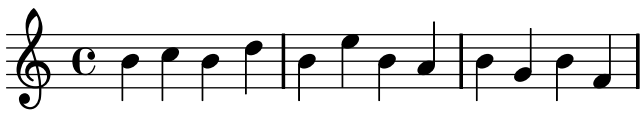
Dans l’exemple suivant, on remplace `c'` dans la commande `\relative c’` par `c''`, afin de calculer l’octave de la première note par rapport au do situé une octave au-dessus du do central :

```
% 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 sur 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.

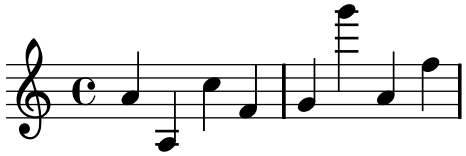
```
\relative c'' {
  b c  % do est à un cran de plus, il sera donc au dessus
  b d  % ré est à 2 crans de plus ou 5 de moins, il sera donc au dessus
  b e  % mi est à 3 crans de plus ou 4 de moins, il sera donc au dessus
  b a  % la est à 6 crans de plus ou 1 de moins, il sera donc en dessous
  b g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

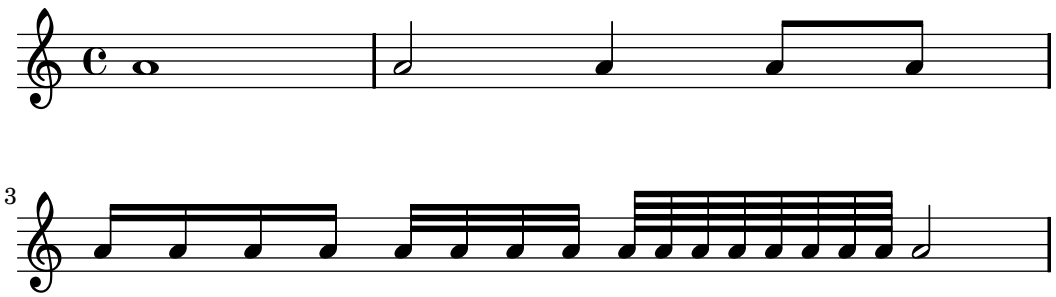
Durées et rythme

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 ajoutées automatiquement.

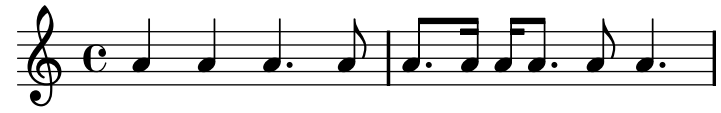
Si aucune durée n’est indiquée pour une note, la dernière durée entrée est utilisée. En l’absence d’indication de durée, la première note est une noire.

```
\relative c'' {
  a1
  a2 a4 a8 a
  a16 a a a a32 a a a a64 a a a a a a a a2
}
```



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

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

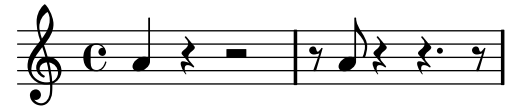


Silences

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

On saisit un *silence* tout comme une note, mais avec la lettre ‘`r`’.

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

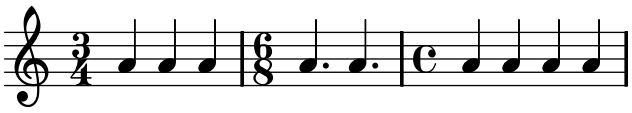


Métrique

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
}
```

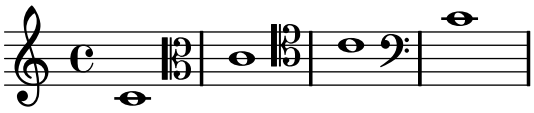


Clefs

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

La *clef* peut être définie à l’aide de la commande `\clef` :

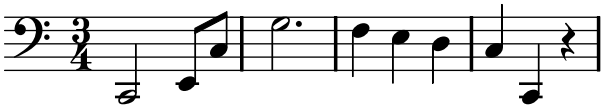
```
\relative c' {
  \clef treble
  c1
  \clef alto
  c1
  \clef tenor
  c1
  \clef bass
  c1
}
```



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 minuscules : `{ 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 sis
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.

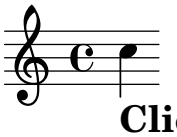
Pour reproduire les exemples, vous pouvez copier et coller le code affiché, mais **à condition** d'ajouter `\relative c'' { }` de la façon suivante :

```
\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 de saisie des fichiers LilyPond], page 94. 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

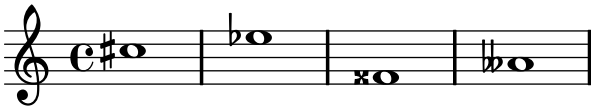
Altérations

Glossaire musical : Section “dièse” dans *Glossaire*, Section “bémol” dans *Glossaire*, Section “double dièse” dans *Glossaire*, Section “double bémol” dans *Glossaire*, Section “altération accidentelle” 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*.

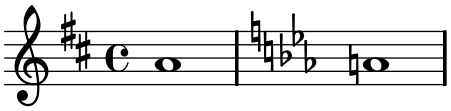
```
cis1 ees fisis, aeses
```



Armures

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

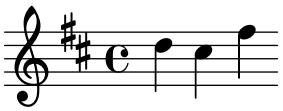
Glossaire musical : Section “altération accidentelle” dans *Glossaire*, Section “armure” dans *Glossaire*, Section “hauteur” dans *Glossaire*, Section “bémol” dans *Glossaire*, Section “bécarre” dans *Glossaire*, Section “dièse” dans *Glossaire*, Section “transposition” dans *Glossaire*.

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 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,

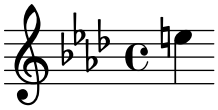
```
\key d \major
d cis fis
```



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

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

```
\key aes \major
e
```



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*.

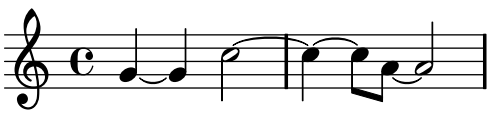
2.2.2 Liaisons

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
```



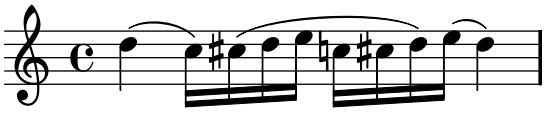
[1] parfois aussi appelée liaison de tenue

Liaisons d'articulation

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

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

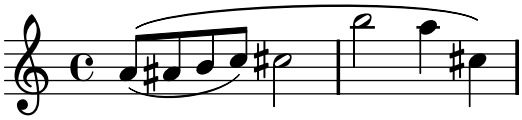
```
d4( c16) cis( d e c cis d) e( d4)
```



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

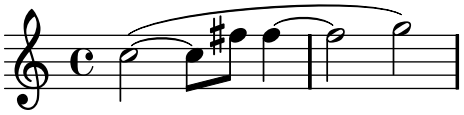


Attention aux types de liaison

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*.

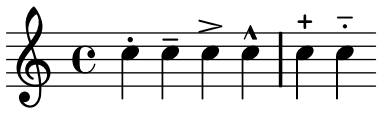
2.2.3 Articulations et nuances

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-_
```

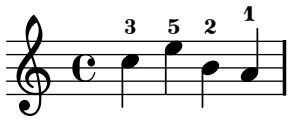


Doigtés

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

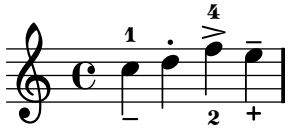
De même, des indications de doigté peuvent être ajoutées à une note en utilisant un tiret (‘-’) et le chiffre à écrire :

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



Articulations et doigtés sont habituellement placés automatiquement, mais vous pouvez indiquer une direction 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^-_+
```

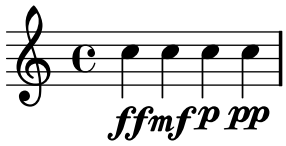


Nuances

Glossaire musical : Section “nuances” dans *Glossaire*, Section “crescendo” dans *Glossaire*, Section “decrescendo” dans *Glossaire*.

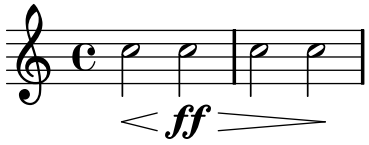
On obtient un signe de *nuance* en ajoutant à la note les lettres du signe, précédées d’un anti-slash ‘\’ :

```
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\!
```



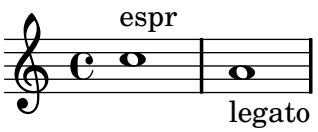
Voir aussi

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

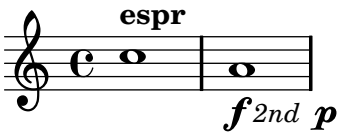
On peut ajouter du texte à une partition :

```
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

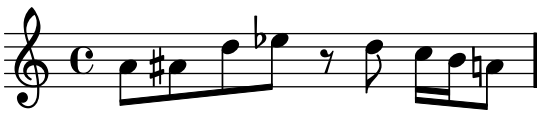
Manuel de notation : Section “Ajout de texte” dans *Manuel de notation*.

2.2.5 Barres de ligature automatiques et manuelles

Glossaire musical : Section “barre de ligature” dans *Glossaire*.

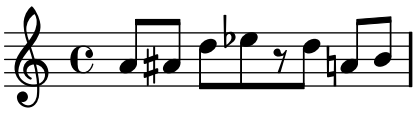
Toutes les barres de ligature sont dessinées automatiquement :

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



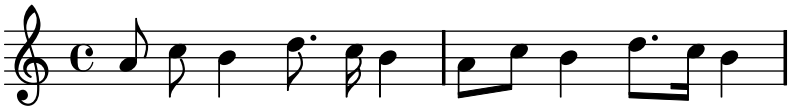
Lorsqu’on n’aime pas la manière dont les notes sont automatiquement groupées, il est possible de les ligaturer manuellement, en marquant la première note à attacher d’un crochet ouvrant `[` et la dernière d’un crochet fermant `]`.

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



Pour désactiver les barres de ligature automatqiues 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*.

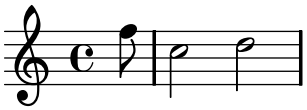
2.2.6 Commandes rythmiques avancées

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
```

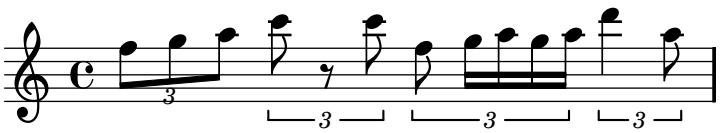


Nolets

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 }
```

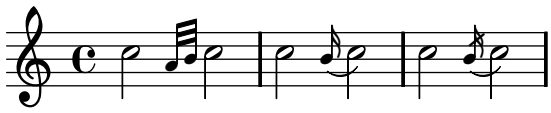


Notes d’ornement

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.

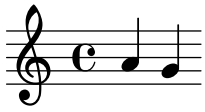
2.3.1 Les expressions musicales en clair

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
```

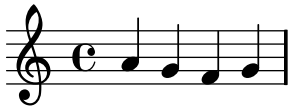
Mettre un groupe de notes entre accolades crée une nouvelle expression musicale, appelée *expression musicale composée*. En voici un exemple avec deux notes :

```
{ 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)
```

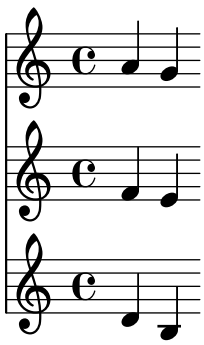
Ceci est une suite d'expressions, où chacune est contenue dans la suivante. Les expressions les plus simples sont les nombres, et de plus grandes expressions sont produites en combinant des expressions avec des opérateurs — comme '`+`', '`*`' et '`/`' — et des parenthèses. Tout comme les expressions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



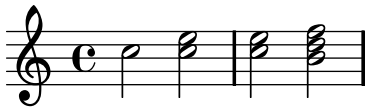
Notez que nous avons ici indenté chaque niveau du fichier d'entrée avec un nombre d'espaces différent. LilyPond se moque — ou presque — de l'espace qu'il peut y avoir ou non au début d'une ligne, mais un code bien indenté est bien plus lisible par des humains.

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>> } >>
}
```

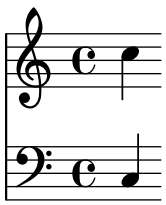


2.3.2 Plusieurs portées

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,, }
  >>
}
```



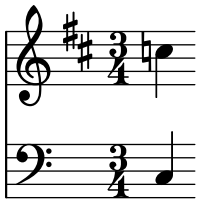
La commande `\new` introduit un « contexte de notation ». Un contexte de notation est un environnement dans lequel les événements musicaux — comme les notes ou les commandes `\clef` — sont interprétés. Pour des pièces simples, ces contextes sont créés automatiquement. Pour des pièces plus complexes,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

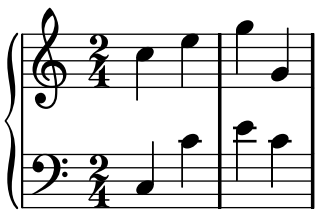
Glossaire musical : Section "accolade" dans *Glossaire*.

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 à clavier" dans *Manuel de notation*, Section "Gravure des portées" dans *Manuel de notation*.

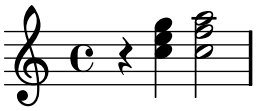
[2] Ce comportement peut être modifié si nécessaire, voir Section "Notation polymétrique" dans *Manuel de notation*.

2.3.4 Combinaison de notes en accords

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

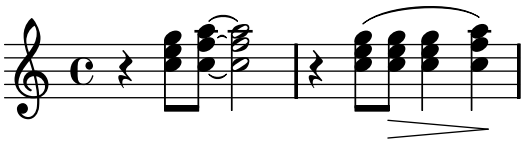
Nous avons vu précédemment comment combiner des notes simultanément, en les encadrant par des angles doubles `<<` et `>>`. Pour produire des accords simples, c'est-à-dire une superposition de notes de même durée, on encadre les hauteurs de notes par des angles simples `<` et `>`, et on écrit la durée 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 indiquat un début ou 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>\!)
```

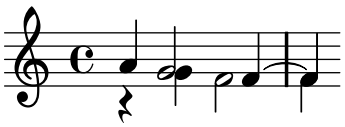


2.3.5 Polyphonie sur une portée

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, *i.e.* avec `{...}`, puis en combinant simultanément les voix et en les séparant par `\\`.

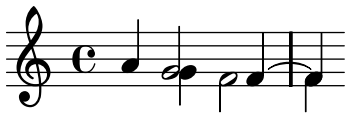
```
<<
  { a4 g2 f4~ f4 } \\
  { r4 g4 f2 f4 }
>>
```



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` à la place d'un silence normal `r` :

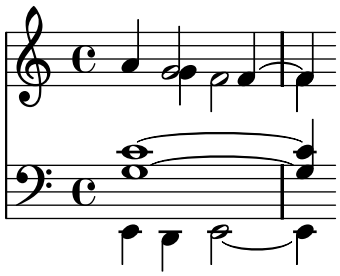
```
<<
  { a4 g2 f4~ f4 } \\
  { s4 g4 f2 f4 }
>>
```

[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

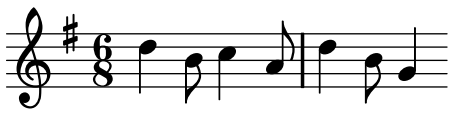
Cette section présente l'écriture vocale et les partitions de variété.

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.*

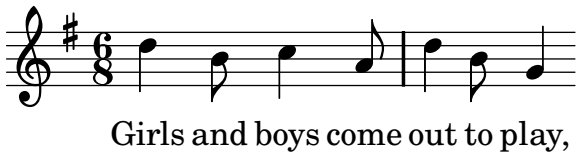
```
\relative c'' {
  \key g \major
  \time 6/8
  d4 b8 c4 a8 d4 b8 g4
}
```



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,
  }
>>
```



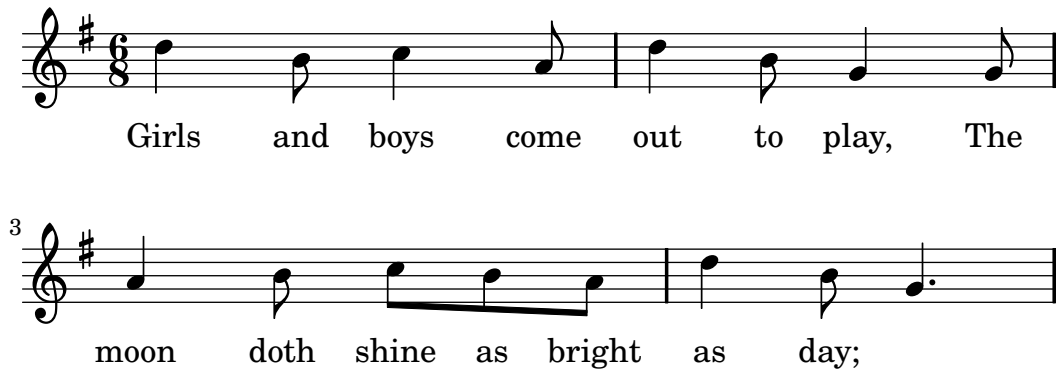
Remarquez les accolades embrassant la musique et celles embrassant les paroles, ainsi que les angles doubles encadrant toute la pièce ; ces derniers indiquent simplement que la musique et les paroles se produisent en même temps.

2.4.2 Alignement des paroles sur une mélodie

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

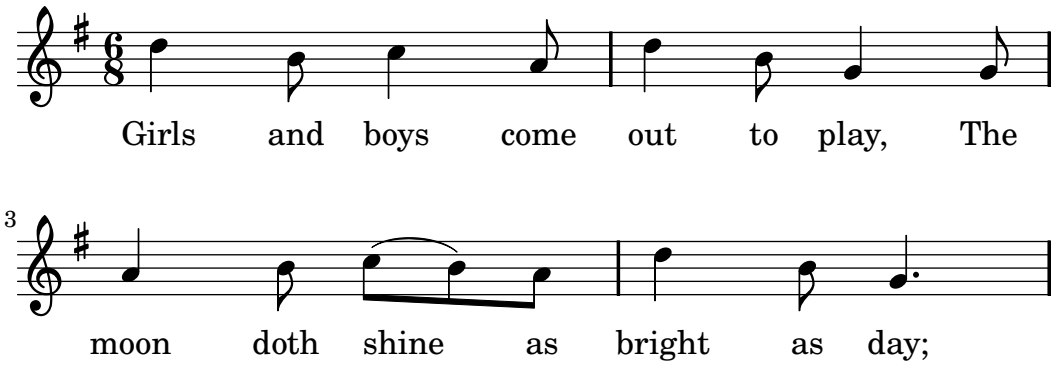
La ligne suivante de la comptine précédente est *The moon doth shine as bright as day.* Ajoutons-la au code.

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



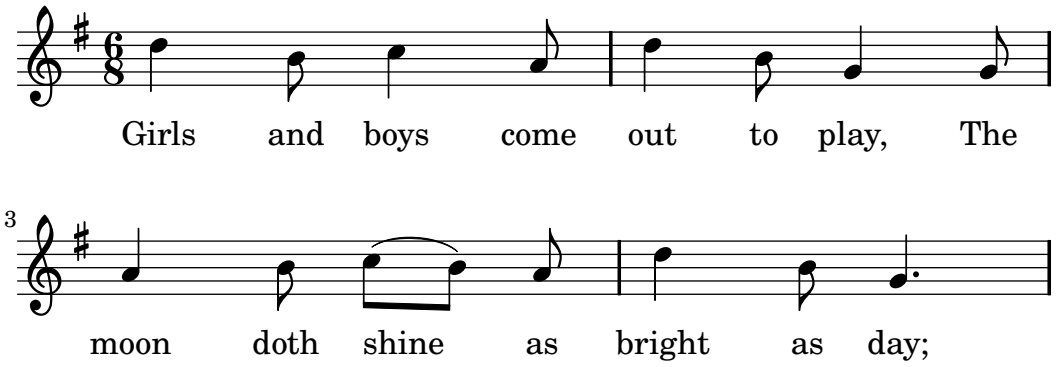
Remarquez que les paroles ajoutées ne s’alignent pas bien avec les notes. Le mot *shine* devrait être chanté sur deux notes au lieu d’une. On appelle ceci un *mélisme* : il s’agit d’une seule syllabe chantée sur plus d’une note. Il existe plusieurs façons d’étaler une sylabe sur plusieurs notes, la plus 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;
  }
>>
```



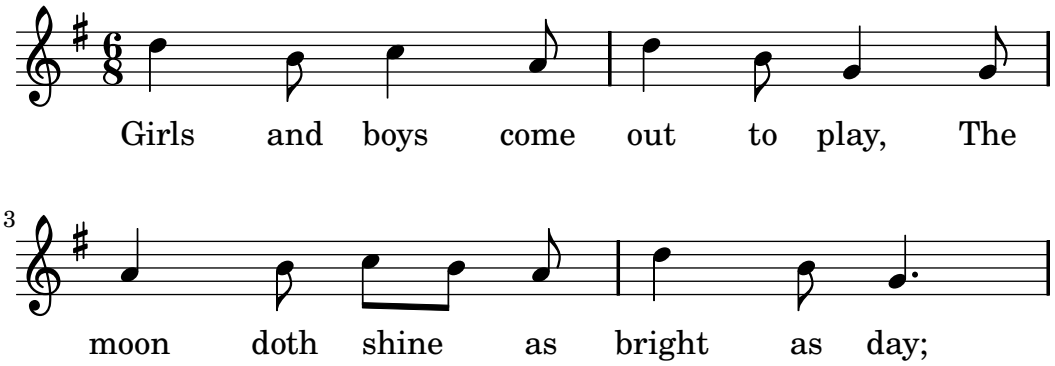
Les paroles sont maintenant correctement alignées, mais les liens de croche automatiques ne conviennent pas pour les notes au-dessus de *shine as*. On peut les corriger en ajoutant des liens de croche manuels, pour ceci consultez Section 2.2.5 [Barres de ligature automatiques et manuelles], page 24.

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



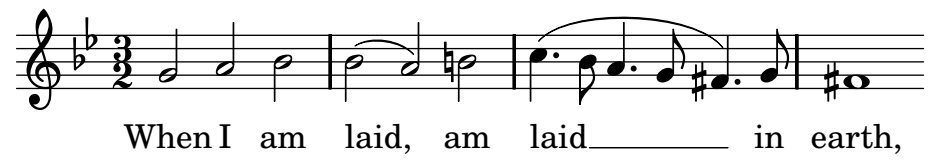
Au lieu d'utiliser une liaison, on peut indiquer le mélisme dans les paroles en insérant un caractère souligné _ pour chaque note du mélisme sauf la première.

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



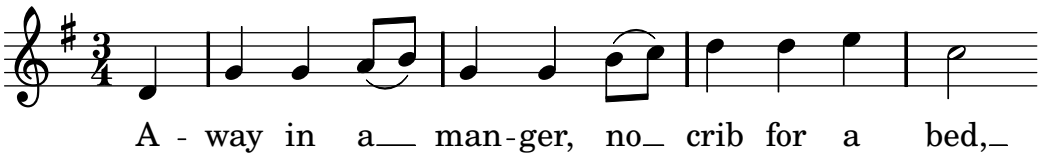
Si une syllabe s'étend sur un grand nombre de notes ou une note très longue, on représente souvent le mélisme par un *trait de prolongation*, qu'on entre avec __. L'exemple suivant montre les trois premières mesures de la plainte de Didon, extraite de *Didon et Énée* de Purcell.

```
<<
  \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,
  }
>>
```



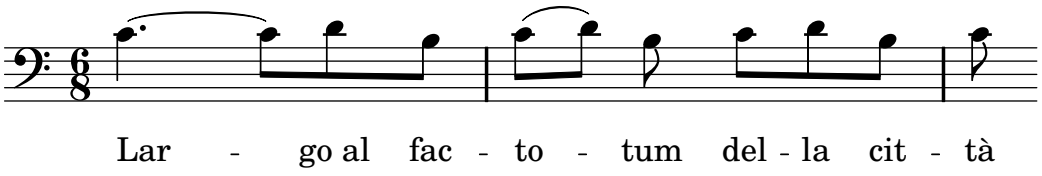
Aucun exemple jusqu'à présent n'a utilisé de mots de plus d'une syllabe. Dans des paroles, de tels mots sont écrits en syllabes séparées par des traits d'union. Avec LilyPond, on utilise deux tirets pour produire un trait d'union centré entre deux syllabes. L'exemple suivant montre tout ce que nous avons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



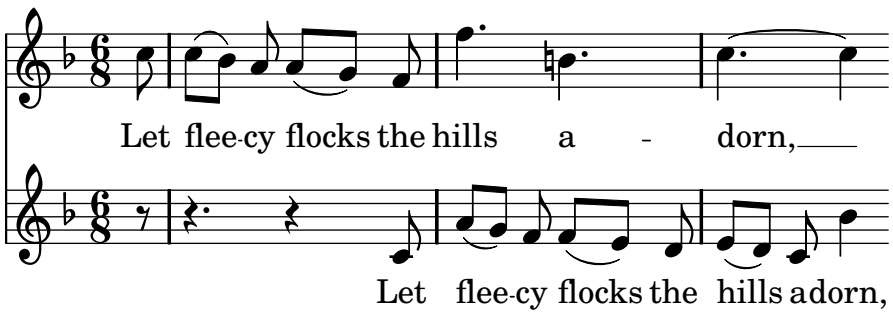
Voir aussi

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

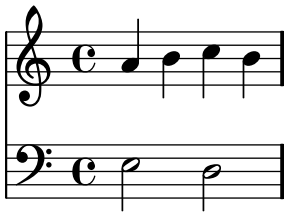
Lorsque l'on combine tous les éléments étudiés précédemment pour écrire des partitions plus longues, les expressions musicales prennent de l'ampleur et, dans le cas des pièces polyphoniques, deviennent profondément imbriquées, jusqu'au point où il devient difficile de se repérer dans le fichier source.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

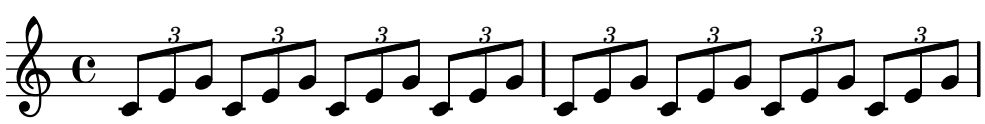
```
violon = \new Staff {
  \relative c'' {
    a4 b c b
  }
}
cello = \new Staff {
  \relative c {
    \clef bass
    e2 d
  }
}
{
  <<
    \violon
    \cello
  >>
}
```



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 "2.11.58"
```

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 voir 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 "2.11.58"
\header {
  title = "Symphonie"
  composer = "Moi"
  opus = "Op. 9"
}

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

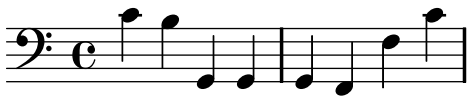
Quand LilyPond traite le fichier, le titre et le compositeur sont imprimés au début de la partition. Vous trouverez plus d'informations sur les titres à la section Section "Création de titres" dans *Manuel de notation*.

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.

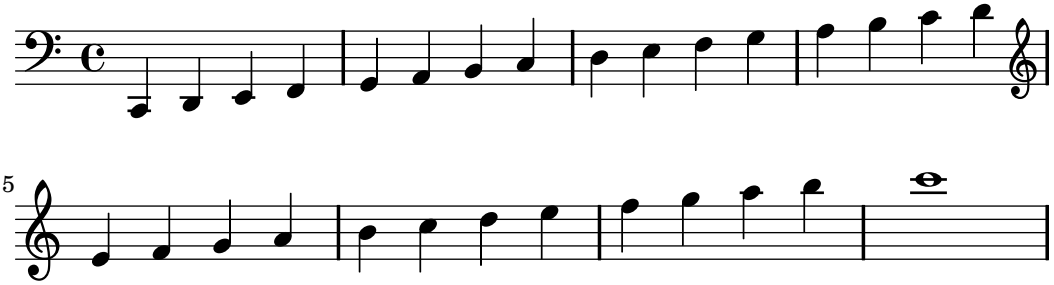
Si vous omettez la commande `\relative`, LilyPond considérera toutes les hauteurs comme des hauteurs absolues. Un `c'` désigne toujours le do central, un `b` se situe une seconde en dessous du do central, et un `g,` est situé sur la première ligne de la portée en clé de fa.

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



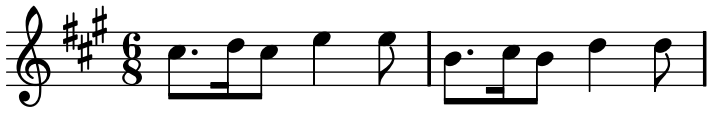
Voici une gamme sur 4 octaves :

```
{
  \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
}
```



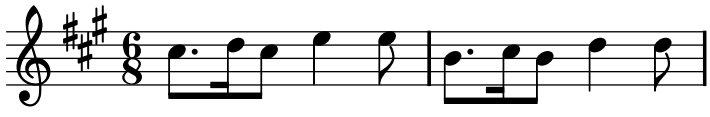
Comme vous pouvez le voir, il faut beaucoup d'apostrophes pour écrire de la musique dans un registre aigu, comme le montre cet extrait de Mozart.

```
{
  \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, donc serait 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 104. 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 recherche 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 "2.11.58"
\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 racourci 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`, même si elles seront 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 vue 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 GrandStaff <<
    ...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 (*backshash* pour \), *elles ne constituent pas* des expressions musicales et ne seront donc pas interprétées comme telles. Elles peuvent de ce fait être mentionnées à l'intérieur du bloc `\score`, tout comme à l'extérieur. En réalité, ces commandes sont la plupart du temps indépendantes du bloc `\score` – la commande `\header` intervient bien souvent avant le bloc `\score` tout simplement parce que les en-têtes apparaissent au début de la partition. C'est donc l'un des raccourcis que LilyPond prendra en considération.

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*) 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 [Économies de saisie grâce aux identificateurs et fonctions], page 95.

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 GrandStaff <<
    ...insérez ici l'intégralité d'un opéra de Wagner...
```

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

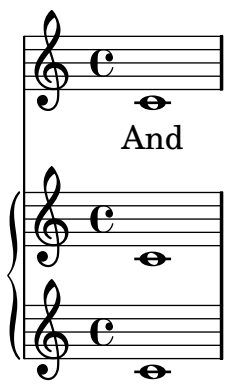
```
    >>
  }   % cette accolade marque la fin de l'expression musicale
  \layout { }
}
```

Un opéra de Wagner multiplierait facilement la longueur de ce manuel par deux ou trois, alors faisons-le en version chant/piano. On n'a plus besoin d'une partition d'orchestre — `GrandStaff` — donc laissons cela de côté. Par contre, un chanteur et un piano *pourraient* nous être utiles.

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



On y voit nettement plus clair maintenant. Nous voici donc avec la partie du chanteur, qui contient un ensemble `Voice`, ce qui dans LilyPond correspond à une voix, au sens de voix d'une polyphonie plutôt que de voix chantée — ce pourrait être une partie de violon par exemple.

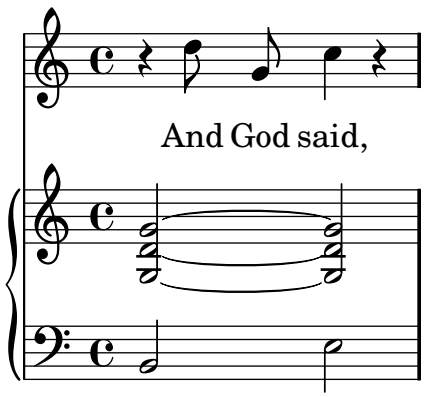
Nous avons également une partie de piano, qui contient deux portées : une pour la main droite, une autre pour la main gauche.

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 pourrins 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 = "singer" <<
      \new Voice = "vocal" { \melodie }
      \addlyrics { \texte }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "upper" { \superieur }
      \new Staff = "lower" {
        \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 respectivemen 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 !

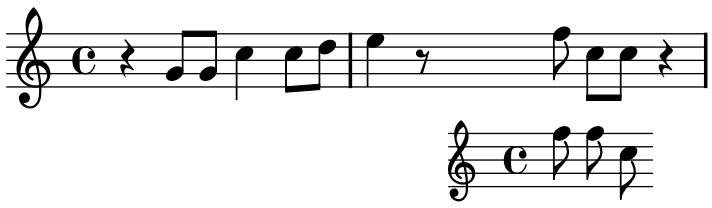
Voir aussi

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 :

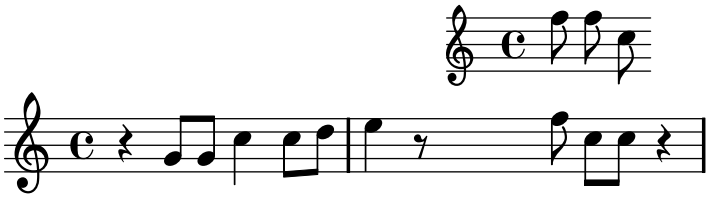
```
\new Staff {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f c c }
      \new Staff {
        f8 f c
      }
    >>
    r4 |
  }
}
```



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



Cet exemple recourt à `\with`, que nous verrons en détail plus avant. C'est un moyen de modifier le comportement par défaut d'une portée individuelle. Nous indiquons ici que la nouvelle portée doit se placer au dessus de la portée « main » 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 86 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 :

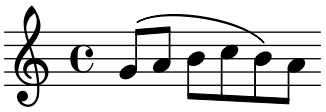
Type de crochet	**Fonction**
`{ .. }`	Délimite un segment de musique séquentielle
`< .. >`	Délimite les notes d'un accord
`<< .. >>`	Délimite des sections simultanées
`( .. )`	Marquent le début et la fin d'une liaison
`\( .. \)`	Marquent le début et la fin d'une liaison de phrasé
`[ .. ]`	Marquent le 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.

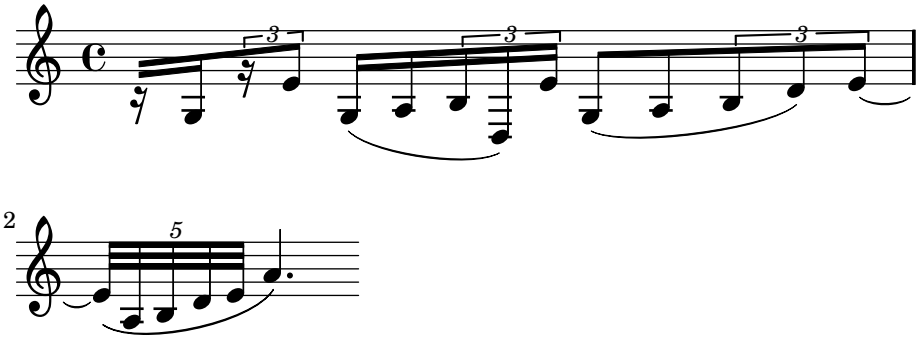
Ainsi, et bien que ce ne soit pas très musical, une liaison de phrasé peut débuter avant l'insertion d'une ligature manuelle et s'arrêter avant la fin de la ligature :

```
{ 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.

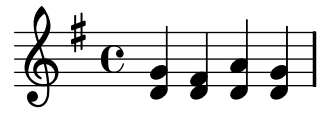
3.2.1 J'entends des Voix

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 fat 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>
```



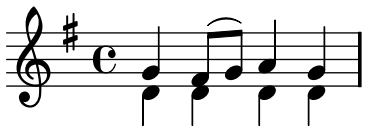
Nous exprimons ici chacun des accords par l'utilisation de symboles inférieur et supérieur simples, `< ... >`, puisque nous n'avons besoin que d'une seule voix. Supposons maintenant que le fa dièse soit une croche, suivie d'un sol croche – une note de passage vers le la ? Nous avons alors deux notes 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 symboles supérieur et inférieur doubles, `<<...>>`. 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 :

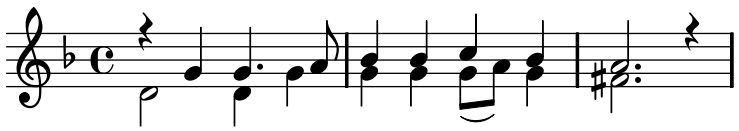
```
\key g \major
%    Voice "1"                    Voice "2"
<< { g4 fis8( g) a4 g }    \\ { d4 d d d }  >> |
```



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

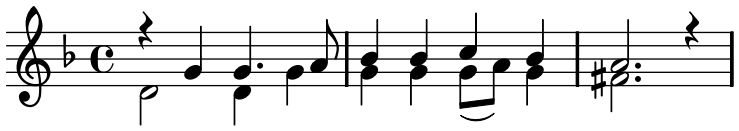
Autre exemple :

```
\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 :

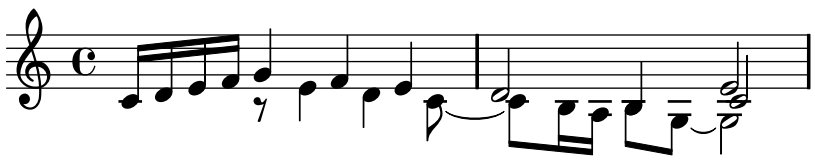
```
\key d \minor
<< {
  % Voice "1"
  r4 g g4. a8 |
  bes4 bes c bes |
  a2. r4 |
} \\ {
  % Voice "2"
  d,2 d4 g |
  g4 g g8( a) g4 |
  fis2. s4 |
} >>
```



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.

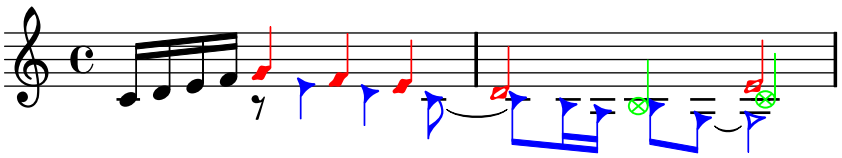
```
\new Staff \relative c' {
  % Voix principale
  c16 d e f
  %    Voice "1"     Voice "2"                  Voice "3"
  << { g4 f e } \\ { r8 e4 d c8 ~ } >> |
  << { d2 e2 }  \\ { c8 b16 a b8 g ~ g2 } \\ { s4 b4 c2 } >> |
}
```



Ces voix sont séparées de la voix principale qui 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
     % La voix 1 continue
    { d2 e2 }
  \\
     % La voix 2 continue
    { 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 cercles 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 86 et Section 4.7.2 [Utilisation de variables dans les retouches], page 92.

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

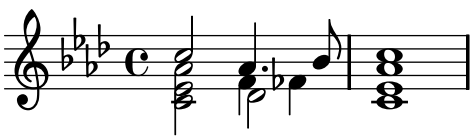
```
\relative c' { noteA << < noteB noteC > \\ noteD >> noteE }
```

`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 :

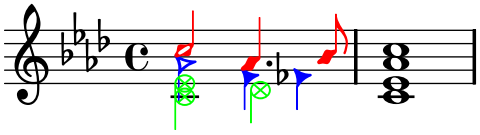
```
\relative c' { noteA ... }
<<
  \relative c'' { < noteB noteC > ... }
\\
  \relative g' { noteD ... }
>>
\relative c' { noteE ... }
```

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.



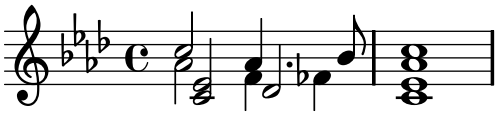
La direction des hampes sert souvent à indiquer dans la continuité deux lignes mélodiques simultanées. Ici, les hampes des notes les plus hautes vont vers le haut, et celles des notes plus basses vers le bas. C'est une première incation que nous avons eu recours à plus d'une voix.

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 ecrire 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 motes de chacune d'entre elles. Une fois de plus, nous reviendrons plus tard sur le code que vous ne comprendriez pas.



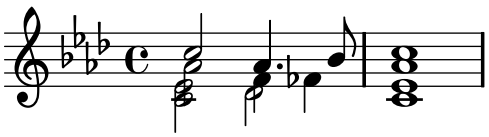
Essayons à présent de coder cette musique en partant de zéro. Comme nous le verrons, certaines difficultés vont se présenter. Partons de ce que nous avons appris : utilisons la construction << \\ >> pour saisir la première mesure dans trois voix :

```
\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 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 89).

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.

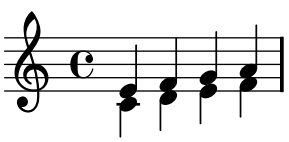
En particulier, la construction `<< \\ >>` que nous avons vue précédemment :

```
\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 } }
>>
```

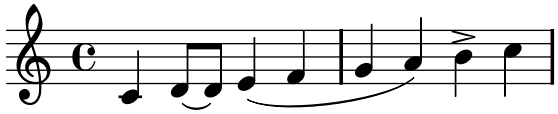
Toutes deux produiront



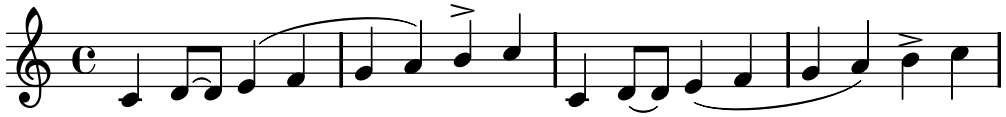
Les commandes `\voiceXXX` fixent la direction des hampes, des liaisons de prolongation et de phrasé, des articulations, des annotations, des points d'augmentation des notes pointées et des doigtés. `\voiceOne` et `\voiceThree` font pointer ces objets vers le haut, alors que `\voiceTwo` et `\voiceFour` 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.

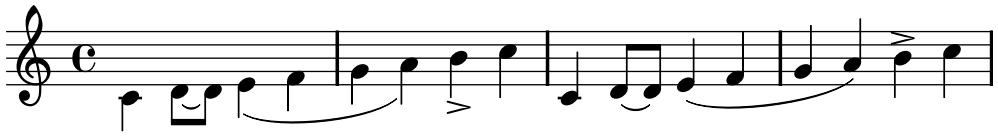
```
\relative c'{
  % Comportement par défaut ou après \oneVoice
  c d8 ~ d e4 ( f g a ) b-> c
}
```



```
\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
}
```

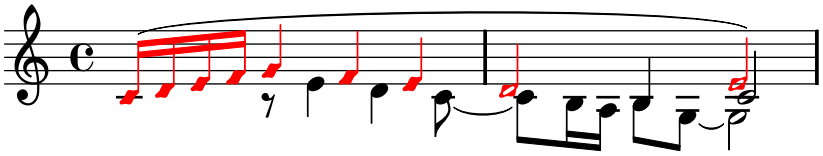


Voyons à présent trois différentes façons d'exprimer un passage polyhonique, à partir d'un exemple de la section précédente. Chacune d'elles aura ses avantages selon les circonstances.

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.

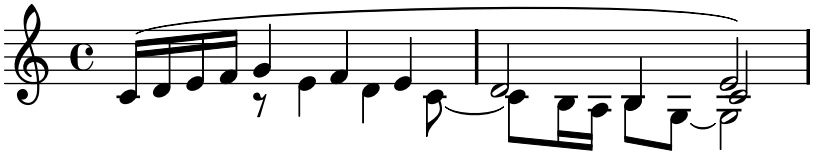
```
\new Staff \relative c' {
  \voiceOneStyle
  % Les notes qui suivent sont monophoniques
  c16^( d e f
  % Début d'une section de trois voix simultanées
  <<
    % Poursuite de la voix principale en parallèle
    { g4 f e | d2 e2) }
    % Initialisation de la seconde voix
    \new Voice {
      % 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 :

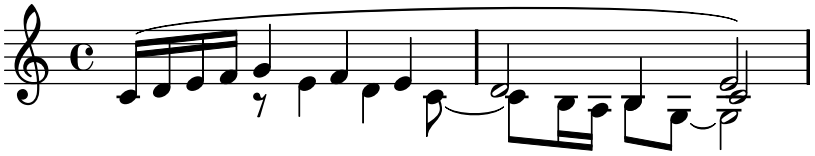
```
\new Staff \relative c' {
  c16^( d e f
  <<
    { g4 f e | d2 e2) }
    \new Voice {
      \voiceTwo
      r8 e4 d c8 ~ |
      <<
        {c8 b16 a b8 g ~ g2}
        \new Voice {
          \voiceThree
          s4 b4 c2
        }
      >>
    }
  >>
}
```



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 au 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 Real music example Section 4.5.3 [Exemple concret], page 90.

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

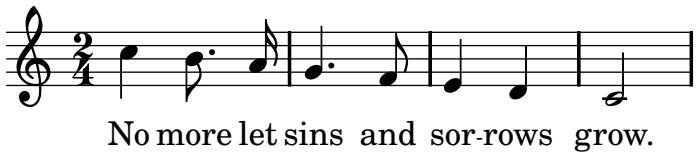
Manuel de notation : Section “Plusieurs voix” dans *Manuel de notation*.

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 grace à la commande `\lyricsto{}` et au nom assigné à la voix en question.

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



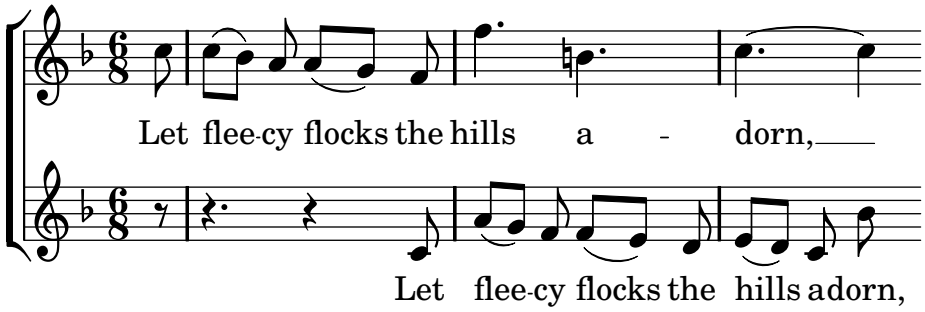
Notez bien que les paroles sont liées à un contexte de voix (`Voice`), *non* à un contexte de portée (`Staff`). Il est donc nécessaire de créer explicitement les contextes `Staff` et `Voice`.

Si la ligature automatique que LilyPond applique par défaut est pleinement adaptée en matière de musique instrumentale, il n'en va pas de même dans le cas d'une musique associée à des paroles, et pour laquelle soit les ligatures sont carrément absentes, soit elles servent à indiquer un mélisme – plusieurs notes pour une même syllabe. Dans l'exemple qui suit, nous utilisons la commande `\autoBeamOff` afin de désactiver les ligatures automatiques.

Nous allons reprendre un extrait de Judas Maccabæus pour illustrer ce que cette technique apporte en flexibilité. Nous commençons par utiliser des variables afin de séparer aussi bien la musique que les paroles, de la structure d'une portée. 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.

```
glogal = { \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 = "SopOne" {
        \glogal
        \SopUnMusique
      }
```

```
      \new Lyrics \lyricsto "SopOne" {
        \SopUnParoles
      }
    >>
    \new Staff <<
      \new Voice = "SopTwo" {
        \glogal
        \SopDeuxMusique
      }
      \new Lyrics \lyricsto "SopTwo" {
        \SopDeuxParoles
      }
    >>
  >>
}
```

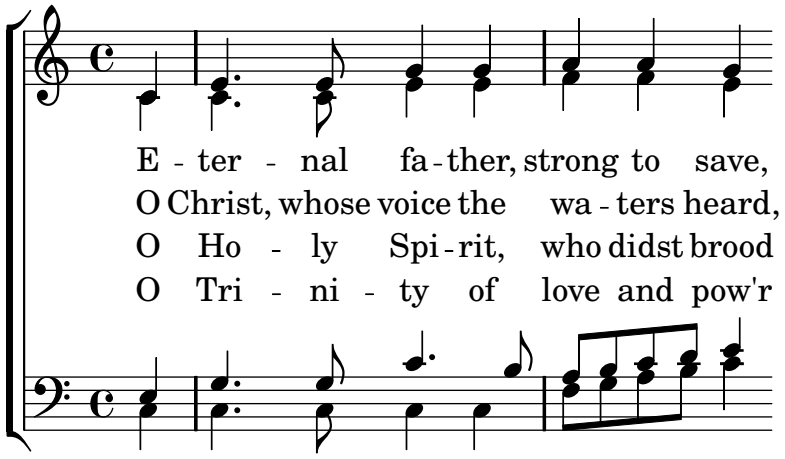


Voici donc la structure de base valable pour toute partition vocale. On peut y ajouter d'autres portées si besoin est, d'autres voix à chaque portée, plusieurs couplets aux paroles, et les variables contenant la musique peuvent même être stockées dans des fichiers indépendants dès lors que leur longueur 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 les notes et les paroles, de la structure de portée. 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 = "Bass"  { \voiceTwo \BasseMusique  }
    >>
  >>
}
```

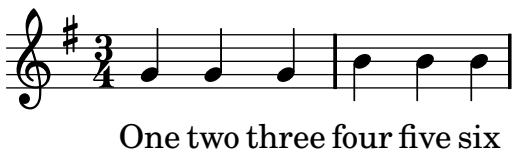


Nous allons terminer en voyant comment coder un couplet pour soliste suivi d'un refrain à deux voix sur deux portées. Les explications sont importantes, dans la mesure où les moyens mis en œuvre pour arriver à enchaîner le solo et la polyphonie dans une seule et même partition sont quelque peu tirés 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 inférieur pour synchroniser les portées ; mais comme nous reportons le parallélisme après le solo, nous utilisons des accolades – un double inférieur 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 inférieur/supérieur 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 = "verse" {
        \coupletnotes \break
      }
      \new Lyrics \lyricsto verse {
        \coupletparoles
      }
    >>
  }
}
```



Voici la ligne du couplet réalisée.

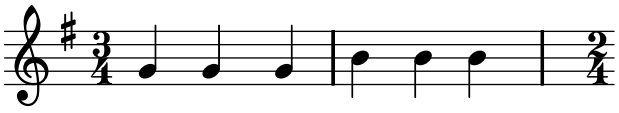
Nous poursuivons avec refrainA, sur la même portée, alors même qu'une deuxième portée s'amorce en parallèle pour contenir refrainB. Cette section parallèle doit s'enchaîner directement à la suite du `\break` de la voix contenant le couplet – il s'agit bien de la *même* voix. Voici cette section parallèle.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

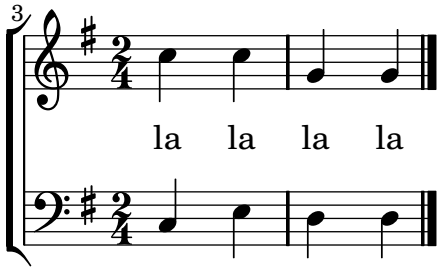
```
<<
  \refrainnotesA
  \new Lyrics \lyricsto verse {
    \refrainwordsA
  }
  \new Staff <<
    \new Voice = "refrainB" {
      \refrainnotesB
    }
    \new Lyrics \lyricsto "refrainB" {
      \refrainwordsB
    }
  >>
>>
```

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 = "verse" {
        \coupletnotes \break
        <<
          \refrainnotesA
          \new Lyrics \lyricsto "verse" {
            \refrainparolesA
          }
          \new Staff <<
            \new Voice = "refrainB" {
              \refrainnotesB
            }
            \new Lyrics \lyricsto "refrainB" {
              \refrainparolesB
            }
          >>
        >>
      }
      \new Lyrics \lyricsto "verse" {
        \coupletparoles
      }
    >>
  }
}
```

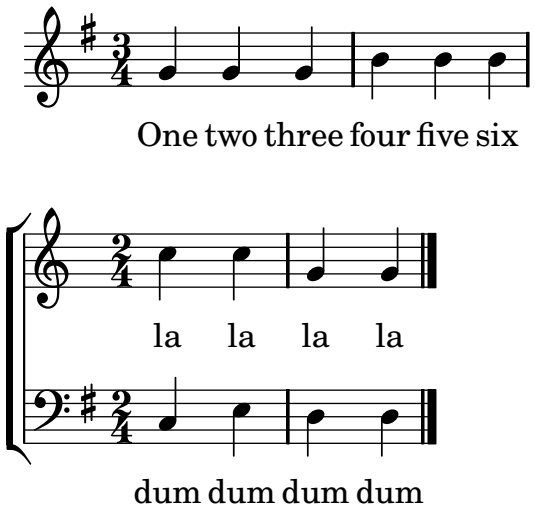




Bien que ce que nous venons de voir constitue un exercice intéressant et fort utile pour comprendre comment s'articulent des blocs séquentiels et simultanés, nous aurions aussi pu coder notre exemple sous la forme de deux blocs `\score` au sein d'un bloc `\book` implicite :

```
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 = "verse" {
      \coupletnotes
    }
    \new Lyrics \lyricsto "verse" {
      \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*.

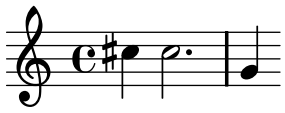
3.3 Contextes et graveurs

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 :

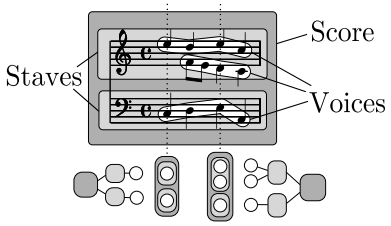
```
cis4 cis2. g4
```



Si le code est assez austère, dans la parition 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 mélant 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 gèrer 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 chifrée, etc.

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

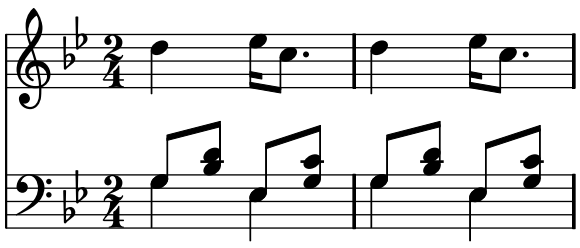
```
\new type expression-musicale
```

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 comment toute déclaration qui ouvre un bloc par une accolade, `{`, ou un double signe inférieur, `<<`, 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 pairées ». 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 inférieur double – 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 musiocal 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 grace 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`.

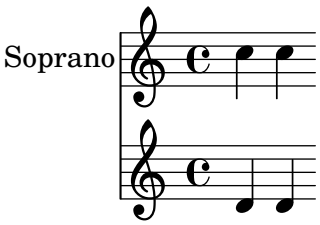
Les noms des propriétés de contexte sont composés de mots accolés sans trait d'union ni caractère souligné, et dont seul le premier n'aura pas d'initiale en majuscule. Voici quelques exemples de celles les plus communément utilisées.

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 dièses (`#`) dans deux cas particuliers : il 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 un double dièse – par exemple `##t`.

Avant de déterminer l'une de ces propriétés, nous devons savoir dans quel contexte elles interviennent. 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
 }
>>
```



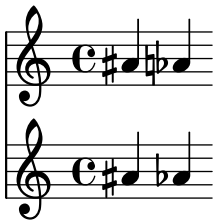
Dans la mesure où le nom de contexte par défaut est `Voice`, la deuxième commande `\set` a défini « Alto » comme propriété `instrumentName` du contexte de voix. Puisque LilyPond n'ira pas chercher une telle propriété dans la contexte `Voice`, celle-ci ne sera pas interpétée. Il ne s'agit pas d'une erreur,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

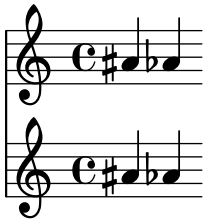
Voici comment supprimer les bécarres supplémentaires pour une portée :

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

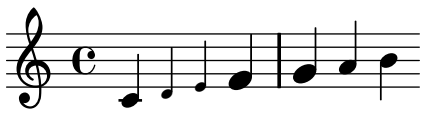


Autre exemple, si la propriété `clefOctavation` est déterminée au niveau du contexte `Score`, elle modifiera la valeur de l'octave en cours pour toutes les portées actives ; cette valeur sera considérée comme étant la nouvelle valeur par défaut pour toutes les portées à venir.

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 affectée par un `\unset` ou un nouveau `\set`. À titre d'exemple, nous allons modifier jouer avec la taille des fontes, ce qui affecte entre autres la grosseur des tes 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 propriétés. N'oubliez pas que les nombres, entiers ou réels, doivent être précédés d'un dièse (`#`) et les valeurs vrai ou

faux de deux dièses – 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.

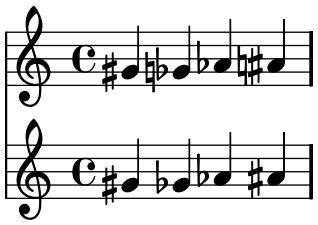
Définition des propriétés de contexte avec \with

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 voulons 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 grace à \set ; un \unset les ramènera à leur valeur par défaut.

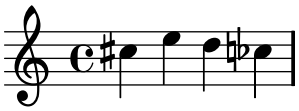
La propriété fontSize constitue une exception : lorsqu'elle est déterminée au sein d'un bloc \with, cela redéfinit la valeur par défaut de la taille de fonte. Une modification est possible par la commande \set, mais la commande \unset fontSize fera revenir à la nouvelle valeur par défaut.

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, prédédé 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 contextes ainsi définies peuvent être adaptées pour chacun des contextes en particulier grace 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 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 chacun des différents contextes contient plusieurs graveurs, et que chacun de ces graveurs est chargé de générer une part spécifique du résultat, qui les barres de mesure, qui la portée, qui les têtes de note, les hampes, etc. Le fait de suprimer un graveur d’un contexte éliminera sa contribution à l’œuvre résultante. 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 propose en reprenant un exemple du chapitre précédant, puor lui suprimer les lignes de la portée. Pour mémoire, les lignes d’une portée sont générées par le Staff_symbol_engraver.

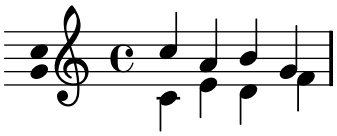
```
\new Staff \with {
  \remove Staff_symbol_engraver
}
\relative c' {
  c4
  \set fontSize = #-4  % pour obtenir des têtes de note plus petites
  d e
  \set fontSize = #2.5  % pour obtenir des têtes de note plus grosses
  f g
  \unset fontSize  % retour à la taille par défaut
  a b
}
```

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

`\consists` *`Nom_du_graveur`*

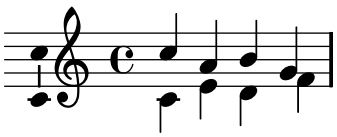
et se place dans un bloc `\with`. Certianes partitions vocales font apparaître un Section “ambitus” dans *Glossaire* au début de la portée, afin d'indiquer ses notes extrèmes. L'ambitus est généré par l'`Ambitus_engraver`, que l'on peut adjoindre à n'importe quel contexte. Si nous l'ajoutons au contexte `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 :

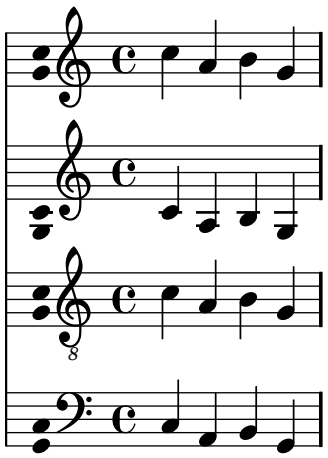
```
\new Staff \with {
    \consists Ambitus_engraver
  }
  <<
  \new Voice
  \relative c'' {
    \voiceOne
    c a b g
  }
  \new Voice
  \relative c' {
    \voiceTwo
    c e d f
  }
>>
```



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 suprimer 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 104, 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 l’on pourrait commencer par les « notes et paroles », pour la partie de soprano.

```
\version "2.11.58"
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’oeil sur l’exemple avec les notes seules :

```
\version "2.11.58"
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 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 "2.11.58"
sopranoMusique = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

sopranoParoles = \lyricmode {
  Laaa Siii Dooo Réé́é
}

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
```

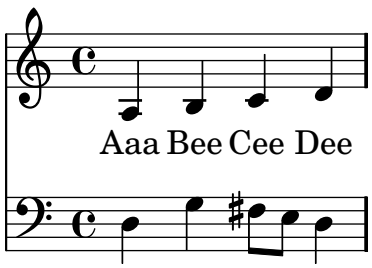
en dessous de tout ce qui concerne la soprano. Il nous faut également encadrer la musique par des `<<` et `>>`, qui feront comprendre à LilyPond que plusieurs évènements — ici, des objets `Staff` — se déroulent en même temps. Le bloc `\score` ressemble maintenant à

```
\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 "2.11.58"

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 = "one" {
        \autoBeamOff
        \sopranoMusique
```

```
        }
        \new Lyrics \lyricsto "one" \sopranoParoles
      >>
      \new Staff \violoncelleMusique
    >>
    \layout { }
    \midi { }
}
```

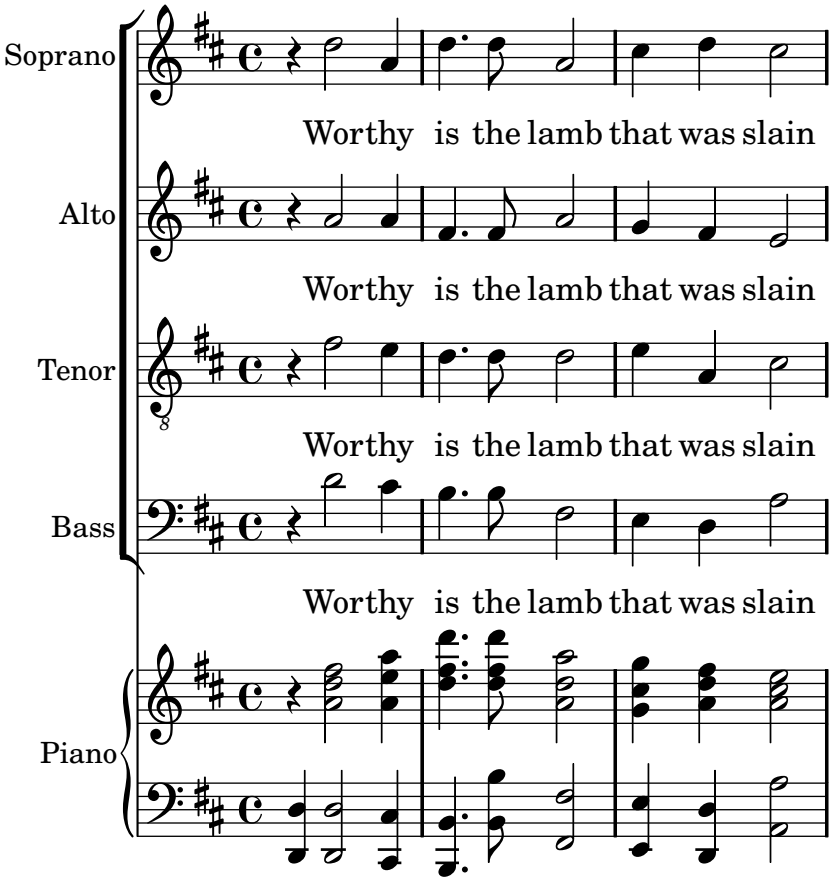


Voir aussi

Les patrons originaux sont disponibles à l'annexe « Modèles », voir Section A.1 [Portée unique], page 104.

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 104 – mais encore faudrait-t-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 postition 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" { \sopranoParoless }
  \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 simplement 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 `supérieur` et `inferieur`.

Les systèmes pour chœur et pour piano doivent être combinés à l’aide de doubles inférieur/supérieur puisqu’ils doivent s’empiler :

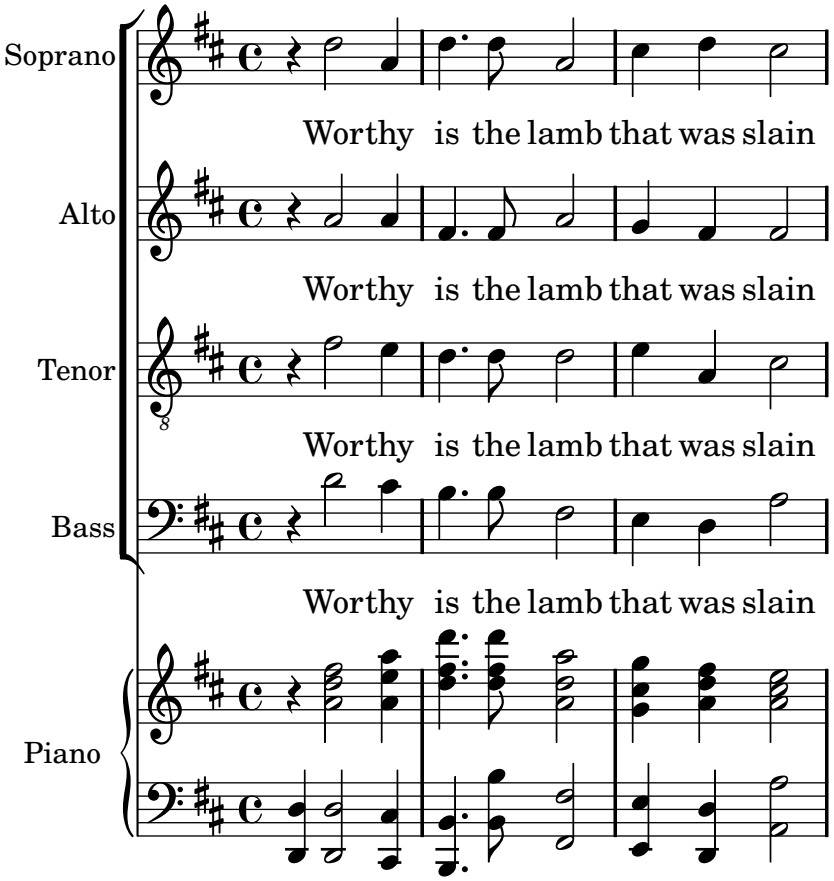
```
<<  % combine ChoirStaff and PianoStaff one above the other
  \new ChoirStaff <<
    \new Staff = "sopranos" <<
      \new Voice = "sopranos" { \global \sopranoMusique }
    >>
    \new Lyrics \lyricsto "sopranos" { \sopranoParoless }
    \new Staff = "altos" <<
      \new Voice = "altos" { \global \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 obtenon :

```
\version "2.11.58"
glogal = { \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
bassMusique = \relative c' {
  \clef "bass"
  r4 d2 cis4 | b4. b8 fis2 | e4 d a'2 |
}
bassParoles = \sopranoParoles
superieur = \relative a' {
  \clef "treble"
  \glogal
  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"
  \glogal
  <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 = "sopranos" <<
        \set Staff.instrumentName = "Soprano"
        \new Voice = "sopranos" { \glogal \sopranoMusique }
      >>
      \new Lyrics \lyricsto "sopranos" { \sopranoParoles }
      \new Staff = "altos" <<
        \set Staff.instrumentName = "Alto"
        \new Voice = "altos" { \glogal \altoMusique }
      >>
      \new Lyrics \lyricsto "altos" { \altoParoles }
      \new Staff = "tenors" <<
        \set Staff.instrumentName = "Tenor"
```

```
        \new Voice = "tenors" { \glogal \tenorMusique }
      >>
      \new Lyrics \lyricsto "tenors" { \tenorParoles }
      \new Staff = "basses" <<
        \set Staff.instrumentName = "Bass"
        \new Voice = "basses" { \glogal \bassMusique }
      >>
      \new Lyrics \lyricsto "basses" { \bassParoles }
    >>  % fin du ChoirStaff (système pour chœur)

    \new PianoStaff <<
      \set PianoStaff.instrumentName = "Piano  "
      \new Staff = "upper" \superieur
      \new Staff = "lower" \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 pret 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 treminons 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 "2.11.58"
\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 = "ManualOne" <<
    \new Voice { \ManuelUnVoixUnMusique }
    \new Voice { \ManuelUnVoixDeuxMusique }
  >>  % fin du contexte de portée ManuelUn
  \new Staff = "ManualTwo" <<
    \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 doît rester synchrone avec lui, nous utilisons un double inférieur/supérieur 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 savoir de quoi il retourne.

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

```
    >>  % fin du contexte de portée ManuelDeux
  >>  % fin du contexte PianoStaff
  \new Staff = "PedalOrgan" <<
    \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 un double inférieur/supérieur ce qui suit une sommande `\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 avez 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 grace à `\voiceOne` et `\voiceTwo`, et mettre en place la métrique et l'armure de chaque portée grace à notre variable `\MetriqueArmure`.

```
\score {
  <<  % Système pianistique et portée de pédalier sont synchrones
    \new PianoStaff <<
      \new Staff = "ManualOne" <<
        \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 = "ManualTwo" <<
        \TimeKey
        \clef "bass"
        \new Voice { \ManuelDeuxMusique }
      >>  % fin du contexte de la portée ManuelDeux
    >>  % fin du contexte PianoStaff
    \new Staff = "PedalOrgan" <<
      \TimeKey
      \clef "bass"
      \new Voice { \PedalierOrgueMusique }
    >>  % fin du contexte de la portée PedalOrgan
  >>
}  % fin du contexte Score
```

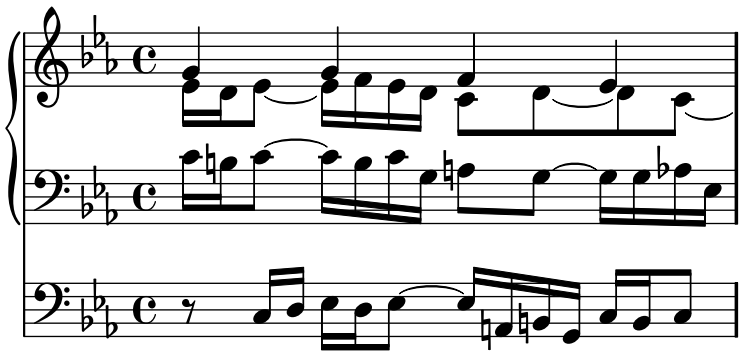
Nous en avons fini avec la structure. Toutes les partitions pour orgue auront cette structure, même si le nombre de voix peut chager. Tout ce qui nous reste à faire maintenant consiste à saisir la musique et à regrouper toutes les parties.

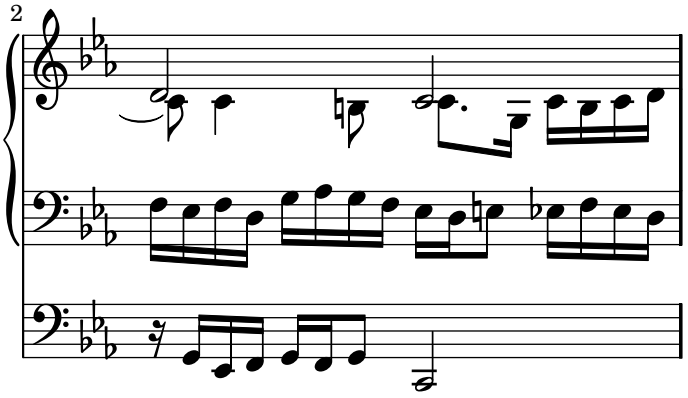
```
\version "2.11.58"

\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
MetriqueArmure = { \time 4/4 \key c \minor }
ManuelUnVoixUnMusique = \relative g' {
  g4 g f ees | d2 c2 |
}
```

```
ManuelUnVoixDeuxMusique = \relative c' {
  ees16 d ees8~ ees16 f ees d c8 d~ d c~ |
  c c4 b8 c8. g16 c b c d |
}
ManuelDeuxMusique = \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 |
}
PedalierOrgueMusique = \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 sont synchrones
    \new PianoStaff <<
      \new Staff = "ManualOne" <<
        \MetriqueArmure  % 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 = "ManualTwo" <<
        \MetriqueArmure
        \clef "bass"
        \new Voice { \ManuelDeuxMusique }
      >>  % fin du contexte de la portée ManuelDeux
    >>  % fin du contexte PianoStaff
    \new Staff = "PedalOrgan" <<
      \MetriqueArmure
      \clef "bass"
      \new Voice { \PedalierOrgueMusique }
    >>  % fin du contexte de la portée PedalOrgan
  >>
}  % fin du contexte Score
```

Jesu, meine Freude

J S Bach




2

4 Retouche des partitions

Ce chapitre indique comment modifier le résultat que vous obtiendrez. LilyPond offre de nombreuses possibilités de réglages, permettant de modifier quasiment chaque élément de votre partition.

4.1 Retouches élémentaires

4.1.1 Introduction aux retouches

4.1.2 Objets et interfaces

4.1.3 Conventions de nom des objets et propriétés

4.1.4 Méthodes de retouche

4.2 Le manuel des références internes

4.2.1 Propriétés des objets de rendu

4.2.2 Propriétés listées par interface

4.2.3 Types de propriétés

4.3 Apparence des objets

4.3.1 Visibilité et couleur des objets

4.3.2 Taille des objets

4.3.3 Longueur et épaisseur des objets

4.4 Positionnement des objets

4.4.1 Comportement automatique

4.4.2 Objets inclus dans la portée

4.4.3 Objets hors de la portée

4.5 Collisions d'objets

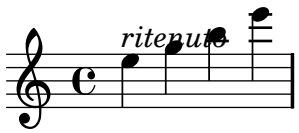
4.5.1 Déplacement d'objets

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.

À FAIRE : les modifications de la gestion des espacements de la version 2.12 feront perdre leur pertinence aux exemples suivants. Ils démontrent cependant la puissance de LilyPond, et justifient à ce titre leur présence dans ces lignes, tant que d'autres exemples n'auront pas été proposés.

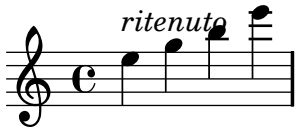
```
% temporary code to break this example:
\override TextScript #'outside-staff-priority = ##f
```

```
e4^\markup{ \italic ritenuto } g b e
```



Le plus simple est ici d'augmenter la distance entre l'objet (du texte comme ici, ou bien des nuances ou des doigtés) et la note. Dans LilyPond, il s'agit de la propriété `padding`, qui se mesure en espaces relatifs à la taille de la portée. Pour la plupart des objets (chacun ayant sa propre valeur), elle est définie à 1.0, ou un peu moins. Nous voulons ici l'augmenter : essayons 1.5.

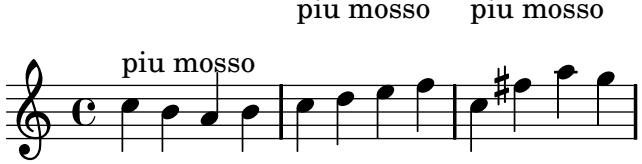
```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'padding = #1.5
e4^\markup{ \italic ritenuto } g b e
```

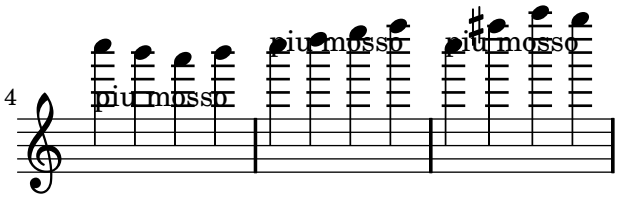


C'est déjà mieux ! Mais on peut certainement encore améliorer le résultat. Il nous semble, après plusieurs essais, que la meilleure valeur dans ce cas soit 2.3. Toutefois, ce constat est le fruit d'expérimentations et de goût personnel en matière de notation. Essayez le même exemple avec 2.3... mais également avec des valeurs plus grandes (ou plus petites). À votre avis, quelle est la meilleure version ?

La propriété `staff-padding` est de nature similaire. `padding` détermine l'espace minimum entre un objet et l'objet le plus proche (le plus souvent une note ou les lignes de la portée) ; `staff-padding` détermine pour sa part l'espace minimum entre un objet et la portée. La différence est subtile, mais vous apparaîtra clairement ici :

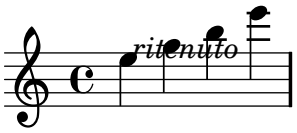
```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
\break
c'4^"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
```





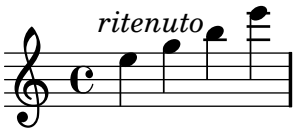
Une autre démarche permet de contrôler totalement la position d'un objet — on peut le déplacer horizontalement ou verticalement. Il suffit d'avoir recours à la propriété `extra-offset`. En fait c'est une méthode plus complexe, qui peut en outre poser des problèmes. Quand on déplace un objet à l'aide de `extra-offset`, le déplacement est effectué après que LilyPond a placé tous les autres objets. Par conséquent, l'objet ainsi déplacé peut venir recouvrir d'autres objets déjà placés.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( 1.0 . -1.0 )
e4^\markup{ \italic ritenuto } g b e
```



Lorsqu'on utilise `extra-offset`, le premier nombre décrit le déplacement horizontal (négatif pour un déplacement vers la gauche) tandis que le deuxième décrit un déplacement vertical (positif pour le haut). Après quelques essais, on peut choisir les valeurs suivantes qui semblent donner un résultat satisfaisant.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( -1.6 . 1.0 )
e4^\markup{ \italic ritenuto } g b e
```



Une fois encore, c'est après quelques tâtonnements que l'on a abouti à ces nombres, au regard du résultat final. Si vous souhaitez que le texte soit plus haut, plus à gauche, etc. essayez vous-même et choisissez aprés avoir regardé le résultat.

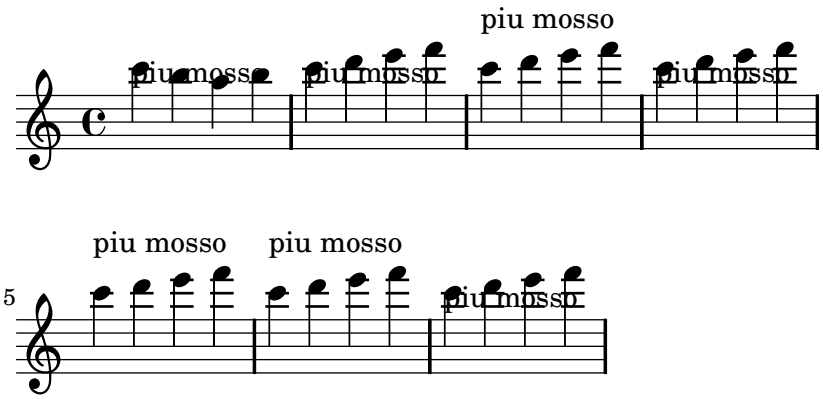
Une dernière mise en garde : dans cette section, nous avons eu recours à

```
\once \override TextScript ...
```

ce qui permet de régler le placement du texte pour la note suivante. Mais si cette note n'a pas de texte, le réglage ne s'appliquera pas et n'attendra **pas** le prochain texte. Pour que ce comportement persiste après la commande, ne mettez pas `\once`. Votre réglage s'appliquera alors partout, jusqu'à ce que vous l'annuliez au moyen de la commande `\revert`. Ceci est expliqué en détail dans Section "The \override command" dans *Manuel de notation*.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b
\once \override TextScript #'padding = #4.6
  a4 b
c4^"piu mosso" d e f
\once \override TextScript #'padding = #4.6
```

```
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\break
\override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\revert TextScript #'padding
c4^"piu mosso" d e f
```



Voir aussi

Dans ce même manuel : Section “The \override command” dans *Manuel de notation*, Section 4.6 [Retouches courantes], page 90.

4.5.2 Correction des collisions d’objets

Dans la section Section 4.5.1 [Déplacement d’objets], page 86, nous avons vu comment déplacer un objet `TextScript`. Ce même procédé peut être appliqué à d’autres types d’objet : il vous suffira de remplacer `TextScript` par le nom de l’objet en question.

Pour trouver cette dénomination, regardez les liens ‘**Voir aussi**’ en bas des pages de la documentation. Par exemple, en bas de la page Section “Nuances” dans *Manuel de notation*, nous trouvons

Voir aussi

Référence du programme : Section “DynamicText” dans *Référence des propriétés internes*, Section “Hairpin” dans *Référence des propriétés internes*. Le placement vertical de ces symboles est contrôlé par Section “DynamicLineSpanner” dans *Référence des propriétés internes*.

Ce qui implique que, pour modifier la hauteur d’une nuance, nous utiliserons

```
\override DynamicLineSpanner #'padding = #2.0
```

Nous ne listerons pas ici tous les types d’objets, mais seulement les plus communs :

Type d’objet	**Nom de l’objet**
Nuances (verticalement)	`DynamicLineSpanner`
Nuances (horizontalement)	`DynamicText`
Laisons de tenue	`Tie`

Liaisons	`Slur`
Indications d'articulation	`Script`
Doigtés	`Fingering`
Textes (`^"texte"`)	`TextScript`
Repères	`RehearsalMark`

4.5.3 Exemple concret

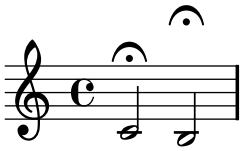
4.6 Retouches courantes

Certains réglages sont si courants que des raccourcis sont fournis sous forme de commandes telles que `\slurUp` ou `\stemDown`. Toutes ces commandes sont décrites dans les différentes sections de la Référence de notation.

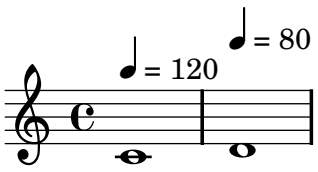
La liste complète des modifications possibles pour chaque type d'objet (tel que liaison ou ligature) se trouve dans la Référence du programme. Cependant, certaines propriétés sont communes à de nombreux objets, et on peut de ce fait définir quelques réglages génériques.

- La propriété `padding` peut être définie de manière à accroître (ou décroître) la distance entre les symboles qui se placent au-dessus ou au-dessous des notes. Ce qui s'applique à tous les objets régis par `side-position-interface`.

```
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
```

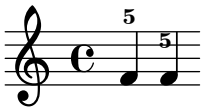


Notez, dans le second exemple, l'importance de savoir à quel contexte correspond l'objet. Dans la mesure où l'objet `MetronomeMark` appartient au contexte `Score`, ses modifications affectées au contexte `Voice` ne l'affecteront pas. Pour plus de détails, voir Section "Élaboration d'une retouche" dans *Manuel de notation*.

- La propriété `extra-offset` permet de déplacer latéralement et verticalement ; c'est pourquoi elle requiert deux nombres. Le premier affecte le placement horizontal (un nombre positif déplace l'objet vers la droite) ; le second le placement vertical (un nombre positif déplace l'objet vers le haut). Cette propriété est de bas niveau : le moteur de formatage ne tient aucun compte des placements qu'elle induit.

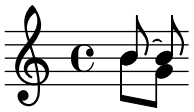
Dans l'exemple suivant, le second doigté est déplacé un peu vers la gauche, et plus bas de 1.8 espaces :

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



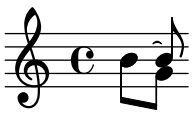
- La propriété `transparent` imprime les objets avec de l'"encre invisible' : l'objet n'est pas visible, mais tous les comportement le concernant s'appliquent quand même. Il occupe une certaine place, intervient dans la gestion des collisions, et on peut lui attacher des liaisons ou des ligatures.

  L'exemple suivant montre comment tenir des notes entre différentes voix, au moyen de liaisons. Ces liaisons de tenue, en principe, ne peuvent relier que deux notes d'une même voix. On introduit donc la liaison dans une autre voix :



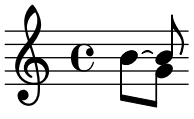
  et on efface la première croche (hampe vers le haut) de ladite voix ; maintenant la liaison semble passer d'une voix à l'autre :

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



  Pour s'assurer que le crochet de la hampe que nous avons effacée ne raccourcira pas la liaison, nous allons également rallonger cette hampe, en attribuant à la propriété `length` la valeur `8` :

```
<< {
  \once \override Stem #'transparent = ##t
  \once \override Stem #'length = #8
  b8~ b8\noBeam
} \\ {
  b[ g8]
} >>
```



Les distances dans LilyPond sont mesurées dans l'unité staff-space (espace de portée) tandis que la plupart des propriétés relatives aux épaisseurs sont mesurées à l'aide de l'unité line-thickness (épaiseur de ligne). Toutefois, certaines d'entre-elles échappent à cette règle : par exemple l'épaisseur des liens de croches est mesurée à l'aide de l'unité staff-space. Pour de plus amples informations, consultez les sections correspondantes de la Référence du programme.

4.7 Autres retouches

4.7.1 Autres utilisations des retouches

4.7.2 Utilisation de variables dans les retouches

4.7.3 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.

Des réglages par défaut (tels que les définitions des blocs `\header{}`) sont contenus dans des fichiers `.ly`. D'autres (comme les définitions des commandes « markup ») sont contenus dans des fichiers `.scm` (Scheme). Malheureusement, des explications plus complètes dépassent le cadre de ce manuel. 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.

- Linux : '*dossierduprogramme*`/lilypond/usr/share/lilypond/current/`'
- Mac OS X : '*dossierduprogramme*`/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 : '*dossierduprogramme*`/LilyPond/usr/share/lilypond/current/`'

Les répertoires '`ly/`' et '`scm/`' sont tout particulièrement intéressants. En effet les fichiers du type '`ly/property-init.ly`' ou encore '`ly/declarations-init.ly`' déterminent toutes les définitions avancées communes.

4.7.4 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.

```
%%  Ceci sert à s'assurer que les indications textuelles resteront à l'intérieur des ma
\override Score.PaperColumn #'keep-inside-line = ##t
```

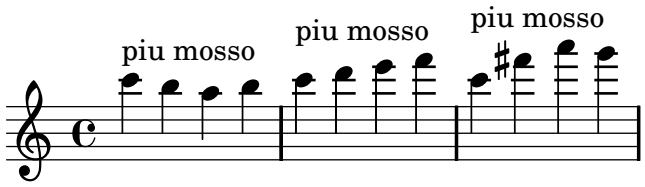
4.7.5 Retouches avancées avec Scheme

Nous avons déjà vu comment le résultat obtenu avec LilyPond peut être largement personnalisé à l'aide de commandes comme `\override TextScript #'extra-offset = ( 1 . -1)`. Cependant, l'utilisation de Scheme ouvre des possibilités encore plus grandes. Pour des explications complètes là-dessus, consultez le Annexe B [Tutoriel Scheme], page 106 et les Section "Interfaces pour les programmeurs" dans *Manuel de notation*.

On peut utiliser Scheme simplement à l'aide des commandes `\override`.

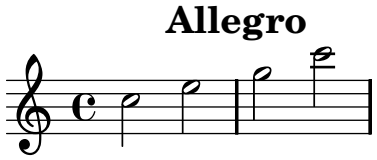
```
decallageTexte = #(define-music-function (parser location padding) (number?)
#{
  \once \override TextScript #'padding = #$padding
#})
```

```
\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
}
```



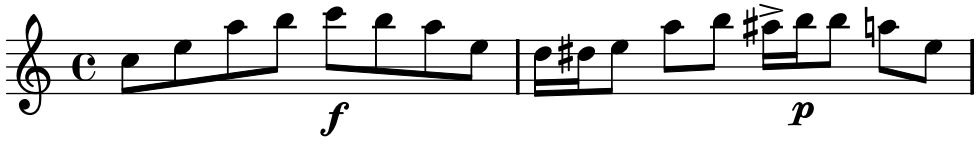
On peut s'en servir pour créer de nouvelles commandes :

```
tempoMarque = #(define-music-function (parser location padding marktext)
                                      (number? string?)
#{
  \once \override Score . RehearsalMark #'padding = $padding
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $marktext }
#})

\relative c'' {
  c2 e
  \tempoMarque #3.0 #"Allegro"
  g c
}
```



On peut même y inclure des expressions musicales :

```
motif = #(define-music-function (parser location x y) (ly:music? ly:music?)
#{
  $x e8 a b $y b a e
#})

\relative c''{
  \motif c8 c8\f
  \motif {d16 dis} { ais16-> b\p }
}
```



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 de saisie des 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 [Économies de saisie grâce aux identificateurs et fonctions], page 95 et Section 5.1.5 [Feuilles de style], page 97.

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 94, 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`.

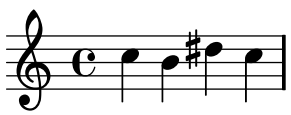
```
fthenp = _\markup{
 \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
g4\fthenp c'8. e16
}
```

5.1.4 Économies de saisie grâce aux identificateurs et fonctions

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

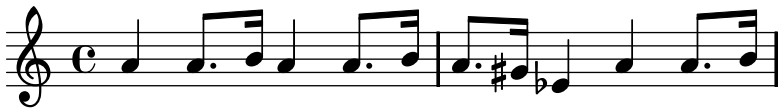
```
corNotes = \relative c'' { c4 b dis c }
\score {
  {
```

```
    \corNotes
  }
}
```



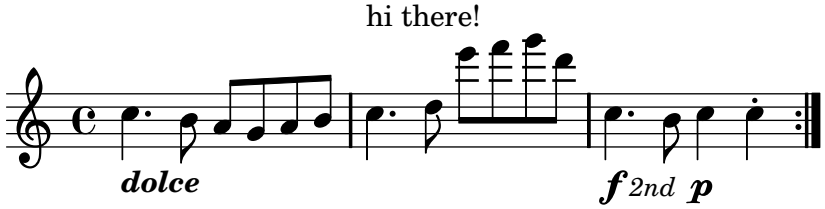
Vous comprendrez combien cela peut être utile pour écrire de la musique minimaliste :

```
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}
}
```



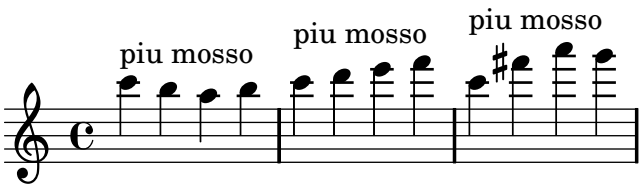
Ces identificateurs sont évidemment utiles pour économiser de la frappe. Mais ils peuvent l'être même si vous ne les utilisez qu'une seule fois : ils réduisent la complexité. Regardons l'exemple précédent sans aucun identificateur. C'est beaucoup plus laborieux à lire, et particulièrement la dernière 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 101. Si vous avez une seule définition, par exemple `\dolce`, pour tous vos fichiers (voir Section 5.1.5 [Feuilles de style], page 97), 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`.

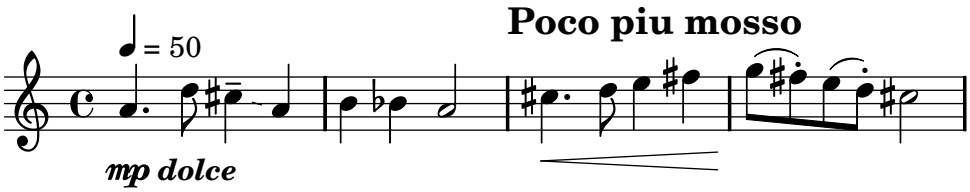
5.1.5 Feuilles de style

La sortie que produit LilyPond peut être largement modifiée — voir Chapitre 4 [Retouche des partitions], page 86 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 4.7.5 [Retouches avancées avec Scheme], page 92.

```
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
}
```

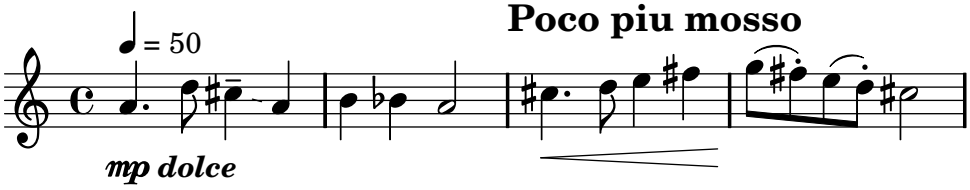


Il y a quelques problèmes de chevauchement ; nous allons arranger cela en utilisant les techniques de Section 4.5.1 [Déplacement d'objets], page 86. On peut aussi faire quelque chose pour les définitions de `mpdolce` et `tempoMark`. Elles produisent le résultat que nous désirons, mais nous pourrions 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"`').

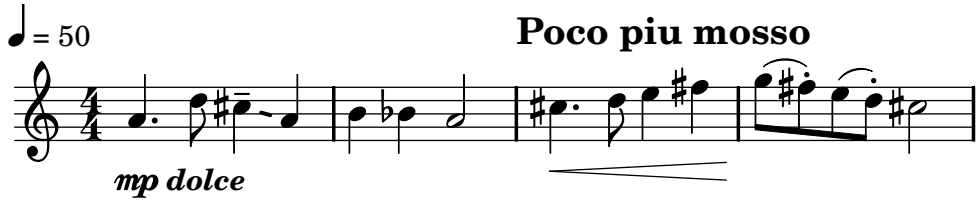
```
\include "definitions.ly"

\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 :

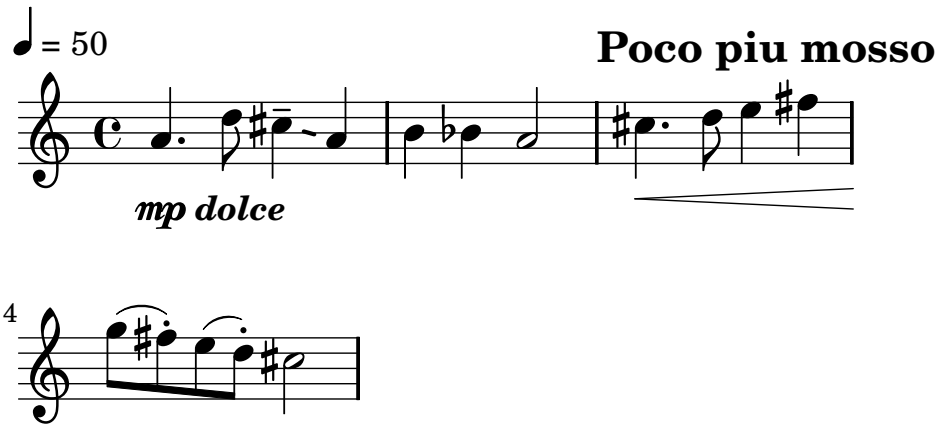
```
%%%  definitions.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override 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 "2.11.58"
#(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 minimaux], page 102.

5.2.3 Exemples minimaux

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
}
```

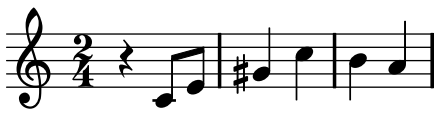
On établira alors une partie séparée en constituant un nouveau fichier :

```
\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 :



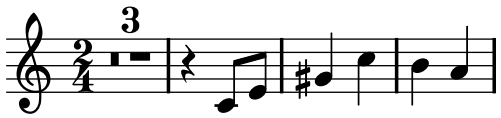
Dans les pièces d’ensemble, il arrive souvent qu’une voix ne joue pas pendant plusieurs mesures. Un silence spécial, appelé silence multi-mesures, l’indique alors. On l’obtient par un ‘`R`’ majuscule, suivi d’une durée : `1` pour une pause, `2` pour une demi-pause, etc. Cette durée peut être multipliée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

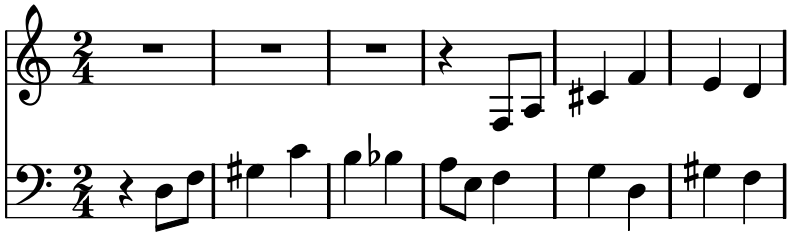
Cette commande assigne la valeur ‘vrai’ — ‘true’ en anglais, et ‘`#t`’ dans le langage Scheme — à cette propriété dans le contexte `Score`. Si l’on ajoute dans la musique ci-dessus le silence multi-mesures et cette option, on obtient le résultat suivant :



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 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

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

<

<< \\ >> 49

\

\\ 49
\autoBeamOff 57
\book 41, 62
\consists 71
\header 42
\layout 42, 73
\lyricmode 57
\lyricsto 56
\midi 42
\new 64
\new ChoirStaff 57
\new Lyrics 56
\new Voice 53
\oneVoice 53
\remove 71
\score 41, 43
\set 67
\shiftOff 56
\shiftOn 56
\shiftOnn 56
\shiftOnnn 56
\unset 67
\voiceFour 53
\voiceFourStyle 51
\voiceNeutralStyle 51
\voiceOne 53
\voiceOneStyle 51
\voiceThree 53
\voiceThreeStyle 51
\voiceTwo 53
\voiceTwoStyle 51
\with 70

A

accents 22
acciaccature 25
accolade 29
accord 30
accords, notes simultanées 30
Aide-mémoire 10
Ajout de texte 24
altération accidentelle 14, 20
Altérations 21
altérations à l'armure 20
Altérations accidentelles automatiques 21
ambitus 72
ambitus, graveur 72
anacrouse 25
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

B

barre de ligature 16, 24
Barres de ligature automatiques 25
Barres de ligature manuelles 25
bécarre 20
bémol 20
Bibliographie 10
blanche 16
bloc de commentaire 18
book, exemple d'utilisation 62
book, livre, ouvrage 41

C

calques (layers) 48
caractère souligné (paroles) 32
casse, prise en compte de 12, 18
chansons 31
clavier, portée pour 29
clef 17
Clefs 18
commentaire 18
commentaire de fin de ligne 18
commentaire-bloc 18
compilation 12
contexte de voix 48
contexte, noms de 66
contexte, propriétés 67
Contextes, création de 64
contextes, les différents 63
Contexts 6, 68, 71
Conversion à partir d'autres formats 10
couplet et refrain 59
Création de fichiers MIDI 42
Création de titres 38
crescendo 23
crochets, imbrication 47

D

décalage, commandes 56
decrescendo 23
dièse 20
distances 92
do central 14
documentation du fonctionnement interne 10
doigté 23
doigtés 23
Doigtés 24
double bémol 20
double dièse 20
durée 16
durée, liaisons de 21
DynamicLineSpanner 89
DynamicText 89

E

Écriture de parties séparées ... 103
Écriture des hauteurs de note ... 18
Écriture des silences ... 18
Écriture du rythme ... 18
Élaboration d'une retouche ... 90
empilement de notes ... 56
en-tête ... 42
engravers ... 66
Engravers and Performers ... 67
Environnement de travail ... 10
épaisseur des caractères ... 2
équilibre ... 2
espacement optique ... 2
espacement régulier ... 3
étendre lilypond ... 10
Exécution de LilyPond ... 10
exemple, SATB ... 77
expression ... 26
expression musicale ... 26
Expression musicale composite ... 43
extra-offset ... 88, 90

F

FDL, GNU Free Documentation License ... 107
fichier PDF ... 12
fonte ... 2
format d'entrée ... 41

G

gamme ... 14
General input and output ... 9
Gestion de l'espace ... 9
graveurs ... 66
graveurs, ajout ... 71
graveurs, suppression ... 71
gravure ... 5
Gravure des portées ... 29

H

Hairpin ... 89
hampes en bas ... 52
hampes en haut ... 52
hauteur ... 14, 20
header ... 42

I

identificateurs ... 43, 95
Ignorer des passages de la partition ... 95
imbrication d'expressions musicales ... 55
imbrication de constructions simultanées ... 55
implicit contexts ... 41
index ... 10
Index de LilyPond ... 10
Index des commandes LilyPond ... 10
Installation ... 10
Instruments à clavier ... 29
Interfaces pour les programmeurs ... 10, 92
intervalle ... 14
invisibles, objets ... 91

J

jargon ... 9

L

La commande set ... 71, 103
langage ... 9
langue ... 9
langues étrangères ... 9
layout ... 42
legato ... 22
levée ... 25
Levées ... 26
liaison d'articulation ... 22
liaison de prolongation ... 21, 22
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 et paroles ... 57
ligatures manuelles ... 24
ligne d'extension ... 32
ligne de prolongation ... 32
LilyPond et les éditeurs de texte ... 12
LilyPond-book ... 10
lilypond-internals ... 10
lire la partition ... 12
livre ... 41
LSR ... 10
Lyrics, création d'un contexte ... 56

M

masquage d'objets ... 91
mélisme ... 32
mesure incomplète ... 25
métrique ... 17
Métrique ... 18
midi ... 42
Mise à jour des fichiers avec convert-ly ... 38, 101
mise en forme ... 42
Mise en forme de la partition ... 42
modèles, création ... 81
modèles, modification des ... 74
Modification des greffons de contexte ... 74
modification des propriétés d'un contexte ... 67
Modification des réglages par défaut d'un contexte ... 71, 74
Modification des réglages prédéfinis ... 10
musique concurrente ... 48
musique simultanée ... 48
Musique vocale ... 35, 36, 63

N

noire ... 16
nolets ... 25
Nolets ... 26

nommage des contextes 66
`Noms de note` 21
`Noms de note dans d'autres langues` 20, 21
`Notation musicale générale` 9, 40
`Notation polymétrique` 29
`Notation spécialisée` 9
note column 56
`note pointée` 16
`notes d'ornement` 25
`Notes d'ornement` 26
`Notes simultanées` 31
nouveaux contextes 64
`nuances` 23
`Nuances` 24, 89

O

`octave` 14
ornementation 25
`ossia` 46
ossias 46

P

padding 87, 90
`paroles` 31
paroles et ligatures 57
paroles, affectation à une voix 56
partition 41, 43
partition, lire 12
PDF 12
phrasé, liaisons de 22
piano, portée pour 29
`Plusieurs partitions dans un même ouvrage` 43
`Plusieurs voix` 53, 56
police 2
`polyphonie` 27
polyphonie 30, 48
portée double 29
portée pour piano 29
portée, positionnement 46
`Portées d'ossia` 47
portées, temporaires 46
prise en compte de la casse 12, 18
prolongation, liaisons de 21
propriétés 10
propriétés d'un contexte, définition avec \with 70
propriétés d'un contexte, modification 67

R

recueil, exemple d'utilisation 62
régulier, espacement 3
régulier, rythme 3
retoucher 10
retouches, distances 92
`ronde` 16

S

SATB, squelette 77
Scheme 10
score, partition 41
scores, multiples 42
sensibilité à la casse 12, 18
shift, commandes 56
`silence` 16
snippets 10
spacing notes 55
staccato 22
stem down 52
stem up 52
structure d'hymne 58
`Structure d'une partition` 46
structure d'une partition vocale 57
structure de ficher 41
`Structure de fichier` 41, 43
suppression d'objets 91
symboles musicaux 2

T

`Tables du manuel de notation` 10
tenue, liaisons de 21
terminologie 9
`The \override command` 88, 89
`Top` 9, 10
`Tout savoir sur les contextes` 64
trait d'union (paroles) 32
transparents, objets 91
`transposition` 20
`triolet` 25
triolets 25
`Tunable context properties` 68, 71
typographie 3, 5

V

`valeur d'une note` 25
variables 10, 43, 95
`Vérification des limites et numéros de mesure` 94
`Vérifications d'octave` 94
versions 38
`Voice` 53
Voice, contexte 48
Voice, création de contextes 53
voix multiples 48
voix multiples sur une portée 30
voix temporaires 55
voix, imbrication 55