LilyPond, un logiciel d'écriture de partitions musicales

Jean Abou Samra

8 janvier 2025

Qu’est-ce que LilyPond ?

LilyPond est un logiciel pour éditer des partitions musicales. Particularités :

- Saisie de la musique sous forme de texte brut
- Bonne qualité typographique
- Configurabilité, extensibilité, « bidouillabilité »
- Notations anciennes et exotiques

Frescobaldi est un éditeur de texte spécialement dédié à LilyPond.

- Aller-retour entre code et rendu (« point-and-click »)
- Coloration syntaxique, autocomplétion
- Saisie MIDI
- Assistant de création de partition

Démarrage

- Installation de LilyPond : Binaires statiques pour Linux (glibc), macOS et Windows, à télécharger sur `lilypond.org`. Disponible dans la plupart des distributions Linux et dans Homebrew et MacPorts. Ou via Frescobaldi !
- Installation de Frescobaldi : Flatpak pour Linux, également dans les distributions, Homebrew, MacPorts.
- Site : https://lilypond.org
- Manuel d'initiation en français : https://lilypond.org/doc/v2.24/Documentation/learning/index.fr.html
- Forum francophone : https://lilypond.community
- Wiki : https://wiki.lilypond.community

Démonstration

Installation

Noms de note en néerlandais et français

Silences et silences multi-mesures, sauts

Clefs

Fine

Tempo

Commentaires

Nuances, articulations, doigtés, texte

n-olets, anacrouse

Portées

`\version`

Hauteurs absolues et relatives

Durées

Armure

Chiffrage de mesure

Bar checks

Liaisons

Ligatures manuelles

Paroles

MIDI

Intégration entre LilyPond et LaTeX

- `lilypond-book` est un script distribué avec LilyPond qui fonctionne comme préprocesseur.
- lyLuaTeX est un package qui exécute LilyPond lors de la compilation par TeX (nécessite LuaTeX).

Dans les deux cas :

```
\begin{lilypond}
...
\end{lilypond}
```

Le langage Scheme

Version simplifiée de Lisp, date de 1975

Plusieurs standards (R5RS, R6RS, R7RS-small). Nombreuses implémentations, LilyPond utilise Guile.

Typage dynamique et latent

Littéraux de base :

```
5 ; entier
-5 ; entier
0.52 ; flottant
#f ; faux
#t ; vrai
"abc" ; chaîne de caractères
'abc ; symbole (unique)
```

Le langage Scheme

Appel de fonction :

```
(+ 3 4)
```

Fonctions (récursion terminale guarantie) :

```
(define (f x)
  (+ x 2))
```

Conditions :

```
(if cond true false)
```

Variables :

```
(define a 42)
(let ((b 43))
  (+ a b))
```

Quoting :

```
'(+ 2 3) ; liste d'un symbole et deux
entiers
`(,(+ 2 3) ,(+ 4 5)) ; liste de deux
entiers
```

Fonctions anonymes :

```
(map (lambda (x) (x + 2))
     '(0 1 2))
```

Scheme dans LilyPond

Le caractère # introduit une expression Scheme.

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

emphasis = ##t
{
  \once \override NoteHead.color = #(if emphasis red black)
  c'
}
```

Dans l'autre sens :

```
measures = #(list #{ { c' d' e' f' } #} #{ { c' b a g } #})
```

Architecture de LilyPond

Au commencement étaient les **expressions musicales**. Représentation interne de la musique sous forme d'arbre.

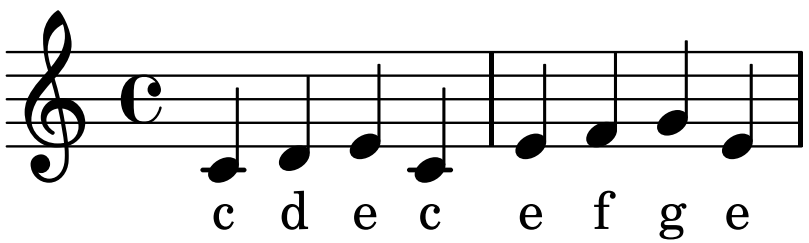
Traduction en **grobs** (graphical objects) via des **graveurs**

Les propriétés des grobs sont calculées via des **callbacks**.

Fonctions musicales

Une fonction musicale s'appelle avec `\` et renvoie une expression musicale.

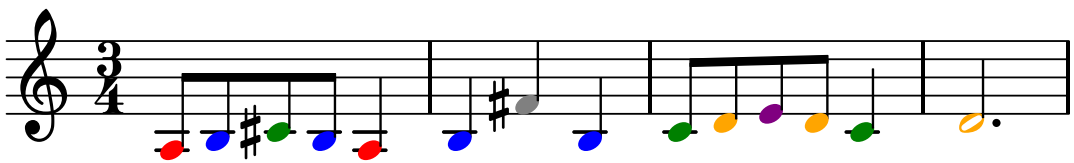
```
withNoteNames =
#(define-music-function (music) (ly:music?)
   #{
     <<
       \new Voice $music
       \new NoteNames $music
     >>
   #})

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



Ambiguité : `\fonction -2`

Callbacks

Un callback calcule une propriété d'un grob à partir d'autres propriétés.



```
\override NoteHead.color =
  #(lambda (grob)
     (case (ly:grob-property grob 'staff-position)
       ((-8) "red")
       ((-7) "blue")
       ((-6) "green")
       ((-5) "orange")
       ((-4) "purple")
       ((-3) "grey")
       (else "black")))
```

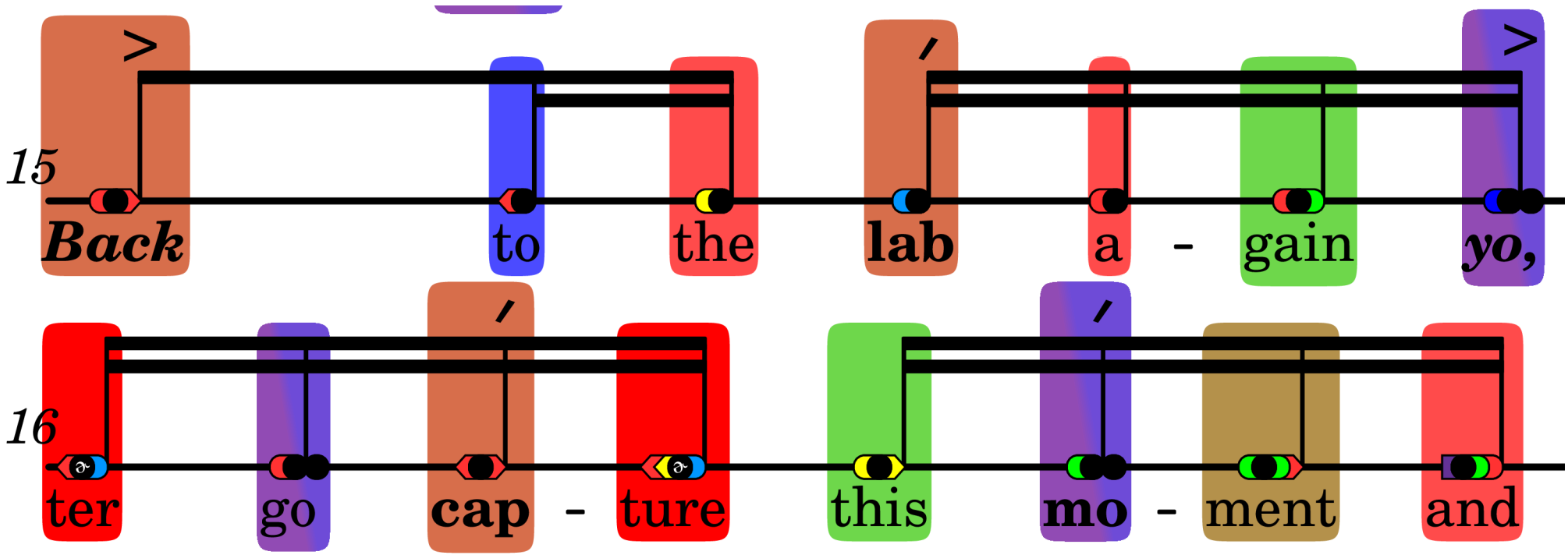
Exemples d'extensions

openLilyLib : edition-engraver (Jan-Peter Voigt) pour injecter des modifications, analysis (Klaus Blum) pour encadrer et surligner des groupes de notes, etc.

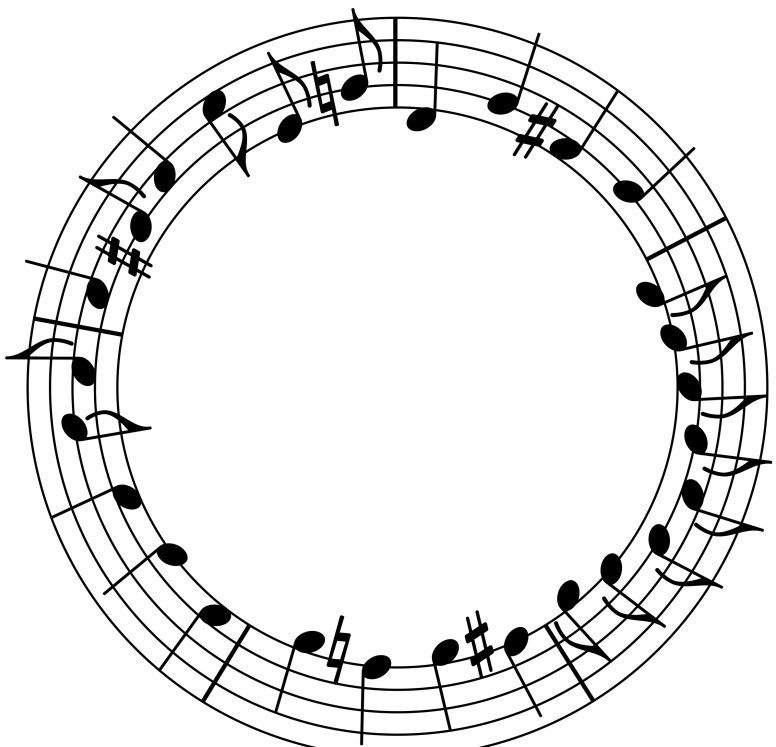
Plusieurs fonctionnalités développées d'abord comme extensions avant d'être intégrées : bends de guitare (Thomas Morley), grilles harmoniques (moi), …

Voire…

Exemples d'extensions



Exemples d’extensions



Développement

Un mélange de C++, Scheme et LilyPond, avec des scripts en Python

Développeurs principaux : Han-Wen Nienhuys, Jan Nieuwenhuizen, David Kastrup, Dan Eble, Jonas Hahnfeld, Werner Lemberg, Reinhold Kainhofer, Joe Neeman, Keith O'Hara, …

Police musicale écrite en METAFONT (la petite sœur de TeX)

≈ 150 000 lignes de code, manuel de notation de ≈ 900 pages en PDF (traduit en français, merci Jean-Charles Malahieude !), ≈ 2 000 tests

Dépôt sur GitLab, https://gitlab.com/lilypond/lilypond

Mailing list de développement, https://lists.gnu.org/mailman/listinfo/lilypond-devel

Un "patch meister" pour faire avancer les contributions

Limitations : interopérabilité

MusicXML : import possible avec `musicxml2ly` (développé avec LilyPond) ou `xml2ly` (outil externe développé par Jacques Menu). Export pas possible.

Pas de compatibilité SMuFL (format de polices musicales)

Limitations : complexité de la syntaxe

```
{ c' } % OK
\score { c' } % KO
\score { { c' } } % OK
\new Staff << { g' } \\ { c' } >> % OK
\score << { g' } \\ { c' } >> % KO
\score { << { g' } \\ { c' } >> } % OK
```

Analyseur syntaxique extrêmement complexe, intriqué avec l'interprétation. Difficile de donner des messages d'erreur faciles à comprendre.

Beaucoup de modes syntaxiques avec des règles subtilement différentes.

```
\mark #2 % OK
\mark 2 % OK
\markup \vspace #2 % OK
\markup \vspace 2 % KO
```

Limitations : complexité de l'interface

Manières de faire des réglages : fonctions globales, variables `\paper`, variables `\layout`, propriétés de grob, propriétés de contexte, …

Organisation d'une partition d'orchestre mal standardisée

Pas d'API bien définie, l'extension est de la chirurgie à cœur ouvert (pose des problèmes de compatibilité)

Scheme est un langage généralement peu apprécié des programmeurs débutants.

Limitations : un logiciel au long historique

Incohérences dues à l'histoire du développement, difficiles à résoudre de manière compatible (exemple : taille de police)

Extensions naturelles impossibles à implémenter : `R*5`

Performance loin d'être optimale, algorithmes difficiles à changer

Compilation longue et complexe, surtout la documentation. Pour Windows, bloquée pendant des années.

Interaction fragile entre Guile et C++

Impossible à compiler en WASM

Merci pour votre attention

Questions ?