

RLily (Ruby to Lilypond)

- [Introduction/Présentation](#)
- [Écriture du code partition](#)
 - [Écriture des notes](#)
 - [Écrire des barres de mesure](#)
 - [Définition de l'octave](#)
 - [Définir la clé](#)
 - [Écrire un accord](#)
 - [Écrire un triolet](#)
 - [Écrire un passage à l'octave \(8v---\)](#)
- [Masquer des éléments de la partition](#)
- [Éléments graphiques](#)
 - [Écrire une marque](#)
 - [Écrire un texte](#)
 - [Entourer une note d'un cercle](#)
- [Indication de l'opus](#)
- [Sous-titre](#)
- [Indication de l'armure](#)
- [Indication du tempo](#)
- [Indication de l'instrument](#)
- [Définir l'espace entre titre et premier système](#)
- [Définir l'espacement entre les systèmes](#)
- [Définir l'espacement entre les portées \(du piano\)](#)
- [Définir la taille de la partition \(notes/portées\)](#)
- [Réglage des options](#)
 - [Demander l'affichage des dimensions](#)
 - [Ajouter un slash entre les systèmes](#)

Introduction présentation

`RLily` permet de construire rapidement une partition à l'aide de Lilypond, en produisant un code façon ruby.

Dans TextMate, avec le Bundle RLily :

1. Écrire le code en ruby (dans un fichier `.rlily`)
2. Taper `CMD + MAJ + P`

Et la partition est produite.

Pour le moment, cette partition est très simple, elle est produite pour le piano (clé de sol et clé de Fa) et ne supporte que les ajouts de notes (et autres marques du code normal de Lilypond comme les doigtés, les liaisons, les accords, etc.)

Quick référence pour la création

1. Créer le fichier RLily (`.rlily`) qui va contenir le code pour la partition (c'est le "fichier source")
2. Définir les données du score ([Données générales du score](#))
3. Définir le contenu de chaque main avec `MD << "<notes>"` et `MG << "<notes>"`

La suite dépend de l'utilisation ou non du Bundle `Phil:RLily`. S'il est présent :

1. Avec le fichier source activé, utiliser le menu `Produire la partition` (ou jouer le raccourci `CMD + MAJ + P`)

Si le bundle n'est pas présent :

1. Ouvrir le fichier `_rlily.rb`
2. Définir la valeur de `SCORE_PATH` en mettant le path du fichier source
3. Runner `_rlily.rb` pour produire la partition

En cas d'échec

Dans certains dossiers, les permissions peuvent manquer. Dans ce cas, le

programme ouvre le fichier `.ly` dans TextMate (donc le fichier Lilypond que RLily produit), et il suffit alors de taper `CMD + R` pour produire la partition (si le Bundle Lilypond est installé).

Données générales du score

Définir dans le code du fichier source (toutes les valeurs peuvent être omises) :

```
SCORE::titre = "LE TITRE DU MORCEAU"
SCORE::compositeur = "LE COMPOSITEUR"
SCORE::armure = "L'ARMURE ('A', 'B', etc.)"
SCORE::metrique = "4/4"
```

Constantes utiles

Le fichier `lib/music/constantes_textuelles.rb` contient des constantes textuelles utiles, pour marquer des barres, etc.

Utilisation de la commande `rlily`

On peut construire une partition à l'aide de :

```
rlily "path/to/file_partition.rb"
```

Pour que ça fonctionne, il faut que la commande `rlily` ait été définie :

```
$ cd /usr/bin
$ sudo touch rlily
[Entrer mot de passe]
$ sudo chmod 0777 rlily
```

ci-dessous, vous devez remplacer `PATH/TO/RUBY2LILY/` par le path à votre dossier téléchargé de ruby2lily :

```
$ sudo echo 'exec PATH/TO/RUBY2LILY/ruby2lily.rb "$@"' > rlily
# $ sudo echo 'exec /Users/philippeperret/Programmation/Programmes/RLily/_rlily.rb "$@"' > rlily
$ sudo chmod u+x rlily
```

Masquer des éléments de la partition

On peut utiliser la pseudo-commande `hide(<element>)` (cacher) pour masquer des éléments de la partition, barres de mesures, hampes, etc.

`<element>` peut être :

Un Symbole	:beam
Une liste de symboles	[:beam, :staff]
Le symbole spécial	:all

Quand `:all` est envoyé à `hide`, peut définir les éléments qui ne doivent pas être masqués à l'aide d'un second argument :

```
hide( :all, { except: [<liste de symboles>] } )
```

Par exemple, pour masquer tout sauf les liaisons :

```
hide( :all, { except: [:slur] } )
```

On utilise à l'inverse la pseudo-commande `show(<element>)` pour que les éléments soient affichés à nouveau.

Par exemple :

```
MD << "c8 d e"
MD << hide(:stem)
MD << "f g a" # Les stems ne seront pas affichés
MD << show(:stem)
MD << "b c e" # Les stems sont remises
```

On peut bien sûr indiquer tout sur une même ligne :

```
MD << "c8 d e #{hide(:stem)} f g a #{show(:stem)} b c e"
```

Note : Si ce qui est à remonter avec `show` est identique à ce qui a été masqué avec `hide`, on peut simplement utiliser `show` sans arguments :

```
MD << "c8 d e"
MD << hide([:stem, :staff])
MD << "f g a" # Les stems et la portée ne seront pas affichés
MD << show
MD << "b c e" # Les stems et la portée ré-apparaissent
```

Éléments qu'on peut masquer :

ÉLÉMENT	CLÉ	EXEMPLE

TOUS	:all / :tous	
Barres de mesure	:bar / :barre	hide(:bar)
Hampes (Stems)	:stem / :hampe	hide(:stem)
Silences (Rest)	:rest / :silence	hide(:rest)
Tête de notes	:head / :tete	hide(:head)
Liaisons	:slur / :liaison	hide(:slur)
Crochets	:beam / :crochet	hide(:beam)
Métrique	:metrique	hide(:metrique)
Clé	:clef / :key	hide(:clef)

Éléments graphiques

Écrire une marque

Note : une “marque” est plus visible, plus grosse qu'un texte. Pour écrire un “texte”, cf. [Écrire un texte](#).

Utiliser la méthode `mark` pour placer une marque de répétition (pas de reprise, de répétition au sens de “répéter avec son groupe” par exemple).

```
mark("<notes>", "<texte de la marque>"[, { options }] )
```

Par exemple :

```
MD << mark("c1 c", "Reprendre ce passage")
```

Par défaut, la marque est placée **au centre de l'élément précédent et l'élément suivant**. Par exemple :

```
MD << mark("c1 c", "entre les deux")
```

... placera le "entre les deux" centré au-dessus de la barre séparant les deux mesures.

On peut demander que l'alignement se fasse à gauche ou à droite à l'aide du second argument de la méthode :

```
MD << mark("sur le premier", left: true)

MD << mark("c4 c c c", "sur le deuxième", right: true)
```

Écrire un texte

Note : un “texte” est plus petit qu’une “Marque”. Pour écrire une marque (de répétition par exemple, cf. [Écrire une marque](#)).

On peut utiliser la méthode `write` (ou ses alias `ecrire` ou `texte`) pour écrire un texte au-dessus/en dessous d'une note.

Le premier argument doit être la note, le deuxième argument doit être le texte et un troisième argument Hash peut définir les options.

```
ecrire( "<note>", "<texte>", {<options>} )
```

Par exemple :

```
MD << write( "c", 'c\'est un DO') + "d"
# => Écrira "C'est un DO" sur la note DO
```

Placer le texte en dessous

On utilise le second argument de l'appel à la méthode en définissant la clé `:up`. Si elle est fausse, le texte est marqué en dessous de la note.

Par exemple :

```
MG << texte('c', 'en dessous', { up: false } )
```

Ne pas étirer le texte

Par défaut, la note suivante sera placée à la fin du texte. Pour éviter ce comportement, on peut ajouter `:stretch => false` (ou `stretch: false`) dans les options.

Par exemple :

```
MD << write("c", "Un texte qui couvrira les autres notes",
{stretch: false} )
```

Noter que cette option ne s'applique qu'au texte visé. Les textes suivants seront à nouveau "étirant".

Entourer une note d'un cercle

Il suffit de placer `\circle` devant, par exemple :

```
MD << "c \circle d e" # la note Ré sera entourée
```

Indication de l'opus

```
SCORE::opus = <opus>
```

Par exemple :

```
SCORE::opus = 13
```

Note : Ne pas mettre "Opus" ou "Op."

Sous-titre

Pour indiquer un sous-titre :

```
SCORE::sous_titre = "<le sous-titre>"
```

Indication de l'armure

```
SCORE::armure = <armure>
```

`<armure>` se désigne comme les accords : "A#" pour La dièse majeur, "Bbm" pour Si bémol mineur, etc.

Indication du tempo

Tempo pour la valeur de la noire

```
SCORE::tempo = <tempo|tempo name>
```

Par exemple :

```
SCORE::tempo = 126
```

```
SCORE::tempo = "Andante"
```

Tempo pour une valeur autre que noire

```
SCORE::tempo = { :duree => <durée>, :tempo => <tempo>, :name  
=> <nom tempo> }
```

Par exemple, pour mettre la croche (8) à 60 :


```
SCORE::tempo = {:duree => 8, :tempo => 60, :name => "Allegro"}
```

Définir l'instrument

```
SCORE::instrument = <instrument>
```

Définir l'espace entre titre et premier système

```
SCORE::espace_titre_systemes = <valeur>
```

Définir l'espacement entre les systèmes

```
SCORE::espace_entre_systemes = <valeur>
```

Définir l'espace entre les portées

```
SCORE::espace_entre_portees = <valeur> # défaut : 3 (ne pas mettre une valeur trop grande)
```

Définir la taille de la partition

```
SCORE::taille = <valeur>
```

Où `<valeur>` peut être :

PARTITION_CLASSIQUE/20	Taille normale pour les partitions classiques
LIVRET_POCHE/11 13, 14, 16	Les livrets de poche
LIVRE_CHANT / 18 23 et 26	Les livres de chant

Écrire le code de la partition

Concaténation

Pour éviter les erreurs en additionnant des strings, la class `String` a été modifiée :

```
"a" + "b"    => "a b"
"a" << "b"   => "a b"
```

Donc il est inutile de terminer ou de commencer les strings de notes par des espaces.

Options pour les fonctions musicales

Ces options sont le second argument de toutes les fonctions musicales. Elles peuvent définir :

```
:duree      La durée de la note/des notes
:octave     L'octave (note : en notation anglosaxone, donc le DO sous la portée de Sol est un Do 4)
:doigte     Le doigté à utiliser
:jeu        Le mode de jeu, défini explicitement avec du code
            lilypond (".", "-", etc.) ou les constantes textuelles
```

Écriture des notes

Les notes s'écrivent comme dans Lilypond, à l'aide de **lettres minuscules de a à g**.

Pour les altérations, on peut utiliser la notation lilypond :

```
ais => la dièse  
aes => la bémol
```

... ou la notation avec `#` et `b` :

```
a#  => la dièse  
ab  => la bémol  
bb  => si bémol
```

Écrire les barres de mesure

L'écriture des barres de mesure dans RLily est très intuitive.

Pour une **double barre**, ils suffit d'écrire... une double-barre :

```
c a || b c
```

Pour la **barre de fin** :

```
c a f ||.
```

Pour une **reprise simple** :

```
|: c d e f :|
```

Pour une **double reprise** :

```
|: c d e f :|: f e d c :|
```

Pour une **reprise avec alternative**

```
|: <notes> |1 <notes première fois> :|2 <notes deuxième fois> ||
```

Plusieurs choses sont à noter ici :

- Seules 2 alternatives sont possibles avec cette écriture. S'il doit y en avoir plus, utiliser le code normal de Lilipond :

```

\repeat volta <x fois> {
  <notes>
}
\alternative {
  { <première fois> }
  { <première alternative> }
  { <deuxième alternative> }
  ...
  { <dernière fois> }
}

```

- La fin de la deuxième alternative doit obligatoirement être indiquée par une double barre (`||`). Mais cette double-barre ne sera pas "écrite". Pour avoir réellement une double-barre sur la partition, il faut donc la répéter :

```
| : c | 1 d : | 2 e || ||
```

Définir l'octave

Si des variables sont employées pour simplifier le code et le rendre plus explicite, il est bon de définir l'octave précise des motifs pour qu'ils puissent être enchainés. On utilise pour ça la fonction `rel` (pour "relative") :

```

rel(<notes>, <octave>)
# La première des <notes> commencera toujours à l'octave <octave>

```

Par exemple :

```

rel("c e g", 5)
# => \relative c' '{ c e g }

```

Note : Pour définir une écriture avec "8ve----", cf. [Écrire à l'octave](#).

Définir la clé

```
MD << <clé>
```

Par exemple :

```
# Par constante (cf. ci-dessous)
MD << CLE_FA

# Explicitement
MD << "\\clef \"bass\""
```

Constantes :

CLE_F/CLE_FA	Clé de fa
CLE_G/CLE_SOL	Clé de sol

Écrire un accord

```
chord("<notes\"|[notes]>[, <options>])
```

Par exemple :

```
MD << chord("c e g")
# => "<c e g>"

MD << chord([c, E, g], :duree => 8)
# => "<c e g>8"
# Noter que les notes peuvent être définies par des minuscules ou des majuscules

MD << chord("c e g", {:duree => 4, :jeu => PIQUE})
# => "<c e g>4-."
```

Pour les `<options>`, cf. [Options pour les fonctions musicales](#)

Écrire un triolet

```
triolet(<notes>[, <options>])
```

Par exemple :

```
MD << triolet([c, d, e])
# => "\tuplet 3/2 { c d e }"

MD << triolet("c d e", :duree => 16)
# => "\tuplet 3/2 { c16 d e }"
```

Pour les `<options>`, cf. [Options pour les fonctions musicales](#)

Écrire à l'octave

```
octave(<notes>[, <nombre d'octave (1 par défaut)>])
```

Par exemple :

```
MD << octave("c' e f g")
# => Ecrit les notes une octave en dessous, avec la marque
"8ve-----"

MD << octave("c''' e f g", 2)
# => Ecrit les notes une octave en dessous, avec la marque
"15ma-----"
```

Réglage des options

Demander l'affichage des dimensions

```
SCORE::option :display_spacing => true

ou

SCORE::option :display_spacing, true
```

Ajouter un slash entre les systèmes

```
SCORE::option :slash_between_systemes => true
```

