LilyPond Contemporary Notation Cookbook: Snippets and Their Grammars

Yoshiaki Onishi
School of Music, University of Delaware
info@yoshionishi.com

Version: December 28, 2024

Cite: Onishi, Yoshiaki. "LilyPond Contemporary Notation Cookbook: Snippets and Their Grammars," (Version December 28, 2024), GitHub repository, https://github.com/yoshiakionishi/lilypond-snippets

Contents

Foreword

0.1 Preamble

This document houses all the codes I built on LilyPond since September 2024. Because I deal with contemporary notations in my compositional practice, I found myself creating codes and turning them into variables in order to repeatedly use them in my projects. I created a dedicated `.ly` file to store these codes for use, which quickly became very lengthy. I thought it would be useful to organize them into a document where I could easily consult and remind myself what they are and how to use them. This is that document.

Because I use LilyPond actively in my daily compositional and musical typesetting activities, this document is a work in progress.

0.2 README


```
% Original Code written by Yoshiaki Onishi
% https://github.com/yoshiakionishi/lilypond-snippets
```

I make this document public because I wish to return something useful to the LilyPond community, but also to seek and implement any improvements other users may find in my codes. Please feel free to reach out to the email address shown on the title page of this document.

In the interest of making the codes found in this document available to as many people as possible, I have avoided using copyrighted musical examples. However, wherever appropriate, I have provided bibliographical sources. Furthermore, I acknowledge that, just as academic work in humanities goes, my ideas are built on those that are formulated by

others; as such, whenever there is a direct source of inspiration for formulating a code, I provide sources.

In creating this document, I make no claim that my notational choices represent an absolute standard that everyone should adhere to. Once the basic principles of notation and typesetting are established (e.g., avoiding collisions, etc.), notation becomes a personal decision for each composer, shaped by careful study of preexisting scores and an evaluation of their musical contexts.

For example, in his book *The Bass Clarinet – A Personal History*, Harry Sparnaay lists nine variants of noteheads for the slap tongue technique.[1] In my work, I created two subcategories of the slap tongue technique: one followed by a pitch and another followed by an air sound (which produces the slap tongue effect that sounds "empty"). To distinguish between the two, I decided to use encircled noteheads—both filled and hollow—and attribute them to each subcategory. Again, this is a method that I have found works for my music, but I would be reluctant to suggest that others should follow the same.[2]

Readers are encouraged to modify my codes in order to suit their desired techniques. This document serves as a record of how I arrived at certain notational choices, because learning LilyPond meant that I would also need to become familiar with Scheme, which proved to be somewhat challenging—even though I have used Common Lisp before owing to programming in OpenMusic—because I had to make many guessworks as I navigated various Scheme codes in other snippets available online. I have also gained familiarity in PostScript language as I continued to familiarize myself with LilyPond.[3]

0.3 Background

After MakeMusic announced that they would cease development of the music notation software program Finale, which I had used for the past twenty-four years, I decided to explore a few other music notation programs to determine the best alternative. At the time of writing this document in late November 2024, a little under three months have passed since I started using LilyPond for my daily typesetting needs. I now open LilyPond more often than Finale and am committed to using it for the foreseeable future. LilyPond appears to me as the way forward both as a composer and a musical typesetter, as other proprietary notation programs, such as Dorico (which MakeMusic has claimed to be the leading program in the industry) and Sibelius, fall short of what I wish to accomplish.

While LilyPond is "just" a music notation software program that I happened to choose, it

1. Harry Sparnaay, *The Bass Clarinet: A Personal History* (Periferia Sheet Music, 2012), 66.

2. This particular notation becomes quickly problematic in terms of rhythmic notation when a bar is longer than a half note (e.g. 1/2, 2/4, 4/8...) For this reason, I tend to favor time signatures that avoids the use of a half note, such as 3/8 or 5/16.

3. See Appendix A for some resources I referred to for Scheme- and PostScript-related matters.

is, in a way, more than a toolkit for a composer. It appears that way to me, at least, because choosing an open-source platform with strong community support and engagement, rather than a proprietary program where desired functionality is subject to the priorities of a small group of salaried developers, reflects a critique of the capitalist/commercialist mindset that often pervades a composer's life.

For example, before transitioning to LilyPond, I briefly explored Dorico. However, as of late September 2024, its functionality for displaying straight flags was very limited; the angle of the straight flags provided by the software was too steep. I consulted the online forum and discovered that another user had posted a question similar to mine. The chief developer of Dorico responded to that post, noting that implementing improvements to this feature was possible but "not currently a high priority."[4] In this tiered structure typical of capitalism, composers may find themselves with increasingly limited creative "freedom."

MakeMusic has heavily advertised on social media platforms that Finale users should migrate to Dorico because it is the "next industry standard." However, this advertising seems to discourage thoughtful consideration of alternatives, leaving little room for reflection or exploration. I became increasingly disillusioned as I witnessed the coercion to invest in a program—however exciting it may appear—with no definite promise of its long-term security and stability.

Of course, it is not my intent to claim that all composers should abandon their proprietary programs of choice, particularly those they have invested money in and/or have been using for many years. It is, however, important to note that:

1. All proprietary programs are at the mercy of the executives who run the companies behind them. "Oh, [insert the name of a proprietary program] is operated by [insert the name of its company], and I just don't see them closing the program down," someone might say. Yet, it happened to Finale.

2. All notation programs, owing to the ways they operate, exert some degree of influence on the way composers compose. As early as the 1980s, Finale's *Mass Mover*, *Note Mover*, and MIDI playback features were already influential in shaping the way composers worked on their music.[5] On the one hand, these features may have helped composers save time. On the other hand, their ready availability may have invited overuse.

3. The lack or underdevelopment of certain functionalities may also push composers to work in certain ways rather than others. Finale benefitted from having the flexibility to implement graphical notation, but even then, many of my composer friends found it practical to use external graphical editing programs to further refine their scores.

4. See: https://forums.steinberg.net/t/straight-flags-angle/766503.
5. For example, watch from 15:20 of https://youtu.be/T1IRlg87Qks.

Even from my personal experience using Finale, I encountered situations where I had to devise creative alternatives to meet my notational goals.

These points implicitly highlight the benefits of learning an additional notation program, ideally an open-source one, alongside the program one primarily uses. LilyPond resonated with me most because of its text-based interface, which I have become increasingly familiar with through my involvement in computer programming. As other users have remarked, I have also found it to be very flexible and extensible. All the snippets I list in this document can be reused with relative ease, allowing me to save time in the long run when using specialized notations in my music. This was not necessarily the case when working on the music notation of extended techniques in Finale.

0.4 How This Document Is Structured

Each chapter of this document addresses a specific element of music notation, such as noteheads, stems, beams, and so on. Some chapters, however, cover topics specific to LilyPond coding, such as Markups and Spanners. Snippets that use more than one snippet covered in earlier chapters, thus simulating practical applications of these snippets, are covered in the chapter *Combinations*. Snippets that do not appear to belong to earlier chapters find their home in the chapter *Miscellanies*.

Each snippet entry includes a musical example, a description, the relevant grammar, the code required for the snippet to function, and, whenever necessary, a "Discussion" section.

0.5 LilyPond Version Used

The version of LilyPond used to create these snippets is `2.24.4`.

0.6 Acknowledgements

I thank the supportive community of LilyPond users, whose exchanges on `lilypond-user` mailing list have inspired me greatly.

Even though I have not met him, I am grateful to Ben Lemon for his generosity in creating and sharing his LilyPond tutorial videos on YouTube. These videos were immensely helpful during the initial stages of learning LilyPond.

I also want to thank my friends who inspired me to start using LilyPond. It was Cole Ingraham who first introduced me to the program in 2016. My initial attempt at using it was not successful, but more recently, Santiago Beis composed and typeset his orchestral piece *Spletna* entirely in LilyPond, which compelled me to give it another try.

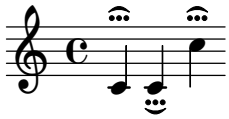
I extend my gratitude to my composition students at the University of Delaware School of Music, with whom I embarked on this journey of learning LilyPond. Even though they were not directly affected by Finale's discontinuation, they remained curious and enthusiastic about exploring this program. I hope that if the programs of their choice ever face a fate similar to Finale (though I sincerely hope they do not), they will be better equipped to adapt without the annoyance and arduous work often associated with transitioning to a new tool.

Table of Contents

Chapter 1

Articulations

1.1 Jeté (Ricochet)



1.1.1 Description

I use this notation to designate jeté/ricochet for string instruments, adding that the number of bounces are undetermined.[1]

I apply this indication *above* the note regardless of how high or low the note is; however, in case of need, I have supplied the version to be used *under* the note.

1.1.2 Grammar

```
NOTE \jete
NOTE \jeteUp
NOTE \jeteDown
```

1. Concerning the technique of adding articulation designs to an internal alist, I was inspired by the following thread on `lilypond-user` mailing list: https://lists.gnu.org/archive/html/lilypond-user/2015-04/msg00105.html

1.1.3 Code

```
\version "2.24.4"

jeteDesign =
\markup
\center-align
\combine \combine \combine
\override #'(filled . #t)
\path  #0.1
#'((moveto    -0.25 0.5)
   (curveto   0.35 1.1 0.85 1.1 1.45 0.5)
   (curveto   0.85 0.8 0.35 0.8 -0.25 0.5)
   (closepath))
\draw-circle #0.2 #0 ##t
\translate #'(0.6 . 0) \draw-circle #0.2 #0 ##t
\translate #'(1.2 . 0)\draw-circle #0.2 #0 ##t
#(append! default-script-alist
    (list
     `(jetelistUp
       . (
           (stencil . ,ly:text-interface::print)
           (text . ,#{ \markup \jeteDesign #})
           ; any other properties
           (toward-stem-shift-in-column . 1.0)
           (outside-staff-priority . #t)
           (padding . 0.5)
           (avoid-slur . around)
           (direction . ,UP))))

    (list
     `(jetelistDown
       . (
           (stencil . ,ly:text-interface::print)
           (text . ,#{ \markup \rotate #180 \jeteDesign #})
           ; any other properties
           (toward-stem-shift-in-column . 1.0)
           (outside-staff-priority . #t)
           (padding . 0.5)
           (avoid-slur . around)
           (direction . ,DOWN)))))
```

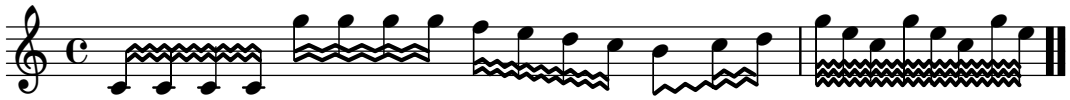
```
40
41  jete =  #(make-articulation 'jetelistUp)
42  jeteUp = #(make-articulation 'jetelistUp)
43  jeteDown = #(make-articulation 'jetelistDown)
44
45
46  {c'4\jete c'4 \jeteDown c''\jeteUp }
```

Table of Contents

Chapter 2

Beams

2.1 Wiggle Beam (zig-zag shaped beam)



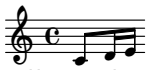
2.1.1 Description

Ordinary beams are replaced with zig-zag beams. A set of forward then backward beams are printed in the amount specified in the argument. I use this notation in such pieces as *jeux enjeux* (2022) for brass quintet, in order to designate somewhat uneven rhythmic figures, which are nonetheless to be played within the time frame indicated.

`\wiggleBeamOne` replaces an 8^{th}-note beam.

`\wiggleBeamTwo` replaces a 16^{th}-note beam.

`\wiggleBeamThree` replaces a 32^{nd}-note beam.

`\wiggleBeam_markup` adds a zig-zag beam at will. This allows beaming of mixed note durations, such as: 

`\wiggleBeamStemAdjust` allows the adjustment of a stem length, in the event the wiggle beam and the stem do not touch each other.

2.1.2 Grammar

```
\wiggleBeamOne #vOffset #howMany #width #rotation
```

`\wiggleBeamTwo #vOffset #howMany #width #rotation`
`\wiggleBeamThree #vOffset #howMany #width #rotation`

NB

- `hOffset` = (`\wiggleBeam_markup` only) the horizontal offset value originating from where the ordinary beam is placed.
- `vOffset` = the vertical offset value originating from where the ordinary beam is placed.
- `howMany` = how many "wiggles" to print. It only accepts integers.
- `width` = how wide each "wiggle" should appear. When in doubt, start with `#1`.
- `rotation` = a positive value would rotate the beam upward, and the negative value would rotate the beam downward.

`NOTE \wiggleBeam_markup #hOffset #vOffset #howMany #width #rotation`

NB

- `hOffset` = the horizontal offset value originating from where the ordinary beam is placed.
- `vOffset` = the vertical offset value originating from where an above-staff markup is placed. Thus, `#0` would place a wiggle beam above the staff line.
- `howMany` = how many "wiggles" to print. It only accepts integers.
- `width` = how wide each "wiggle" should appear. When in doubt, start with `#1`.
- `rotation` = a positive value would rotate the beam upward, and the negative value would rotate the beam downward.
- More than one `\wiggleBeam_markup` may be added in sequence, provided that for each instance all the arguments are defined.

`\wiggleBeamStemAdjust #fromMiddleLine #howFar NOTE`

NB

- `fromMiddleLine` = (`\wiggleBeamStemAdjust` only) = determines one end of the stem, `#0` being the middle line of an ordinary 5-line staff.
- `howFar` = (`\wiggleBeamStemAdjust` only) = computes how long the stem should be extended. A positive value would draw the stem upward, and a negative value would

draw the stem downward. An integer corresponds to the distance between two staff lines of an ordinary 5-line staff.

2.1.3 Code

```
wiggleBeamOne =
#(define-music-function (vOffset howMany howWide howTilted)
   (number? number? number? number?) #{
     \once \override Voice.Beam.stencil = #ly:text-interface::print
     \once \override Voice.Beam.text = \markup {
       \translate #(cons 0 vOffset)
       \postscript #(string-append
             "newpath
              1 setlinejoin
              1 setlinecap
              0.35 setlinewidth
              0.13 0 moveto "
             (number->string howMany)
             " {" (number->string (* 0.6 howWide))  " "
             (number->string (+ 0.5 howTilted)) " rlineto "
             (number->string (* 0.6 howWide))
             " -0.5 rlineto} repeat
              stroke"
                   )

     }
   #})


wiggleBeamTwo =
#(define-music-function (vOffset howMany howWide howTilted )
   (number? number? number? number?) #{
     \once \override Voice.Beam.stencil = #ly:text-interface::print
     \once \override Voice.Beam.text = \markup {
       \translate #(cons 0 vOffset)
       \postscript #(string-append
             "newpath
              1 setlinejoin
              1 setlinecap
              0.35 setlinewidth
              0.13 0 moveto "
```

```
              (number->string howMany)
              " {" (number->string (* 0.6 howWide)) " "
              (number->string (+ 0.5 howTilted)) " rlineto "
              (number->string (* 0.6 howWide))
              " -0.5 rlineto} repeat
               stroke newpath
               0.35 setlinewidth
               1 setlinejoin
               0.13 -0.75 moveto "
              (number->string howMany)
              " {" (number->string (* 0.6 howWide))  " "
              (number->string (+ 0.5 howTilted)) " rlineto "
              (number->string (* 0.6 howWide))
              " -0.5 rlineto} repeat
               stroke"
                     )
      }
    #})


wiggleBeamThree =
#(define-music-function (vOffset howMany howWide howTilted )
   (number? number? number? number?)
   #{
     \once \override Voice.Beam.stencil = #ly:text-interface::print
     \once \override Voice.Beam.text = \markup         {
       \translate #(cons 0 vOffset)
       \postscript #(string-append
             "newpath
              1 setlinejoin
              1 setlinecap
              0.35 setlinewidth
              0.13 0 moveto "
               (number->string howMany) " {"
               (number->string (* 0.6 howWide))  " "
               (number->string (+ 0.5 howTilted)) " rlineto "
               (number->string (* 0.6 howWide))
             " -0.5 rlineto} repeat
              stroke
              newpath
              0.35 setlinewidth
```

```
                1 setlinejoin
                0.13 -0.75 moveto "
                 (number->string howMany) " {"
                 (number->string (* 0.6 howWide))  " "
                 (number->string (+ 0.5 howTilted)) " rlineto "
                 (number->string (* 0.6 howWide))
                 " -0.5 rlineto} repeat
                stroke
                newpath
                0.35 setlinewidth
                1 setlinejoin
                0.13 -1.5 moveto "
                 (number->string howMany) " {"
                 (number->string (* 0.6 howWide)) " "
                 (number->string (+ 0.5 howTilted)) " rlineto "
                 (number->string (* 0.6 howWide))
                 " -0.5 rlineto} repeat
                stroke"
                       )
      }
    #})


wiggleBeam_markup =
#(define-music-function (hOffset vOffset howMany howWide howTilted )
   (number? number? number? number? number?)
   #{
     ^\markup         {
       \translate #(cons hOffset vOffset)
       \postscript #(string-append
             "newpath
              1 setlinejoin
              1 setlinecap
              0.35 setlinewidth
              0.17 0 moveto "
             (number->string howMany) " {"
             (number->string (* 0.6 howWide))  " "
             (number->string (+ 0.5 howTilted)) " rlineto "
             (number->string (* 0.6 howWide))
             " -0.5 rlineto} repeat
              stroke"
```

```
                      )

      }
    #})

wiggleBeamStemAdjust =
#(define-music-function (fromMiddleLine howFar)
   (number? number?)
   #{
     \once \override Stem.stencil = #ly:text-interface::print
     \once \override Stem.text = \markup {
       \postscript #(string-append
             "newpath
              0.12 setlinewidth
              0 " (number->string fromMiddleLine) " moveto
              0 " (number->string howFar) " rlineto
              stroke"
                     )
     }
   #})

{
  \wiggleBeamTwo #0 #9 #1.01 #0 c'16 c'
  \wiggleBeamStemAdjust #-3 #3.4 c' c'
  \wiggleBeamTwo #0 #5 #1.82 #0 g''
  \wiggleBeamStemAdjust #2.5 #-3 g''
  \wiggleBeamStemAdjust #2.5 #-3 g'' g''
  \wiggleBeamTwo #-1 #9 #1.01 #-0.15 f''
  \wiggleBeamStemAdjust #1.5  #-3.5 e''
  \wiggleBeamStemAdjust #1 #-3.5 d''
  \wiggleBeamStemAdjust #0.5 #-3.5 c''
  \wiggleBeamOne #-3.5 #5 #1.4 #0.15 b'8
  c''16 \wiggleBeam_markup #0 #-4.8 #2 #1.4 #0.15 d''
  \wiggleBeamThree #-1.3 #19 #0.73 #0  g''32
  \wiggleBeamStemAdjust #1.5  #-4 e''
  \wiggleBeamStemAdjust #0.5  #-3 c'' g'' e''
  \wiggleBeamStemAdjust #0.5  #-3  c''
  \wiggleBeamStemAdjust #2.5  #-5 g'' e''
  \bar ".."
}
```

2.1.4 Discussion

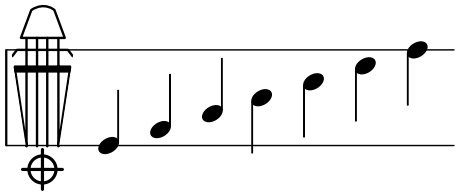
1. Admittedly, while the current setup allows great flexibility in making the wiggle beams appear, it is entirely possible that some of the parameters be automated.
2. When using many wiggle beams, it may be easier to make the score proportionally notated, in order to avoid the micromanagement of the parameters.

Table of Contents

Chapter 3

Clefs

3.1 String Position Clef



3.1.1 Description

String position clef to indicate bowing position. See Discussion for the associated command, `\normalClef`.

3.1.2 Grammar

`\strPosClef`

3.1.3 Code

```
1 strPosClefDesign = #(ly:make-stencil (list 'embedded-ps
2 "gsave
3 currentpoint translate
4 /fingboardpath
5 {
```

```
6  newpath
7
8  -0.55 7.5 moveto
9  0 -3 rlineto
10 1 -6.5 rlineto
11 -1 -1 rlineto
12 0 -3 rlineto
13 4.1 0 rlineto
14 0 3 rlineto
15 -1 1 rlineto
16 1 6.5 rlineto
17 0 3 rlineto
18 closepath
19
20 } def
21
22 fingboardpath clip
23 newpath
24 0.15 setlinewidth
25 0.5 4.75 moveto
26 0 -6.8 rlineto
27 -0.75 5 rlineto
28 3.5 0 rlineto
29 -0.75 -5 rlineto
30 0. 6.8 rlineto
31 stroke
32 0.35 setlinewidth
33 -0.4 2.75 moveto
34 3.75 0 rlineto
35 stroke
36
37 %inner two line
38 newpath
39 0.15 setlinewidth
40 1.16 4.75 moveto
41 0. -6.8 rlineto
42 1.8 4.75 moveto
43 0. -6.8 rlineto
44 stroke
45
46 %bridge
```

```
47 newpath
48 -0.4 3.6 moveto
49 0.3 0.4 rlineto
50 3.2 0 rlineto
51 0.3 -0.4 rlineto
52 stroke
53
54 %tailpiece
55 0.15 4.75 moveto
56 1 setlinecap
57 1 setlinejoin
58 2.75 0 rlineto
59 -0.65 1.75 rlineto
60 -0 -0  -0.6 0.55 -1.45 0  rcurveto
61 closepath
62 stroke
63
64 %mutesign
65 newpath
66 0.2 setlinewidth
67 1 setlinecap
68 1.5 -2.25 moveto
69 0 -2.5 rlineto
70 0.25 -3.5 moveto
71 2.5 0 rlineto
72 stroke
73 newpath
74 1.5 -3.5 0.85 0 360 arc
75 stroke
76 grestore")
77         (cons 0 3)
78         (cons 0 1))
79
80 strPosClefSize =
81 #(lambda (grob)
82    (let* ((sPCS (ly:grob-property grob 'font-size 0.0))
83           (mult (magstep sPCS)))
84      (ly:stencil-scale
85       strPosClef
86       mult mult)))
87
```

```
88  strPosClef = {
89    \override Staff.Clef.stencil = \strPosClefDesign
90  }
91
92  normalClef = {
93    \revert Staff.Clef.stencil
94  }
95
96  {
97    \override Staff.StaffSymbol.line-positions = #'(6  -6)
98    \override Staff.LedgerLineSpanner.stencil = ##f
99    \override Staff.TimeSignature.stencil = ##f
100   \override Staff.BarLine.stencil = ##f
101   \strPosClef c'4 e' g' b' d'' f'' a''
102 }
```

3.1.4 Discussion

1. With the current design, `c'` would place a note at the lower end of the fingerboard. `a''` would place a note on the same line as the bridge.

2. The current design comes with the mute sign. If the mute sign is not needed, remove the following portion of the code above:

```
64  %mutesign
65  newpath
66  0.2 setlinewidth
67  1 setlinecap
68  1.5 -2.25 moveto
69  0 -2.5 rlineto
70  0.25 -3.5 moveto
71  2.5 0 rlineto
72  stroke
73  newpath
74  1.5 -3.5 0.85 0 360 arc
75  stroke
```

3. Once `\strPosClef` is used, in order to revert back to the normal clef, `\normalClef` must be used.

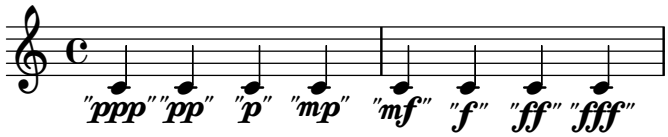
4. See Prescriptive Notation for String Instruments for a possible use of this clef.

Table of Contents

Chapter 4

Dynamics

4.1 Dynamics in Quotation Marks



4.1.1 Description

Dynamics in quotation marks, also known as *effort dynamics*, indicate those with which certain techniques must be carried on, understanding that the perceived dynamics will be quieter than what are indicated. Examples abound in scores by Helmut Lachenmann and others for such techniques as air sound, bowing directly on the bridge, etc..

4.1.2 Grammar

```
NOTE \qppp
NOTE \qpp
NOTE \qp
NOTE \qmp
NOTE \qmf
NOTE \qf
NOTE \qff
NOTE \qfff
```

4.1.3 Code

```
\version "2.24.4"

qmp = #(make-dynamic-script
        (markup #:combine
                #:combine
                #:translate '(-0.85 . -0.1)
                #:normal-text (#:italic #:fontsize 0.75 "\"")
                #:dynamic "mp"
                #:translate '(3.25 . -0.1)
                #:normal-text (#:italic #:fontsize 0.75 "\"")))
qp = #(make-dynamic-script
       (markup #:combine
               #:combine
               #:translate '(-0.95 . -0.1)
               #:normal-text (#:italic #:fontsize 0.75 "\"")
               #:dynamic "p"
               #:translate '(1.35 . -0.1)
               #:normal-text (#:italic #:fontsize 0.75 "\"")))
qpp = #(make-dynamic-script
        (markup #:combine
                #:combine
                #:translate '(-0.95 . -0.1)
                #:normal-text (#:italic #:fontsize 0.75 "\"")
                #:dynamic "pp"
                #:translate '(2.75 . -0.1)
                #:normal-text (#:italic #:fontsize 0.75 "\"")))
qppp = #(make-dynamic-script
         (markup #:combine
                 #:combine
                 #:translate '(-0.95 . -0.1)
                 #:normal-text (#:italic #:fontsize 0.75 "\"")
                 #:dynamic "ppp"
                 #:translate '(4.25 . -0.1)
                 #:normal-text (#:italic #:fontsize 0.75 "\"")))

qmf = #(make-dynamic-script
        (markup #:combine
                #:combine
                #:translate '(-0.85 . 0)
```

```
                #:normal-text (#:italic #:fontsize 0.75 "\"")
                #:dynamic "mf"
                #:translate '(3.25 . 0)
                #:normal-text (#:italic #:fontsize 0.75 "\""))))
qf = #(make-dynamic-script
       (markup #:combine
               #:combine
               #:translate '(-0.75 . 0)
               #:normal-text (#:italic #:fontsize 0.75 "\"")
               #:dynamic "f"
               #:translate '(1.65 . 0)
               #:normal-text (#:italic #:fontsize 0.75 "\"")))
qff = #(make-dynamic-script
        (markup #:combine
                #:combine
                #:translate '(-0.75 . 0)
                #:normal-text (#:italic #:fontsize 0.75 "\"")
                #:dynamic "ff"
                #:translate '(2.75 . 0)
                #:normal-text (#:italic #:fontsize 0.75 "\"")))
qfff = #(make-dynamic-script
         (markup #:combine
                 #:combine
                 #:translate '(-0.75 . 0)
                 #:normal-text (#:italic #:fontsize 0.75 "\"")
                 #:dynamic "fff"
                 #:translate '(3.85 . 0)
                 #:normal-text (#:italic #:fontsize 0.75 "\"")))

{

  c'4\qppp
  c'4\qpp
  c'4\qp
  c'4\qmp

  c'4\qmf
  c'4\qf
  c'4\qff
  c'4\qfff

```

```
81  }
82
83  \layout {
84    \context {
85      \Score    proportionalNotationDuration = #(ly:make-moment 1/9)
86    }
87  }
```

4.1.4 Discussion

In scores by Lachenmann, in concordance with German quotation marks (*Anführungszeichen*), the opening quotation mark points left, and placed on the bottom line, and the closing quotation mark points right and sits at the top of the last character. It would be possible to achieve this by adjusting the parameters in the Scheme code.[1]

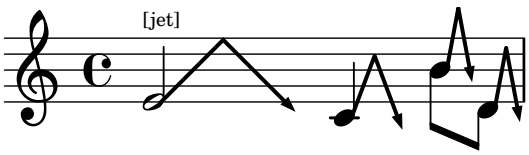
Table of Contents

1. See: https://lilypond.org/doc/v2.24/Documentation/extending/markup-construction-in-scheme

Chapter 5

Noteheads

5.1 Jet Whistle (for flute)



5.1.1 Description

Implementation of the jet whistle, as described in *The Techniques of Flute Playing* by Carin Levine and Christina Mitropoulos-Bott.[1]

5.1.2 Grammar

```
\jet NOTE #X-length
```

5.1.3 Code

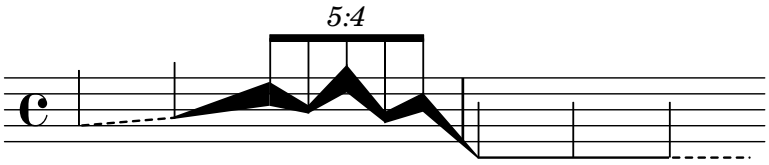
```
jet = #(define-music-function (pitchthing width) (ly:music? number?)
          (define p1 (ly:music-property pitchthing 'pitch))
          (define steps (+ -6 (ly:pitch-steps p1)))
          (define radToDeg (* 180 (/ 1 3.141592653589793)))
          #{ #pitchthing ^\markup {
            \postscript
```

1. Carin Levine and Christina Mitropoulos-Bott, *The techniques of flute playing = Die Spieltechnik der Flöte* (Kassel ; New York: Bärenreiter, 2003), 18.

```
            #(string-append "gsave newpath 0.2 setlinewidth 1.15 "
                            (number->string
                                    (+ -2.5 (* 0.5 steps))) " moveto "
                            (number->string
                                    (* 0.5 width)) " 4 rlineto "
                            (number->string
                                    (* 0.5 width)) " -4 rlineto
                    stroke
                    newpath
                    0.1 setlinewidth "
                            (number->string (+ 1.15 width)) " "
                            (number->string (+ -2.55 (* 0.5 steps)))
                            " moveto "
                            (number->string
                             (* radToDeg (atan (/ (* width 0.5) 4))))
                            " rotate
                    0 -1 rlineto
                    -0.35 1 rlineto
                    0.7 0 rlineto
                    -0.35 -1 rlineto
                    closepath
                    fill
                    grestore
                    ")
          } #})

\score {
  {
    \jet e'2^\markup {\fontsize #-5 {[jet]}} #8
    \jet c'4 #3
    \stemDown  \jet b'8 #1.5
    \jet d'8 #1.5
  }

  \layout {
    \context {
      \Score proportionalNotationDuration = #(ly:make-moment 1/10)
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```

Table of Contents

5.2 Line as a Notehead



5.2.1 Description

These functions replace an ordinary notehead with a dashed or a continuous line. For the continuous line, it is possible to adjust the beginning and ending thicknesses.

5.2.2 Grammar

```
\dashedLineNotehead NOTE1 PITCH #x-dist
\modularLineNotehead NOTE1 PITCH #beginningThick #endingThick #x-dist
```

NB

1. `NOTE1` specifies with which note the line starts. If necessary, the duration must be set, as well.
2. `PITCH` specifies with which pitch the line ends. Enter only the pitch; this information is used to determine the angle of the line, and it has no effect in displaying the rhythm.
3. `x-dist` specifies how long the line is.
4. `beginningThick` (for `modularLineNotehead` only) specifies how thick the beginning part of the line should be. `#15` gives a thin line, similar to the `\dashedLineNotehead` line. `#100` is as thick as a space between two neighboring lines of a staff.
5. `endingThick` (for `modularLineNotehead` only) specifies how thick the ending part of the line should be. `#15` gives a thin line, similar to the `\dashedLineNotehead` line. `#100` is as thick as a space between two neighboring lines of a staff.

5.2.3 Code

```
1
2 % See the entry on "Noteheadless" for its code;
3 % it is required for the snippet to run properly.
4
5 dashedLineNotehead =
6 #(define-music-function
7   (beginning end x-distance) (ly:music? ly:music? number?)
```

```
  (let*
   (
     (p1 (ly:music-property beginning 'pitch))
     (p2 (ly:music-property end 'pitch))
     (steps
      (-
       (+ (* 7 (ly:pitch-octave p2)) (ly:pitch-notename p2))
       (+ (* 7 (ly:pitch-octave p1)) (ly:pitch-notename p1))
       )
      )
     )
   #{
     {

       \once \override Voice.NoteHead.stencil = #ly:text-interface::print
       \once \override Voice.NoteHead.stem-attachment = #'(0 . 0)
       \once \override Staff.LedgerLineSpanner.stencil = ##f
       \once \override Voice.NoteHead.text = \markup         {
         % \translate #(cons 0 0)
         \postscript
         #(string-append
           "newpath 1 setlinecap
              0.15 setlinewidth
              0 0 moveto
              [.4 .4 .4 .4] 3 setdash "
           (number->string x-distance)  " " (number->string (* steps 0.5))
           " rlineto stroke"
           )
       }
       #beginning
       \revert Voice.NoteHead.stencil
       \revert Staff.LedgerLineSpanner.stencil
     }
   #})
  )


modularLineNotehead =
#(define-music-function
  (beginning end beginningThickness endingThickness x-distance)
  (ly:music? ly:music? number? number? number?)
```

```
49    (let*
50     (
51       (p1 (ly:music-property beginning 'pitch))
52       (p2 (ly:music-property end 'pitch))
53       (steps
54        (-
55         (+ (* 7 (ly:pitch-octave p2)) (ly:pitch-notename p2))
56         (+ (* 7 (ly:pitch-octave p1)) (ly:pitch-notename p1))
57         )
58        )
59       )
60     #{
61       {
62
63         \once \override Voice.NoteHead.stencil = #ly:text-interface::print
64         \once \override Voice.NoteHead.stem-attachment = #'(0 . 0)
65         \once \override Voice.LedgerLineSpanner.transparent = ##t
66         \once \override Voice.NoteHead.text = \markup         {
67           \postscript
68           #(string-append
69             "newpath 1 setlinecap 0.1 setlinewidth -0.05 0 moveto 0 "
70             (number->string (* beginningThickness 0.005)) " rlineto "
71             (number->string x-distance) " "
72             (number->string (+ (- (* endingThickness 0.005)
73                                   (* beginningThickness 0.005) )
74                                (* steps 0.5)))
75             " rlineto 0 "
76             (number->string  (* endingThickness -0.01))  " rlineto "
77             (number->string (* -1 x-distance))  " "
78             (number->string (- (* endingThickness 0.005)
79                                (* beginningThickness 0.005)
80                                (* steps 0.5)))
81             " rlineto
82                closepath
83                fill"
84             )
85         }
86         #beginning
87         \revert Voice.NoteHead.stencil
88         \revert Staff.LedgerLineSpanner.stencil
89       }
```

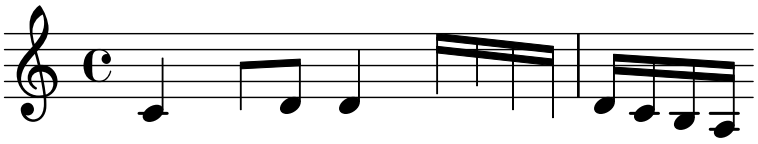
```
    #})
  )


\score {
  {
    \omit Staff.Clef
    \dashedLineNotehead g'4 a' #6
    \modularLineNotehead a' d'' #15 #150 #6
    \override TupletNumber.text = #tuplet-number::calc-fraction-text

    \stemUp  \tuplet 5/4 {
      \modularLineNotehead d''8 b' #150 #50 #2.5
      \modularLineNotehead b' f'' #50 #175 #2.5
      \modularLineNotehead f'' a' #175 #70 #2.5
      \modularLineNotehead a' c'' #70 #120 #2.5
      \modularLineNotehead c'' c' #120 #15 #3.5
    }
    |
    \modularLineNotehead c'4 c' #15 #15 #12
    \noteheadless c'
    \dashedLineNotehead c' c' #5
  }

  \layout {
    \context {
      \Score proportionalNotationDuration = #(ly:make-moment 1/10)
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}


```

5.2.4 Discussion

See Prescriptive Notation for String Instruments for a possible use of this notehead.

Table of Contents

5.3 Noteheadless



5.3.1 Description

This snippet is hardly my own idea, as I largely quoted this technique from one of the snippets available on LSR.[2] However, I list it here because:

1. it took a while for me to find the workaround for maintaining the musical spacing as a result of omitting noteheads. It is worth noting that because merely disabling `NoteHead.stencil` will render the spacing to be squished, the approach of specifying `##t` for `NoteHead.transparent` (which itself will *not* eliminate the ledger lines) then `##t` for `NoteHead.no-ledgers` is effective in maintaining the general spacing.
2. I use this in conjunction with other notehead alterations, e.g. Line as a notehead.

5.3.2 Grammar

```
\noteheadless NOTE
\noteheadlessOn NOTE
\noteheadlessOff
```

NB

1. `\noteheadless` affects only one note immediately following.
2. For a group of notes, use `\noteheadlessOn` to toggle on the function. `\noteheadlessOff` will toggle off the function.

5.3.3 Code

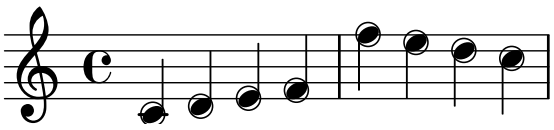
```
1
2  %% Inspired by:
3  %% http://lsr.di.unimi.it/LSR/Item?id=796
4
5
6  noteheadless = {
7    \once \override Voice.NoteHead.transparent = ##t
8    \once \override Voice.NoteHead.no-ledgers = ##t
```

2. See: http://lsr.di.unimi.it/LSR/Item?id=796

```
}

noteheadlessOn = {
  \override Voice.NoteHead.transparent = ##t
  \override Voice.NoteHead.no-ledgers = ##t
}
noteheadlessOff = {
  \revert Voice.NoteHead.transparent
  \revert Voice.NoteHead.no-ledgers
}


{
  c'4 \noteheadless c'8 d' d'4
  \noteheadlessOn e'16 f' c' b |
  \noteheadlessOff d' c' b a
}

```

Table of Contents

5.4 Slap Tongue, Type A



5.4.1 Description

In my music, I use encircled noteheads to denote slap tongues. Type A, encircled filled notehead, is used for a slap tongue with a regular note immediately following.

5.4.2 Grammar

`\slapA NOTE`

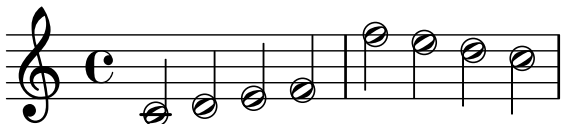
NB It only affects one note, owing to the `\once \override` functions within the code.

5.4.3 Code

```
slapA = #(define-music-function (note)   (ly:music?)
            #{ \once \override Voice.NoteHead.stencil =
               #ly:text-interface::print
               \once \override Voice.NoteHead.text =
               \markup {
                 \concat {
                   \musicglyph "noteheads.s2"
                   \postscript "newpath
                   -0.675 0.025 0.75 0 360 arc
                   closepath stroke"
                 }
               }
               $note #})

{
  \slapA c'4 \slapA d' \slapA e' \slapA f'
  \slapA f'' \slapA e'' \slapA d'' \slapA c''
}

```

Table of Contents

5.5 Slap Tongue, Type B



5.5.1 Description

In my music, I use encircled noteheads to denote slap tongues. Type B, encircled hollow notehead, is used for a slap tongue with an air sound immediately following.

5.5.2 Grammar

`\SlapB NOTE`

NB It only affects one note, owing to the `\once \override` functions within the code.

5.5.3 Code

```
slapB = #(define-music-function (note)   (ly:music?)
            #{ \once \override Voice.NoteHead.stencil =
               #ly:text-interface::print
               \once \override Voice.NoteHead.text =
               \markup {
                 \concat {
                   \musicglyph "noteheads.s1"
                   \postscript "newpath
                   -0.675 0.025 0.75 0 360 arc
                   closepath stroke"
                 }
               }
               $note #})
{
  \SlapB c'4 \SlapB d' \SlapB e' \SlapB f'
  \SlapB f'' \SlapB e'' \SlapB d'' \SlapB c''
}

```

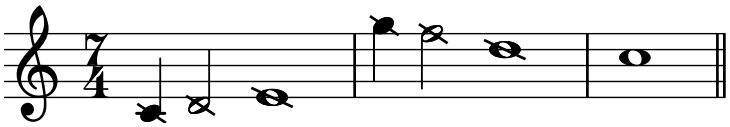
5.5.4 Discussion

As the musical example shows, when the Type B Slap Tongue notehead is applied to a quarter note, it could invite confusion in terms of rhythm. As a slap tongue itself is a

short sound, I only use the slap tongue noteheads on eighth notes or shorter note durations.

Table of Contents

5.6 Slashed Notehead



5.6.1 Description

Noteheads with backslashes applied.[3] I use this notehead to indicate, for example, notes on the piano whose strings are prepared, thus producing pitch/sound different from what is expected normally.

5.6.2 Grammar

```
\slashNote NOTE
\slashNoteOn NOTE
\slashNoteOff
```

NB `\slashNote` only affects one note, owing to the `\once \override` functions within the code. For a group of notes to have slashes applied, use `\slashNoteOn`. `\slashNoteOff` cancels the application.

5.6.3 Code

```
1
2  % Inspired by the code provided by Jean Abou Samra
3  % https://lists.gnu.org/archive/html/lilypond-user/2022-11/msg00333.html
4
5  slashNote =
6  \once \override Voice.NoteHead.stencil =
7  #(grob-transformer
8    'stencil
9    (lambda (grob original)
10     (let* ((added-markup
11             #{
12               \markup \general-align #Y #CENTER
13               #(case (ly:grob-property grob 'duration-log)
14                  ((0) #{ \markup \concat {
15                    \musicglyph "noteheads.s0"
16                    \postscript
```

3. The code provided by Jean Abou Samra in the following discussion thread on lilypond-user was very helpful in creating this code: https://lists.gnu.org/archive/html/lilypond-user/2022-11/msg00333.html

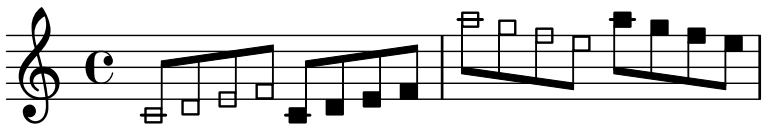
```
                    "gsave
                     0.17 setlinewidth
                     -2.3 0.6 moveto
                     0.3 -0.6 lineto
                     stroke
                     grestore"
                       } #})

                  ((1) #{ \markup \concat {
                    \musicglyph "noteheads.s1"
                    \postscript
                    "gsave
                     0.17 setlinewidth
                     -1.5 0.6 moveto
                     0.3 -0.6 lineto
                     stroke
                     grestore"
                       } #})

                  ((2) #{ \markup \concat {
                    \musicglyph "noteheads.s2"
                    \postscript
                    "gsave
                     0.17 setlinewidth
                     -1.5 0.6 moveto
                     0.3 -0.6 lineto
                     stroke
                     grestore"
                       } #}))
             #})
            (added-stencil (grob-interpret-markup grob added-markup)))
      (if (ly:stencil? original)
          (ly:stencil-add original added-stencil)
          added-stencil))))


slashNoteOn =
\override Voice.NoteHead.stencil =
#(grob-transformer
  'stencil
```

```
58    (lambda (grob original)
59      (let* ((added-markup
60              #{
61                \markup \general-align #Y #CENTER
62                #(case (ly:grob-property grob 'duration-log)
63                   ((0) #{ \markup \concat {
64                     \musicglyph "noteheads.s0"
65                     \postscript
66                     "gsave
67                      0.17 setlinewidth
68                      -2.3 0.6 moveto
69                      0.3 -0.6 lineto
70                      stroke
71                      grestore"
72                        } #})
73                   ((1) #{ \markup \concat {
74                     \musicglyph "noteheads.s1"
75                     \postscript
76                     "gsave
77                      0.17 setlinewidth
78                      -1.5 0.6 moveto
79                      0.3 -0.6 lineto
80                      stroke
81                      grestore"
82                        } #})
83                   ((2) #{ \markup \concat {
84                     \musicglyph "noteheads.s2"
85                     \postscript
86                     "gsave
87                      0.17 setlinewidth
88                      -1.5 0.6 moveto
89                      0.3 -0.6 lineto
90                      stroke
91                      grestore"
92                        } #}))
93              #})
94             (added-stencil (grob-interpret-markup grob added-markup)))
95        (if (ly:stencil? original)
96            (ly:stencil-add original added-stencil)
97            added-stencil))))
98
```

```
99
100 slashNoteOff = \revert Voice.NoteHead.stencil
101
102 {
103   \time 7/4
104   \slashNote c'4
105   \slashNote d'2
106   \slashNote e'1
107   \slashNoteOn g''4 f''2 d''1
108   \slashNoteOff c''1 \bar "||"
109 }
```

Table of Contents

5.7 Square Notehead



5.7.1 Description

Filled and hollow square noteheads.

5.7.2 Grammar

```
\squareHollowNotehead NOTE
\squareHollowNoteheadOn NOTES
\squareHollowNoteheadOff
\squareFilledNotehead NOTE
\squareFilledNoteheadOn NOTES
\squareFilledNoteheadOff

\slashNoteOn NOTE
\slashNoteOff
```

NB `\squareHollowNotehead` and `\squareFilledNotehead` only affect one note, owing to the `\once \override` functions within the code. For a group of notes, use `\squareHollowNoteheadOn` or `\squareFilledNoteheadOn`. `\squareHollowNoteheadOff` and `\squareFilledNoteheadOff` cancel the application.

5.7.3 Code

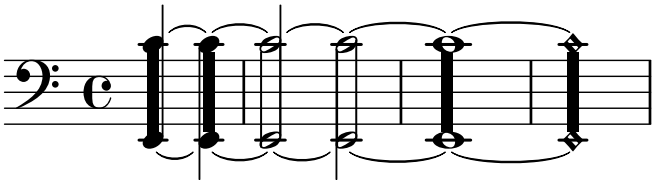
```
\version "2.24.4"

% See also: https://lsr.di.unimi.it/LSR/Item?id=516

squareHollowNoteheadDesign =
#(ly:make-stencil '(path 0.15 (moveto  0.05 0.425
                                       rlineto 1. 0
                                       rlineto 0 -0.875
                                       rlineto -1. 0
                                       closepath)
                         )
                  (cons -0.025 1.125)
                  (cons -1 1))
```

```
14
15 squareHollowNotehead =
16 #(define-music-function (note) (ly:music?)
17    #{\once \override Voice.NoteHead.stencil =
18      \squareHollowNoteheadDesign $note #})
19
20 squareHollowNoteheadOn =
21 #(define-music-function (note) (ly:music?)
22    #{\override Voice.NoteHead.stencil =
23      \squareHollowNoteheadDesign $note #})
24
25 squareHollowNoteheadOff = \revert Voice.NoteHead.stencil
26
27 squareFilledNoteheadDesign =
28 #(ly:make-stencil '(path 0.15 (moveto  0.05 0.425
29                                        rlineto 1. 0
30                                        rlineto 0 -0.875
31                                        rlineto -1. 0
32                                        closepath)
33                           round
34                           round
35                           #t)
36                   (cons -0.025 1.125)
37                   (cons -1 1))
38
39
40 squareFilledNotehead =
41 #(define-music-function (note) (ly:music?)
42    #{\once \override Voice.NoteHead.stencil =
43      \squareFilledNoteheadDesign $note #})
44 squareFilledNoteheadOn =
45 #(define-music-function (note) (ly:music?)
46    #{\override Voice.NoteHead.stencil =
47      \squareFilledNoteheadDesign $note #})
48
49 squareFilledNoteheadOff = \revert Voice.NoteHead.stencil
50
51 {
52   \squareHollowNotehead c'8
53   \squareHollowNoteheadOn d' e' f'
54   \squareHollowNoteheadOff
```

```
55    \squareFilledNotehead c'8
56    \squareFilledNoteheadOn d' e' f'
57    \squareFilledNoteheadOff
58    \squareHollowNotehead a''8
59    \squareHollowNoteheadOn g'' f'' e''
60    \squareHollowNoteheadOff
61    \squareFilledNotehead a''8
62    \squareFilledNoteheadOn g'' f'' e''
63    \squareFilledNoteheadOff
64  }
```

Table of Contents

5.8 Tone Cluster



5.8.1 Description

Inspired by the tone cluster notation of Henry Cowell and others. See **Discussion**.

5.8.2 Grammar

```
\toneClusterBar NOTE1 NOTE2 yOffset yLengthAdjust
\toneClusterBarHollow NOTE1 NOTE2 yOffset yLengthAdjust
\toneClusterBarWhole NOTE1 NOTE2 yOffset yLengthAdjust
```

NB

1. The order of pitch boundaries as shown by `NOTE1` and `NOTE2` does not matter; `NOTE1` can be upper or lower pitch boundary, and vice versa for `NOTE2`. See **Code**.

2. `yOffset` indicates where the upper part of the cluster sign begins. When set to `#0`, it starts right at the top line of the ordinary 5-line staff. Each positive/negative integer will bring the beginning point up/down by a space of two neighboring lines of the staff.

3. `yLengthAdjust` indicates any value by which the cluster bar may be extended or reduced. When set to `#0`, the cluster bar will be as long as the distance between the lower boundary of the upper notehead and upper boundary of the lower notehead. Each positive/negative integer will add/reduce the length of the bar by a space of two neighboring lines of the staff.

   For this reason, when the tone cluster sign is applied to a quarter-note dyad, you may wish to set the upper part of the cluster bar right in the middle of the notehead. In the snippet shown, the first cluster's `yOffset` is set to `#1`. `yLengthAdjust` is also set to `#1`, meaning that the cluster bar will go down to the center of the lower notehead. The second cluster intentionally shows what happens when the bar only touches the two boundaries of the noteheads.

4. `\toneClusterBarHollow` shows the notation (quite à la Cowell) specifically for hollowed noteheads. Some people may prefer this notation, instead.

5. `\toneClusterBarWhole` is specifically for the tone cluster notation as applied to a whole-note dyad, owing to width being wider than the quarter or half noteheads.

6. These functions may be used in tandem with other noteheads, as well as ties. See **Code**.

5.8.3 Code

```
1
2  toneClusterBar =
3  #(define-music-function (note1 note2 yOffset yLengthAdjust)
4     (ly:music? ly:music? number? number?)
5     (let* (
6             (note1p (ly:music-property note1 'pitch))
7             (note2p (ly:music-property note2 'pitch))
8             (note1pnumber (+ (* 7 (ly:pitch-octave note1p))
9                              (ly:pitch-notename note1p)))
10            (note2pnumber (+ (* 7 (ly:pitch-octave note2p))
11                             (ly:pitch-notename note2p)))
12            (pitchDistance (abs (- note1pnumber note2pnumber)))
13            )
14       #{
15         < #note1
16         #note2 > ^\markup {
17           \postscript
18           #(string-append
19             "gsave
20              newpath
21              0.3 " (number->string (- yOffset 0.5)) " moveto
22              0.7 0 rlineto
23              0 " (number->string (- (* -0.5 pitchDistance)
24                                     (- yLengthAdjust 1))) " rlineto
25              -0.7 0 rlineto
26              closepath
27              fill
28              grestore")
29         }
30       #}
31       )
32    )
33
34
```

```
35 toneClusterBarHollow =
36 #(define-music-function (note1 note2 yOffset yLengthAdjust)
37    (ly:music? ly:music? number? number?)
38    (let* (
39            (note1p (ly:music-property note1 'pitch))
40            (note2p (ly:music-property note2 'pitch))
41            (note1pnumber (+ (* 7 (ly:pitch-octave note1p))
42                             (ly:pitch-notename note1p)))
43            (note2pnumber (+ (* 7 (ly:pitch-octave note2p))
44                             (ly:pitch-notename note2p)))
45            (pitchDistance (abs (- note1pnumber note2pnumber)))
46            )
47      #{
48        < #note1
49        #note2 > ^\markup {
50          \postscript
51          #(string-append
52            "gsave
53             newpath
54             0.1 " (number->string (- yOffset 0.5)) " moveto
55             0 " (number->string (- (* -0.5 pitchDistance)
56                                    (+ 0.5 yLengthAdjust))) " rlineto
57             0.125 setlinewidth
58             1.3 "(number->string (+ 0.75 (- yOffset 0.5))) " moveto
59             0 " (number->string (- (* -0.5 pitchDistance)
60            (+ 0.75 yLengthAdjust))) " rlineto
61             stroke
62             grestore")
63        }
64      #}
65      )
66    )
67
68
69 toneClusterBarWhole =
70 #(define-music-function (note1 note2 yOffset yLengthAdjust)
71    (ly:music? ly:music? number? number?)
72    (let* (
73            (note1p (ly:music-property note1 'pitch))
74            (note2p (ly:music-property note2 'pitch))
75            (note1pnumber (+ (* 7 (ly:pitch-octave note1p))
```

```
                              (ly:pitch-notename note1p)))
            (note2pnumber (+ (* 7 (ly:pitch-octave note2p))
                              (ly:pitch-notename note2p)))
            (pitchDistance (abs (- note1pnumber note2pnumber)))
            )
      #{
        < #note1
        #note2 > ^\markup {
          \postscript
          #(string-append
            "gsave
             newpath
             0.125 setlinewidth
             0.55 " (number->string (- yOffset 0.5)) " moveto
             0 " (number->string (- (* -0.5 pitchDistance)
                                    (- yLengthAdjust 1))) " rlineto
             0.75 0 rlineto
             0 " (number->string (abs (- (* -0.5 pitchDistance)
                 (- yLengthAdjust 1)))) " rlineto
             closepath fill
             grestore")
        }
      #}
      )
    )


{
  \time 4/4
  \partial 2
  \clef "F"
  \stemUp \toneClusterBar c'4~ e,~ #1 #1
  \stemDown \toneClusterBar e,~ c'4~ #0.5 #0
  \stemUp \toneClusterBarHollow c'2~ e,~ #0.5 #-0.5
  \stemDown \toneClusterBarHollow c'2~ e,~ #0.5 #-0.5
  \toneClusterBarWhole c'1~ e,~ #0.5 #0
  \toneClusterBar c'1~\harmonic e,~\harmonic #0.5 #0
}
```

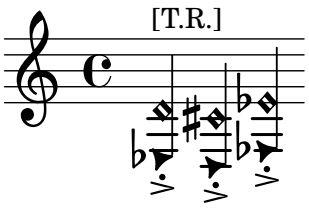
5.8.4 Discussion

There have been some discussions on `lilypond-user` mailing list in the past that readers may consult for further ideas on implementing different types of tone cluster notation:

- https://lists.gnu.org/archive/html/lilypond-user/2008-10/msg00484.html (This one in particular lists other notational conventions established by other composers)
- https://lists.gnu.org/archive/html/lilypond-user/2020-12/msg00130.html

Table of Contents

5.9 Tongue Ram (for flute)



5.9.1 Description

Implementation of the tongue ram notation, as described in *The Techniques of Flute Playing* by Carin Levine and Christina Mitropoulos-Bott.[4]

5.9.2 Grammar

```
\tgrWithIndication NOTE
\tgr NOTE
```

NB

1. `\language "english"` needs to be specified.
2. `\tgr` and `\tgrWithIndication` are followed by a pitch to be fingered on the instrument. The code will copy and reproduce a resultant pitch a major seventh down. Use `\tgrWithIndication` for showing the markup with the indication "T.R." (tongue ram). For more details, see: FluteXpansions.

5.9.3 Code

```
tgrWithIndication = #(define-music-function (note1) (ly:music?)
       (let*         (
                (p1 #{ #(ly:music-deep-copy note1) \harmonic #})
                (p2 #{ \transpose c df, #(ly:music-property note1 'pitch)#})
                (d1  (ly:music-property note1 'duration))
                )
          #{ < $p1
             \single \override NoteHead.stencil = #ly:text-interface::print
             \single \override NoteHead.text =
             \markup \musicglyph "noteheads.s2triangle"
             %\single \override Stem.stencil
             $p2 > $d1 ^\markup {\override #'(font-size . -2) {[T.R.]} }  #}
```

4. Levine and Mitropoulos-Bott, *The techniques of flute playing = Die Spieltechnik der Flöte*, 28.

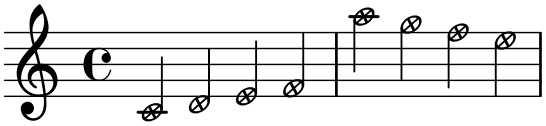
```
          ))
tgr = #(define-music-function (note1) (ly:music?)
          (let*         (
                   (p1 #{ #(ly:music-deep-copy note1) \harmonic #})
                   (p2 #{ \transpose c df, #(ly:music-property note1 'pitch)#})
                   (d1  (ly:music-property note1 'duration))
                   )
            #{ < $p1
               \single \override NoteHead.stencil = #ly:text-interface::print
               \single \override NoteHead.text =
               \markup \musicglyph "noteheads.s2triangle"
               %\single \override Stem.stencil
               $p2 > $d1  #}
            ))

{\language "english" \tgrWithIndication d'4-.-> \tgr cs'4-.-> \tgr ef'4-.->}
```

5.9.4 Discussion

I want to improve this code so that I can add markups to the note. It is slightly awkward at the moment.

Table of Contents

5.10 X In A Hollow Notehead



5.10.1 Description

While LilyPond Notation Reference provides an example of an X-in-a-circle notehead, its shape differs from the regular notehead.[5] This implementation simulates a hollow notehead with which the X notehead is combined.

5.10.2 Grammar

```
\cirX NOTE
```

5.10.3 Code

```
% Stem attachment function inspired by:
% https://lsr.di.unimi.it/LSR/Snippet?id=518
cirX = #(define-music-function (note) (ly:music?)
          #{
            \temporary \override NoteHead.stencil =
            #ly:text-interface::print
            \temporary \override NoteHead.text =
            \markup
            \translate #'(0.6 . 0)
            \pad-x #-0.22
            \rotate #35
            \scale #'(1 . 0.65)
            \combine \combine \combine \combine
            \override #'(thickness . 2)
            \draw-line #'(0.05 . 0.6)
            \override #'(thickness . 2)
            \draw-line #'(-0.05 . -0.6)
            \override #'(thickness . 2)
            \draw-line #'(0.6 . 0.1 )
            \override #'(thickness . 2)
            \draw-line #'(-0.6 . -0.1 )
            \draw-circle #0.65 #0.175 ##f
```

5. https://lilypond.org/doc/v2.24/Documentation/notation/modifying-stencils

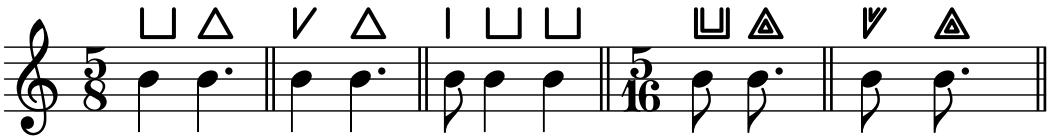
```
23
24              \temporary \override NoteHead.stem-attachment =
25              #(lambda (grob)
26                 (let* ((stem (ly:grob-object grob 'stem))
27                        (dir (ly:grob-property stem 'direction UP))
28                        (is-up (eqv? dir UP)))
29                   (cons dir (if is-up 0.2 -0.2))))
30              #note
31              \revert NoteHead.stencil
32              \revert NoteHead.text
33              \revert NoteHead.stem-attachment
34            #})
35  {
36    \cirX c'4 \cirX d' \cirX e' \cirX f'
37    \cirX a''4 \cirX g'' \cirX f'' \cirX  e''
38  }
```

Table of Contents

Chapter 6

Markups

6.1 Conducting Patterns



6.1.1 Description

Conducting patterns. While there are several examples of conducting patterns available on LSR,[1] the conducting shapes in my implementation are not affected by the horizontal length of given durations.

6.1.2 Grammar

```
NOTE \condOne
NOTE \condTwoA
NOTE \condTwoB
NOTE \condThree
NOTE \condDoubleTwoA
NOTE \condDoubleTwoB
NOTE \condDoubleThree
```

1. See: https://lsr.di.unimi.it/LSR/Item?id=523 and https://lsr.di.unimi.it/LSR/Item?id=259

6.1.3 Code

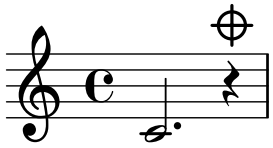
```
1
2  condOnePattern =
3  #'((moveto 0.25 1.75)
4     (rlineto 0 -1.75))
5
6  condTwoPatternA =
7  #'((moveto 0.25 1.75)
8     (rlineto 0 -1.75)
9     (rlineto 2 0)
10    (rlineto 0 1.75))
11
12 condDoubleTwoPatternA =
13 #'((moveto 0.25 1.75)
14    (rlineto 0 -1.75)
15    (rlineto 2 0)
16    (rlineto 0 1.75)
17    (moveto 0.65 1.75)
18    (rlineto 0 -1.35)
19    (rlineto 1.2 0)
20    (rlineto 0 1.35))
21
22 condTwoPatternB =
23 #'((moveto 0.25 1.75)
24    (rlineto 0 -1.75)
25    (rlineto 1.25 1.75))
26
27 condDoubleTwoPatternB =
28 #'((moveto 0.25 1.75)
29    (rlineto 0 -1.75)
30    (rlineto 1.25 1.75)
31    (moveto 0.6 1.75)
32    (rlineto 0 -0.7)
33    (rlineto 0.5 0.7))
34
35 condThreePattern =
36 #'((moveto 1.15 1.75)
37    (rlineto -1 -1.75)
38    (rlineto 2 0)
39    (closepath))
```

```
40
41  condDoubleThreePattern =
42  #'((moveto 1.15 1.75)
43     (rlineto -1 -1.75)
44     (rlineto 2 0)
45     (closepath)
46     (moveto 1.15 1.05)
47     (rlineto -0.385 -0.7)
48     (rlineto 0.75 0)
49     (closepath))
50
51
52  condOne = ^\markup {
53    \override #'(line-join-style . round)
54    \path #0.25 #condOnePattern
55  }
56
57  condTwoA = ^\markup {
58    \override #'(line-join-style . round)
59    \path #0.25 #condTwoPatternA
60  }
61  condTwoB = ^\markup {
62    \override #'(line-join-style . round)
63    \path #0.25 #condTwoPatternB
64  }
65  condDoubleTwoA = ^\markup {
66    \override #'(line-join-style . round)
67    \path #0.25 #condDoubleTwoPatternA
68  }
69
70  condDoubleTwoB = ^\markup {
71    \override #'(line-join-style . round)
72    \path #0.25 #condDoubleTwoPatternB
73  }
74
75  condThree = ^\markup {
76    \override #'(line-join-style . round)
77    \path #0.25 #condThreePattern
78  }
79
80  condDoubleThree = ^\markup {
```

```
  \override #'(line-join-style . round)
  \path #0.25 #condDoubleThreePattern
}

%% Source inspired by
%% and adapted from: http://lsr.di.unimi.it/LSR/Item?id=629
spacerVoice = \new Voice {
  \override MultiMeasureRest.transparent = ##t
  \override MultiMeasureRest.minimum-length = #14
  R16*5
}


\score {
  {
    \time 5/8
    b'4 \condTwoA b'4. \condThree \bar "||"
    b'4 \condTwoB b'4. \condThree \bar "||"
    b'8 \condOne b'4 \condTwoA b'4 \condTwoA \bar "||"
    \time 5/16
    << {b'8 \condDoubleTwoA b'8. \condDoubleThree}
            \spacerVoice >> \bar "||"
    << {b'8 \condDoubleTwoB b'8. \condDoubleThree}
            \spacerVoice >> \bar "||"
  }

}

```

Table of Contents

6.2 Mute Sign



6.2.1 Description

Implementation of the mute sign, used to indicate that vibrating strings must be dampened at a specified moment. Its provenance can be traced back to Carlos Salzedo's *Modern Study of the Harp*.[2]

6.2.2 Grammar

```
NOTE/REST^\mutesign
```

6.2.3 Code

```
mutesign = \markup {
  \translate #'(0.5 . 0)
  \postscript

  "newpath
0.2 setlinewidth
1 setlinecap
0 0 moveto
0 2.5 rlineto
-1.25 1.25 moveto
2.5 0 rlineto
stroke
newpath
0 1.25 0.85 0 360 arc
stroke"

{ c'2. r4^\mutesign }

```

Table of Contents

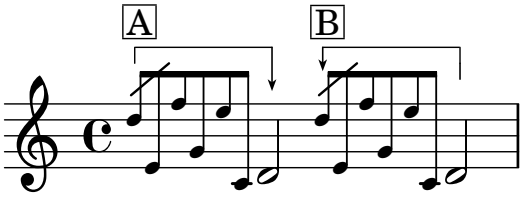
2. Carlos Salzedo, *L'étude moderne de la harpe... Modern study of the harp* (New York - Boston, G. Schirmer, 1921), 19.

Chapter 7

Spanners

This chapter covers snippets that take advantages of spanners (text, line, etc.) in one way or another. Because functions such as `\startTextSpan` and `\stopTextSpan` activate and deactivate these snippets, caution must be paid when using more than one of them at the same time. See Example in **Combinations** to avoid conflicts between or among the spanner snippets.

7.1 Grace Note Brackets



7.1.1 Description

Replication of grace note brackets seen in scores by Pierre Boulez (e.g. *Sur Incises*,[1] *...explosante-fixe...*[2]). Bracket A in the example shows that the grace notes are to be played *before* the beat to which they are applied. Whereas Bracket B shows that the grace notes are to be played *on* the beat to which they are applied.

1. Pierre Boulez, *Sur incises : pour trois pianos, trois harpes et trois percussions-claviers (1996/1998)* (Universal Edition, 1998).
2. Pierre Boulez, *... explosante-fixe ... transitoire VII : (version 1991/93)* (Universal Edition, 1991).

7.1.2 Grammar

```
\graceNoteBeforeBeatOn NOTE
\graceNoteBeforeBeatOff NOTE
\graceNoteAfterBeatOn NOTE
\graceNoteAfterBeatOff NOTE
```

7.1.3 Code

```
\version "2.24.4"

\language "english"

% This code includes snippet for grace note
% slashes, which has been taken from:
% https://lsr.di.unimi.it/LSR/Item?id=1048


graceNoteBeforeBeatOn =
#(define-music-function (starting_note) (ly:music?)
   #{
     \once \override TextSpanner.style = #'line
     \once \override TextSpanner.bound-details.left.text =
     \markup { \draw-line #'(0 . -1) }
     \once \override TextSpanner.bound-details.right.text =
     \markup {
       \postscript
       "newpath 0 0 moveto
0 -2.5 rlineto
stroke
newpath
-0.275 -2 moveto
0.275 -0.75 rlineto
0.275 0.75 rlineto
-0.275 -0.2 rlineto
closepath
fill"
     }
     \once \override TextSpanner.Y-offset = #5
```

```
      \once \override TextSpanner.bound-details.left.padding = #0.5
      \once \override TextSpanner.bound-details.right.padding = #-0.25
      #starting_note
      \startTextSpan
   #})


graceNoteBeforeBeatOff =
#(define-music-function (ending_note) (ly:music?)
   #{
     #ending_note
     \stopTextSpan
   #})


graceNoteAfterBeatOn =
#(define-music-function (starting_note) (ly:music?)
   #{
     \once \override TextSpanner.style = #'line
     \once \override TextSpanner.bound-details.right.text =
     \markup {
       \combine \draw-line #'(0 . -1)
       \postscript "newpath
0 -1 moveto
0 -1 rlineto
stroke"
     }
     \once \override TextSpanner.bound-details.left.text =
     \markup {
       \postscript
       "newpath 0 0 moveto
0 -1 rlineto
stroke
newpath
-0.275 -0.75 moveto
0.275 -0.75 rlineto
0.275 0.75 rlineto
-0.275 -0.2 rlineto
closepath
fill"
     }
```

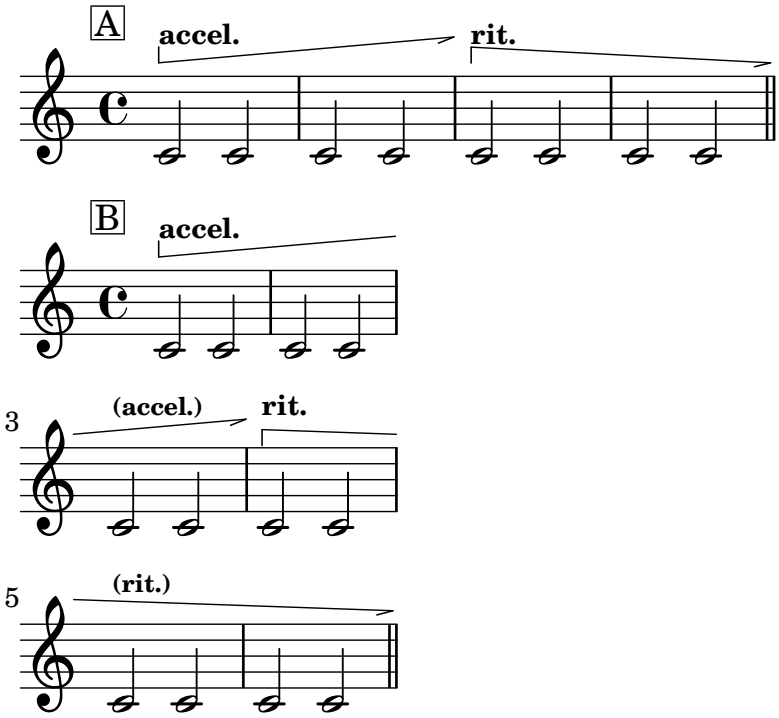
```
72      \once \override TextSpanner.Y-offset = #2
73      \once \override TextSpanner.bound-details.left.padding = #0.5
74      \once \override TextSpanner.bound-details.right.padding = #-0.25
75      #starting_note
76      \startTextSpan
77    #})
78
79
80  graceNoteAfterBeatOff =
81  #(define-music-function (ending_note) (ly:music?)
82     #{
83       #ending_note
84       \stopTextSpan
85     #})
86
87  %%%%%%%%%%%%%%%%%LSR SNIPPET START%%%%%%%%%%%%%%%%%
88
89  #(define (degrees->radians deg)
90     (* PI (/ deg 180.0)))
91
92  slash =
93  #(define-music-function (ang stem-fraction protrusion)
94     (number? number? number?)
95     (remove-grace-property 'Voice 'Stem 'direction)
96     #{
97       \once \override Stem.stencil =
98       #(lambda (grob)
99          (let* ((x-parent (ly:grob-parent grob X))
100                (is-rest? (ly:grob?
101                           (ly:grob-object x-parent 'rest)))
102                (beam (ly:grob-object grob 'beam))
103                (stil (ly:stem::print grob)))
104           (cond
105            (is-rest? empty-stencil)
106            ((ly:grob? beam)
107             (let* ((refp (ly:grob-system grob))
108                    (stem-y-ext (ly:grob-extent grob grob Y))
109                    (stem-length
110                     (- (cdr stem-y-ext) (car stem-y-ext)))
111                    (beam-X-pos (ly:grob-property beam 'X-positions))
112                    (beam-Y-pos (ly:grob-property beam 'positions))
```

```
113                 (beam-slope (/ (- (cdr beam-Y-pos) (car beam-Y-pos))
114                                (- (cdr beam-X-pos) (car beam-X-pos))))
115                 (beam-angle (atan beam-slope))
116                 (dir (ly:grob-property grob 'direction))
117                 (line-dy (* stem-length stem-fraction))
118                 (line-dy-with-protrusions (if (= dir 1)
119                                (+ (* 4 protrusion) beam-angle)
120                                (- (* 4 protrusion) beam-angle)))
121                 (ang (if (> beam-slope 0)
122                          (if (= dir 1)
123                              (+ (degrees->radians ang) (* beam-angle 0.7))
124                              (degrees->radians ang))
125                          (if (= dir 1)
126                              (degrees->radians ang)
127                              (- (degrees->radians ang) (* beam-angle 0.7)))))
128                 (line-dx (/ line-dy-with-protrusions (tan ang)))
129                 (protrusion-dx (/ protrusion (tan ang)))
130                 (corr (if (= dir 1) (car stem-y-ext) (cdr stem-y-ext))))
131             (ly:stencil-add
132              stil
133              (grob-interpret-markup grob
134                                     (markup
135                                       #:translate
136                                       (cons (- protrusion-dx)
137                                       (+ corr
138                                          (* dir
139                                             (- stem-length
140                                                (+ stem-fraction protrusion)))))
141                                       #:override '(thickness . 1.7)
142                                       #:draw-line
143                                       (cons line-dx
144                                        (* dir line-dy-with-protrusions)))))))
145             (else stil))))
146    #})
147
148 startSlashedGraceMusic = {
149   \slash 40 1 0.5
150   \override Flag.stroke-style = #"grace"
151 }
152 stopSlashedGraceMusic = {
153   \revert Flag.stroke-style
```

```
}

startAcciaccaturaMusic = {
  \slash 40 1 0.5
  s1*0(
  \override Flag.stroke-style = #"grace"
}
stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  s1*0)
}
%%%%%%%%%%%%%%%%%% LSR SNIPPET END %%%%%%%%%%%%%%%%%%


{
  \grace {
    \startSlashedGraceMusic
    \graceNoteBeforeBeatOn d''8^\markup{\box A} e' f'' g' e'' c'
  }
  \graceNoteBeforeBeatOff d'2
  \grace {
    \startSlashedGraceMusic
    \graceNoteAfterBeatOn d''8^\markup{\box B} e' f'' g' e'' c'
  }
  \graceNoteAfterBeatOff d'2
}
```

Table of Contents

7.2 Tempo Arrows



7.2.1 Description

Replication of accelerando and rallentando arrows chiefly seen in scores by Tōru Takemitsu.[3] The snippets also handle line break.

7.2.2 Grammar

```
\accelArrow #Line_angle ... \stopTextSpan
\rallArrow #Line_angle ... \stopTextSpan
```

NB

1. `#Line_angle` sets how angled the horizontal line should be. `#5` should be more than sufficient for a short line. When it goes over a line break or it extends for a long time, a smaller number may be recommended, such as `#2`.

3. Examples abound, but see: Tōru Takemitsu, *Fantasma/cantos : for clarinet and orchestra* (Schott ; Schott Japan, 1993) and Tōru Takemitsu, *Les yeux clos II : for piano* (Schott ; Schott Japan, 1990) Other composers from the same publishing company, e.g. Toshio Hosokawa, have also adopted variants of the arrows in their music.

2. These commands only set the tempo arrows; as such, indications such as `accel.` and `rall.` need to be added separately.

7.2.3 Code

```
\version "2.24.4"

% freely modified from: https://lsr.di.unimi.it/LSR/Item?id=1168
% as well as http://lsr.di.unimi.it/LSR/Item?id=1023


accelArrow =
#(define-music-function (line_angle) (number?)

   (define x_value (cos (* (/ 3.14159265358979 180) (- 90 line_angle))))
   (define y_value (sin (* (/ 3.14159265358979 180) (- 90 line_angle))))
   #{
     \tweak direction #up
     \tweak style  #'line
     \tweak thickness  #1
     \tweak to-barline ##t
     \tweak rotation #(list line_angle -1 0 )
     \tweak bound-details.left.stencil #ly:text-interface::print
     \tweak bound-details.left.text \markup \postscript
     #(string-append
       "gsave newpath
0 0 moveto "
       (number->string x_value) " "
       (number->string y_value)
       " rlineto
stroke
grestore")
     \tweak bound-details.left-broken.stencil #ly:text-interface::print
     \tweak bound-details.left-broken.text ##f

     \tweak bound-details.right.stencil #ly:text-interface::print
     \tweak bound-details.right.text \markup \postscript
     "newpath
0 0 moveto
-1 -0.3 rlineto
stroke"
```

```
      \tweak bound-details.right-broken.stencil #ly:text-interface::print
      \tweak bound-details.right-broken.text ##f
      \tweak font-shape  #'upright
      \tweak bound-details.left.padding  #0
      \tweak bound-details.right.padding  #0
      \tweak breakable ##t
      \tweak after-line-breaking ##t

      \startTextSpan
   #})

rallArrow =
#(define-music-function (line_angle) (number?)

   (define x_value (cos (* (/ 3.14159265358979 180) (- 90 line_angle))))
   (define y_value (sin (* (/ 3.14159265358979 180) (- 90 line_angle))))
   #{
     \tweak direction #up
     \tweak style  #'line
     \tweak thickness  #1
     \tweak to-barline ##t
     \tweak rotation #(list (* -1 line_angle) 1 0 )
     \tweak bound-details.left.stencil #ly:text-interface::print
     \tweak bound-details.left.text \markup \postscript
     #(string-append
       "gsave
newpath
0 0 moveto "
       (number->string x_value) " "
       (number->string (* -1 y_value))
       " rlineto
stroke
grestore")
     \tweak bound-details.left-broken.stencil #ly:text-interface::print
     \tweak bound-details.left-broken.text ##f

     \tweak bound-details.right.stencil #ly:text-interface::print
     \tweak bound-details.right.text \markup \postscript
     "newpath
0 0 moveto
-1 -0.3 rlineto
```

```
78  stroke"
79       \tweak bound-details.right-broken.stencil #ly:text-interface::print
80       \tweak bound-details.right-broken.text ##f
81       \tweak font-shape  #'upright
82       \tweak bound-details.left.padding  #0
83       \tweak bound-details.right.padding  #0
84       \tweak breakable ##t
85       \tweak after-line-breaking ##t
86
87       \startTextSpan
88     #})
89
90  \score {
91    \layout {
92      indent = 0
93    }
94    {
95      c'2^\markup{\translate #'(-4 . 2) \box "A"}
96      ^\markup {\translate #'(0 . 1.5) \tiny \bold "accel."}
97              \accelArrow #5   c'2
98      c'2 \after 2 \stopTextSpan c'2
99      c'2 ^\markup {\translate #'(0 . 1.5) \tiny \bold "rit."}
100             \rallArrow #3   c'2
101     c'2 \after 2 \stopTextSpan c'2 \bar "||"
102   }
103 }
104
105 \score {
106   \layout {
107     indent = 0
108     line-width = 40
109   }
110   {
111     c'2^\markup{\translate #'(-4 . 2) \box "B"}
112     ^\markup {\translate #'(0 . 1.5) \tiny \bold "accel."}
113             \accelArrow #5 c'2
114     c'2 c'2
115     c'2^\markup {\translate #'(0 . 1.5) \teeny \bold "(accel.)"}
116             \after 2 \stopTextSpan c'2
117     c'2 ^\markup {\translate #'(0 . 1.5) \tiny \bold "rit."}
118             \rallArrow #2 c'2 \break
```

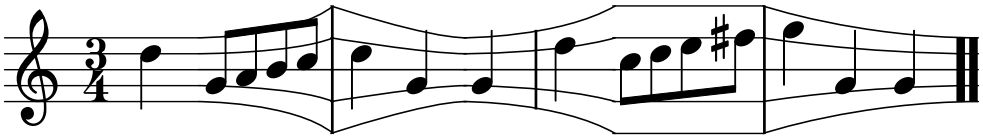
```
119       c'2^\markup {\translate #'(0 . 1.5) \teeny \bold "(rit.)"} c'2
120       c'2 \after 2 \stopTextSpan c'2 \bar "||"
121     }
122   }
```

Table of Contents

Chapter 8

Staff Lines

8.1 Expanding, Shrinking and Bloated Staff Lines



8.1.1 Description

I made this code as a proof of concept after having read some excellent snippets on LSR.[1]

8.1.2 Grammar

```
\expandingStaff #X-length
\shrinkingStaff #X-length
\bloatedStaff
\normalStaff
```

8.1.3 Code

```
1
2  shrinkingStaff =
```

1. See: https://lsr.di.unimi.it/LSR/Item?id=878, https://lsr.di.unimi.it/LSR/Item?id=1005, and https://lsr.di.unimi.it/LSR/Item?id=1007.

```
#(define-music-function
  (staffDist)
  (number?)

  #{
    \stopStaff
    \once  \override Staff.StaffSymbol.stencil = #ly:text-interface::print
    \once  \override Staff.StaffSymbol.text = \markup {
      \postscript #(string-append
          "newpath
          0 4 moveto
          0 4 6 2 " (number->string staffDist) " 2 curveto
          0 2 moveto
          0 2 6 1 " (number->string staffDist) " 1 curveto
          0 0 moveto "
          (number->string staffDist) " 0 lineto
          0 -2 moveto
          0 -2 6 -1 " (number->string staffDist) " -1 curveto
          0 -4 moveto
          0 -4 6 -2 " (number->string staffDist) " -2 curveto
          stroke")


    }
    \override Staff.StaffSymbol.line-positions = #'(-4 -2 0 2 4 )
    \startStaff
  #})

normalStaff = {
  \stopStaff
  \revert Staff.StaffSymbol.line-positions
  \revert Staff.StaffSymbol.stencil
  \startStaff
}

expandingStaff =
#(define-music-function
  (staffDist)
  (number?)

  #{
```

```
44
45      \stopStaff
46      \once  \override Staff.StaffSymbol.stencil = #ly:text-interface::print
47      \once  \override Staff.StaffSymbol.text = \markup {
48        \postscript #(string-append
49            "newpath
50            0 2 moveto
51            0 2 6 2 " (number->string staffDist) " 4 curveto
52            0 1 moveto
53            0 1 6 1 " (number->string staffDist) " 2 curveto
54            0 0 moveto "
55            (number->string staffDist) " 0 lineto
56            0 -1 moveto
57            0 -1 6 -1 " (number->string staffDist) " -2 curveto
58            0 -2 moveto
59            0 -2 6 -2 " (number->string staffDist) " -4 curveto
60            stroke ")
61      }
62
63      \startStaff
64      \override Staff.StaffSymbol.line-positions = #'(-8 -4 0 4 8 )
65    #})
66
67  bloatedStaff = {
68    \stopStaff
69    \override Staff.StaffSymbol.line-positions = #'(-8 -4 0 4 8 )
70    \override Staff.LedgerLineSpanner.stencil = ##f
71    \startStaff}
72
73
74
75  % to adjust the length of the individual barlines, see:
76  % https://lilypond.org/doc/v2.24/Documentation/internals/barline
77
78  {
79
80    \override Staff.LedgerLineSpanner.transparent = ##t
81    \numericTimeSignature
82    \time 3/4
83    \once \override Staff.BarLine.bar-extent = #'(-2 . 2)
84    d''4 \expandingStaff #8.5
```

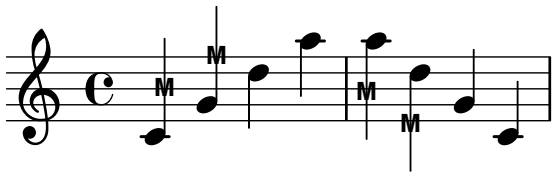
```
85
86    g'8 a' b' c''
87    \once \override Staff.BarLine.bar-extent = #'(-4 . 4)
88    \shrinkingStaff #8.5
89    d''4 g' \expandingStaff #9.5 g'
90    \once \override Staff.BarLine.bar-extent = #'(-2.5 . 2.5)
91
92
93    e''4 \bloatedStaff c''8 d'' e'' fs''
94    \once \override Staff.BarLine.bar-extent = #'(-4 . 4)
95
96    \shrinkingStaff #13.5
97
98    g''4 g' g'
99    \bar ".."
100
101 }
102
103 \layout {
104   \context{
105     \Score   proportionalNotationDuration = #(ly:make-moment 1/6)
106   }
107 }
108
109
```

Table of Contents

Chapter 9

Stems

9.1 ”M” on Stem



9.1.1 Description

This function attaches ”M” to the stem. I have used this to indicate **M**ultiphonics on woodwind instruments in my pieces. This function lengthens the stem in order to give a balanced look, especially combined with stems/flags.

9.1.2 Grammar

```
\MOnStemOn NOTE ...
\MOnStemOff
```

NB `\MOnStemOn` toggles the feature on, while `\MOnStemOff` toggles it off.

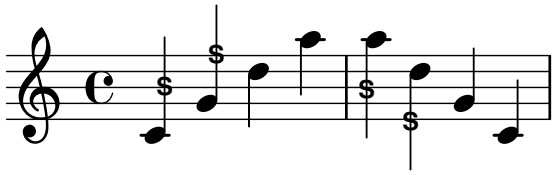
9.1.3 Code

```
1  MOnStemOn = {
2    \override Stem.length = #12
3    \override Stem.details.beamed-lengths = #'(5.5)
4    \override Stem.stencil =
5    #(lambda (grob)
```

```
      (let* ((x-parent (ly:grob-parent grob X))
             (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
        (if is-rest?
            empty-stencil
            (ly:stencil-combine-at-edge
             (ly:stem::print grob)
             Y
             (- (ly:grob-property grob 'direction))
             (grob-interpret-markup grob
                                    (markup
                                     #:center-align
                                     #:teeny #:sans #:bold "M"))
             -3.5))))
}

MOnStemOff = {
  \revert Stem.length
  \revert Stem.details.beamed-lengths
  \revert Stem.stencil
  \revert Flag.stencil
}

{
  \MOnStemOn c'4 g' \MOnStemOff d'' a''
  \MOnStemOn a'' d'' \MOnStemOff g' c'
}
```

Table of Contents

9.2 "S" on Stem



9.2.1 Description

This function attaches "S" to the stem. I have used this to indicate **S**plit tone on clarinet/bass clarinet in my pieces. This function lengthens the stem in order to give a balanced look, especially combined with stems/flags.

9.2.2 Grammar

`\SOnStemOn NOTE ...`
`\SOnStemOff`

NB `\SOnStemOn` toggles the feature on, while `\SOnStemOff` toggles it off.

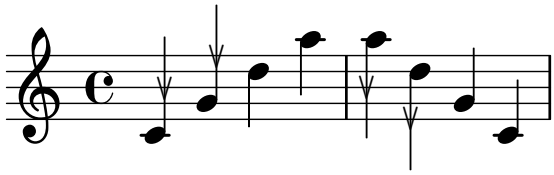
9.2.3 Code

```
SOnStemOn = {
  \override Stem.length = #12
  \override Stem.details.beamed-lengths = #'(5.5)
  \override Stem.stencil =
  #(lambda (grob)
     (let* ((x-parent (ly:grob-parent grob X))
            (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
       (if is-rest?
           empty-stencil
           (ly:stencil-combine-at-edge
            (ly:stem::print grob)
            Y
            (- (ly:grob-property grob 'direction))
            (grob-interpret-markup grob
                                   (markup
                                    #:center-align
                                    #:teeny #:sans #:bold "S"))
            -3.5))))
}

```

```
21 SOnStemOff = {
22   \revert Stem.length
23   \revert Stem.details.beamed-lengths
24   \revert Stem.stencil
25   \revert Flag.stencil
26 }
27
28 {
29   \SOnStemOn c'4 g' \SOnStemOff d'' a''
30   \SOnStemOn a'' d'' \SOnStemOff g' c'
31 }
```

Table of Contents

9.3 ”V” on Stem



9.3.1 Description

This function attaches ”V” to the stem. I have used this to designate a note with a differentiated timbre from others, for example ”brassy tone” for bassoon in my *Gz III* (2019-21) for bass clarinet and bassoon. This function lengthens the stem in order to give a balanced look, especially combined with stems/flags.

9.3.2 Grammar

```
\VOnStemOn NOTE ...
\VOnStemOff
```

NB `\VOnStemOn` toggles the feature on, while `\VOnStemOff` toggles it off.

9.3.3 Code

```
VOnStemOn = {
  \override Stem.no-stem-extend = ##f
  \override Stem.length = #12
  \override Stem.details.beamed-lengths = #'(5.5)
  \override Stem.stencil =
  #(lambda (grob)
     (let* ((x-parent (ly:grob-parent grob X))
            (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
       (if is-rest?
           empty-stencil
           (ly:stencil-combine-at-edge
            (ly:stem::print grob)
            Y
            (- (ly:grob-property grob 'direction))
            (grob-interpret-markup grob
                                   (markup
                                    #:center-align
                                    #:teeny #:sans #:musicglyph "scripts.upbow"))
            -3.5))))
```

```
20  }
21
22  VOnStemOff = {
23    \revert Stem.length
24    \revert Stem.stencil
25    \revert Flag.stencil
26  }
27
28
29  {
30    \VOnStemOn c'4 g' \VOnStemOff d'' a''
31    \VOnStemOn a'' d'' \VOnStemOff g' c'
32  }
```

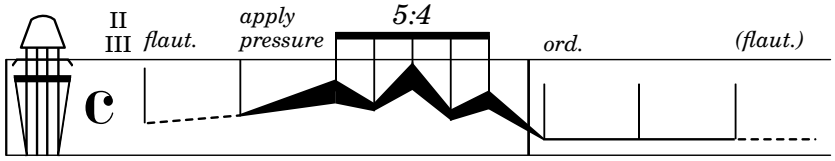
Table of Contents

Chapter 10

Combinations

This chapter presents examples that combine several snippets from the previous chapters. **Variables Used** provides a comprehensive list of all the variables required to generate the snippet. Among these, indented variables indicate "variables of a variable," i.e., dependent variables necessary for the main variables to function. The **Code** section only lists the score portion of the LilyPond code.

10.1 Prescriptive Notation for String Instruments



10.1.1 Description

An example of a prescriptive notation for a string instrument. Vertical placement of the notehead corresponds to the position at which bowing takes place. Horizontally it shows the change of the bow pressure against the string(s).

10.1.2 Variables Used

```
\strPosClef
  \strPosClefDesign
  \strPosClefSize
```

\dashedLineNotehead
\modularLineNotehead
\noteheadless

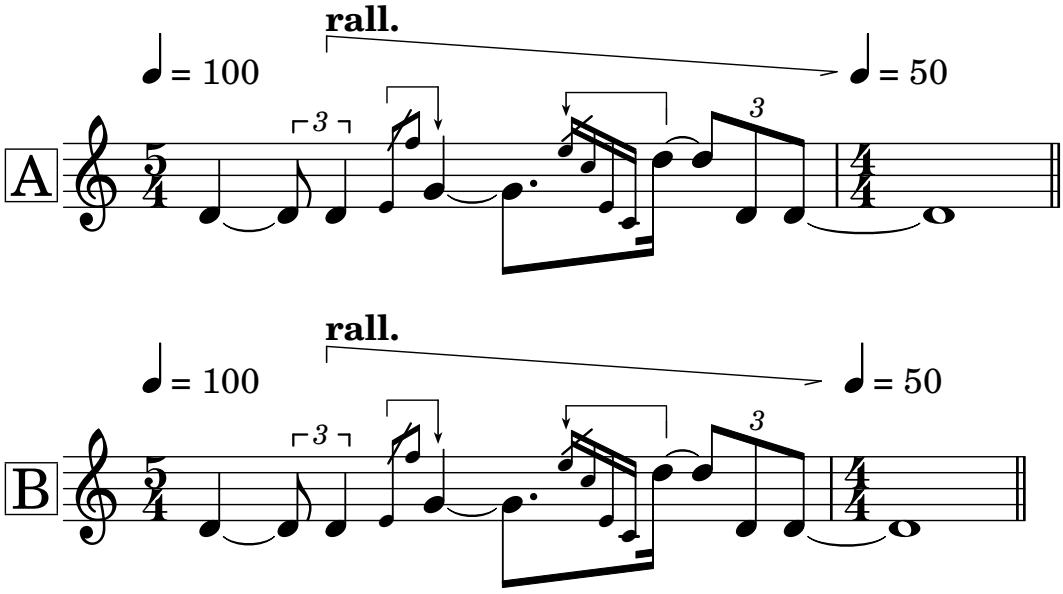
10.1.3 Code

```
1
2  \score {
3    {
4      \override Staff.StaffSymbol.line-positions = #'(6  -6)
5      \strPosClef
6      \dashedLineNotehead g'4
7              ^\markup {\fontsize #-4 \italic flaut.}
8              ^\markup \translate #'(-2.5 . -0) \center-column
9                          {\translate #'(0 . -1.5) \fontsize #-4 II
10                         \fontsize #-4 III}
11                 a' #6
12     \modularLineNotehead a'
13             ^\markup \column {\translate #'(0 . -1.5)
14                         \fontsize #-4 \italic apply \fontsize #-4
15                         \italic pressure}
16                 d'' #15 #150 #6
17     \override TupletNumber.text = #tuplet-number::calc-fraction-text
18     \stemUp  \tuplet 5/4 {
19       \modularLineNotehead d''8 b' #150 #50 #2.5
20       \modularLineNotehead b' f'' #50 #175 #2.5
21       \modularLineNotehead f'' a' #175 #70 #2.5
22       \modularLineNotehead a' c'' #70 #120 #2.5
23       \modularLineNotehead c'' e' #120 #15 #3.5
24     }
25     |
26     \modularLineNotehead e'4
27             ^\markup {\fontsize #-4 \italic ord.}
28                 e' #15 #15 #12
29     \noteheadless e'
30     \dashedLineNotehead e'
31             ^\markup {\fontsize #-4 \italic (flaut.)}
32                 e' #5
33   }
34
35   \layout {
```

```
36      \context {
37        \Score proportionalNotationDuration = #(ly:make-moment 1/10)
38                        \override SpacingSpanner.uniform-stretching = ##t
39      }
40    }
41  }
42
```

Table of Contents

10.2 Multiple Instances Of Spanners At Once



10.2.1 Description

Invoking two or more Text Spanners (that require `\stopTextSpan` for them to finish their processes) all on one single layer could cause the spanners to behave unexpectedly. This entry is an attempt to avoid such unexpected behaviors by invoking a spanner per layer (A), or per staff line (B).

10.2.2 Variables Used

```
\startSlashedGraceMusic
\stopSlashedGraceMusic
\graceNoteBeforeBeatOn
\graceNoteBeforeBeatOff
\graceNoteAfterBeatOn
\graceNoteAfterBeatOff
\rallArrow
```

10.2.3 Code

```
1
2  %%%%%%%%%%%%%%%%%% A %%%%%%%%%%%%%%%%%%
3  \score {
4    \new Staff = "allElementsCombined"
5    \with {instrumentName = \markup {\fontsize #4 \box "A"}}  {
6      \numericTimeSignature
7      \override Score.MetronomeMark.Y-offset = #5.75
8      \tempo 4 = 100
9      \time 5/4
10     <<
11       {
12         \tieNeutral \stemNeutral d'4~
13         \tuplet 3/2 {d'8 d'4}
14         \stemUp \grace  {
15           \startSlashedGraceMusic \graceNoteBeforeBeatOn e'8 f''
16           \stopSlashedGraceMusic
17         } \graceNoteBeforeBeatOff g'4~
18         \stemNeutral g'8.[ \grace {
19         \startSlashedGraceMusic \graceNoteAfterBeatOn
20           e''16 c'' e' c' \stopSlashedGraceMusic
21         }
22         \graceNoteAfterBeatOff d''16]~
23         \tuplet 3/2 {d''8 d'8 d'8~} |
24         \time 4/4
25         d'1 \bar"||"
26       }
27       \\
28       {
29         s4  \tuplet 3/2 {
30           s8 \override Voice.TextSpanner.Y-offset = #6.5
31           s4^\markup {\translate #'(0 . 6.5) \bold "rall."}
32           \rallArrow #4
33         } s2. \tempo 4 = 50 s4*4 \stopTextSpan
34       }
35     >>
36   }
37 }
38
39
```

```
40
41
42
43  %%%%%%%%%%%%%%%%%% B %%%%%%%%%%%%%%%%%%
44  \score {
45    <<
46      \new Staff = "tempoLine" \with {
47        \remove Clef_engraver
48        \remove Staff_symbol_engraver
49        \remove Time_signature_engraver
50      }
51      {
52        \numericTimeSignature
53        \override Score.MetronomeMark.Y-offset = #6
54        \tempo 4 = 100
55        \time 5/4
56        s4  \tuplet 3/2 {
57          s8 \override Voice.TextSpanner.Y-offset = #-2.25
58          s4^\markup {\translate #'(0 . 0) \bold "rall."}
59          \rallArrow #4} s2 \after 64*15 \stopTextSpan s8*2 |
60        \tempo 4 = 50  s4*4
61      }
62      \new Staff = "music"
63      \with { instrumentName = \markup {\fontsize #4 \box "B"}}
64      {
65        \tieNeutral \stemNeutral d'4~
66        \tuplet 3/2 {d'8 d'4}
67        \grace {
68          \startSlashedGraceMusic \graceNoteBeforeBeatOn e'8 f''
69          \stopSlashedGraceMusic
70        } \graceNoteBeforeBeatOff g'4~
71        g'8.[ \grace { \startSlashedGraceMusic \graceNoteAfterBeatOn
72          e''16 c'' e' c' \stopSlashedGraceMusic
73        }
74        \graceNoteAfterBeatOff d''16]~
75        \tuplet 3/2 {d''8 d'8 d'8~} |
76        \time 4/4
77        d'1 \bar"||"
78      }
79    >>
80  }
```

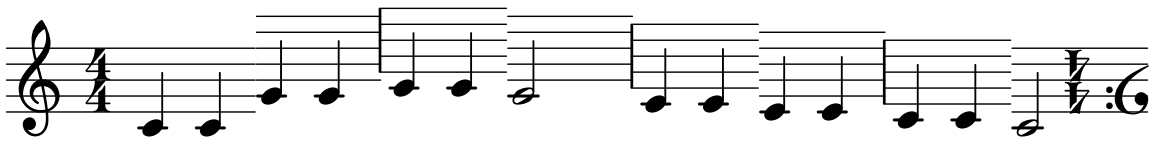
Table of Contents

Chapter 11

Miscellanies

This chapter presents snippets that do not really belong to any of the other preceding chapters but I learned tremendously from making. Quite often I have made these snippets as a diversion.

11.1 Shifting Staffs, Rotated Clef and Time Signature



11.1.1 Description

Staff lines that are shifted so that, when the note moves away from the middle C, the staff lines move accordingly. The excerpt ends with a time signature and a clef that are rotated 180 degrees.

11.1.2 Code

```
\version "2.24.4"
\language "english"

staone = {
```

```
  \stopStaff
  \override Staff.StaffSymbol.line-positions =
  #'(0 2 4 6 8)
  \startStaff
}
statwo = {
  \stopStaff
  \override Staff.StaffSymbol.line-positions =
  #'(1 3 5 7 9)
  \startStaff
}
stathree = {
  \stopStaff
  \override Staff.StaffSymbol.line-positions =
  #'(-1 1 3 5 7)
  \startStaff
}
stafour = {
  \stopStaff
  \override Staff.StaffSymbol.line-positions =
  #'(-2 0 2 4 6)
  \startStaff
}
stafive = {
  \stopStaff
  \override Staff.StaffSymbol.line-positions =
  #'(-3 -1 1 3 5)
  \startStaff
}
stanorm = {
  \stopStaff
  \revert Staff.StaffSymbol.line-positions
  \startStaff
}
{
  \numericTimeSignature
  \time 4/4

  c'4 c' \staone g' g' \statwo a' a' \staone g'2
  \stathree f'4 f' \stafour e' e' \stafive d' d' \stanorm
  \override TextScript.outside-staff-priority = ##f
```

```
46    \once \override TextScript.extra-offset = #'(0 . -4.5)
47    c'2 ^\markup \concat {
48      {
49        \hspace #3 \rotate #180
50        {\compound-meter #'(4 4)}
51      }
52      {
53        \translate-scaled #'(1 . 0.5)
54        \rotate #180 \musicglyph "clefs.F"
55      }
56    }
57    \bar ""
58
59  }
60
61  \layout {
62    \context{
63      \Score   proportionalNotationDuration = #(ly:make-moment 1/7)
64    }
65  }
```

Table of Contents

Chapter 12

Exploring Scheme

12.1 Introduction

Scheme, one of the dialects of the Lisp family of programming languages, is used in LilyPond as its extension language. Scheme allows LilyPond users to explore the inner workings of the program, enabling significant customization. The snippets in this document would not exist without taking advantage of it.[1]

However, learning Scheme can be daunting. In his unfinished book on Scheme and LilyPond, Urs Liska refers to its "thorny path."[2] While I have experience with Common Lisp (another Lisp dialect) from my work with OpenMusic, adjusting to Scheme's grammatical nuances still took some time.

This chapter does not aim to be a comprehensive guide to using Scheme in LilyPond.[3] Instead, it offers suggestions for newcomers to familiarize themselves with Scheme.

12.1.1 Step 1a: Focus on the Scheme Language Itself

Scheme is a language distinct from LilyPond, and understanding this distinction is essential. For simpler LilyPond tasks, Scheme may not be necessary. However, when working with internal parameters, Scheme allows deeper customization. It is beneficial to first study Scheme independently, learning its syntax and concepts by writing simple code.

1. For newcomers: parts of LilyPond code written in Scheme are often enclosed in `#(` and `)`. Numerical values preceded by `#`, and number pairs such as `\#'(1 . -2)`, are also part of the Scheme language.

2. Urs Liska, *Understanding Scheme In LilyPond*, Web Page, December 19, 2020, https://scheme-book.readthedocs.io/en/latest/.

3. For a deeper dive, refer to the resource by Liska, as well as Jean Abou Samra, *Extending LilyPond*, Web Page, December 19, 2021, https://extending-lilypond.gitlab.io/en/index.html. LilyPond also provides its own Extending Manual: https://lilypond.org/doc/v2.24/Documentation/extending/index

12.1.2 Step 1b: Get Used to Prefix Notation

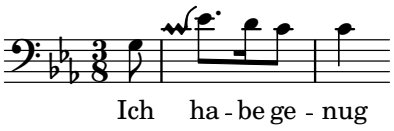
Scheme, like its Lisp relatives, uses prefix notation (Cambridge Polish Notation). Here are examples:

```
(+ 12 34)
>> This expression results in the value of 46.

(+ 4 (* 3 9))
>> This expression first resolves the multiplication: (+ 4 27), which is 31.
```

If you are new to this, I recommend starting with Daniel P. Friedman and Matthias Felleisen, *The little Schemer (4th ed.)* (Cambridge, MA, USA: MIT Press, 1996), ISBN: 0262560992. While you might be eager to dive into using Scheme in LilyPond, learning Scheme as a programming language will make the process smoother.[4]

12.1.3 Step 2: Study Lots of Snippets

Once familiar with Scheme, study how it integrates with LilyPond by reviewing snippets from LSR. Start with shorter examples and analyze their structure. Here is an example snippet for adding the *Schleifer* ornament:[5]



The corresponding code:[6]

```
1  % Implementation by Martin Straeten of the Schleifer ornament
2  % as used by Johann Sebastian Bach, contributed to the user
3  % mailing list. In this case, it functions like a set of (always?)
4  % two grace notes, hence using a modified grace note to represent
5  % it in LilyPond makes sense.
6  %
7  % Code styling and user interface by Simon Albrecht 2024.
8
9  schleiferMarkup = \markup {
10   \large \halign #.2 \raise #0.0
```

4. Liska and Samra's resources serve as excellent refreshers later on.

5. https://lsr.di.unimi.it/LSR/Item?id=1185

6. The mailing list thread referenced in the preamble is available at https://lists.gnu.org/archive/html/lilypond-user/2021-09/msg00352.html

```
11    \combine
12    \halign #.8 \musicglyph "scripts.prall"
13    \rotate #140 \normalsize \raise #2.4 \musicglyph "flags.u3"
14  }
15  schleiferGrace =
16  #(define-music-function (note) (ly:music?)
17     #{
18       \grace {
19         \once\override NoteHead.stencil = #ly:text-interface::print
20         \once\override NoteHead.X-extent = #'(-2 . -0)
21         \once\override NoteHead.text = \schleiferMarkup
22         \once\omit Stem
23         \once\omit Flag
24         $note
25       }
26     #})
27
28  \relative {
29    \time 3/8
30    \partial 8
31    \clef bass
32    \key c \minor
33    g8
34    \schleiferGrace c es8. d16 c8
35    c4
36  }
37  \addlyrics {
38    Ich ha -- be ge -- nug
39  }
```

The `\schleiferGrace` variable creates a customized ornament using Scheme's `define-music-function` macro. For a deeper understanding of the macro syntax, refer to the *LilyPond – Internals Reference*.[7]

Taking the variable `\schleiferGrace`, we see that invoking it creates an instance of activating a Scheme function that starts at Line 16. `define-music-function` is a macro that allows you to create a function that operates on LilyPond. According to Chapter 4: Scheme Functions in *LilyPond – Internals Reference*,[8], the syntax for `define-music-function` is:

7. https://lilypond.org/doc/v2.24/Documentation/internals/scheme-functions
8. https://lilypond.org/doc/v2.24/Documentation/internals/scheme-functions#index-define_002dmusic_002dfunction

```
(define-music-function (arg1 arg2 …)
                       (type1? type2? …)
    function-body)
```

In the code, the argument's name is `note`, and it is tested according to the type specified in `type1?`, which in this case is `ly:music?`. According to the *Internal Reference*, `ly:music?` is a function that checks whether the object—in this case, `note`—is a `Music` object. Thus, it becomes clear that this function will not work unless it is followed by a musical note.

From Line 17 to Line 26, we see that a LilyPond code snippet has been inserted, as `#{` and `#}` signify the boundary of the LilyPond code within the Scheme code. This means that as part of invoking the variable `\schleiferGrace`, it passes through this LilyPond fragment, which is responsible for creating a grace note. Here, the notehead of the grace note is replaced with `\schleiferMarkup`, which is defined in Lines 9 to 14 of the code.[9]

Lines 22 and 23 show that the stem and flag are omitted from the grace note, while Line 24's `$note` signifies that the original argument `note` is called upon.[10] In this way, the *Schleifer* ornament is created from a note that follows the variable `\schleiferGrace`. This note is transformed into a grace note with a customized stencil setting, all done within the Scheme code.

12.1.4 Step 3: Hack the Codes

Once you study a code and become familiar with how it operates, experimenting with the code by hacking is a good way to deepen your understanding. Below, I give one example using the preceding *Schleifer* ornament snippet.

The *LilyPond – Internal Reference* reveals that the object `NoteHead` has its own standard settings, as well as support for about a dozen other interfaces.[11] One of them is the `grob-interface`, which makes it possible to change the color of a graphical object, or *Grob*.[12] Further reading in the *LilyPond – Notation Reference* shows that it is possible to override the color of an object.[13] Let us now tweak the *Schleifer* ornament code to allow us to change the ornament's color.

Following the reference, add the following line underneath `\once\override NoteHead.X-extent`:

9. The technique of sequential overrides, invoking the Scheme command `#ly:text-interface::print`, sets the `.stencil` of the notehead to use whatever is defined in the `.text` parameter. This technique is frequently used and is very useful in customizing notation. See also: https://lilypond.org/doc/v2.24/Documentation/notation/modifying-stencils.

10. Refer to this page for the difference between `#` and `$`: https://lilypond.org/doc/v2.24/Documentation/extending/lilypond-scheme-syntax

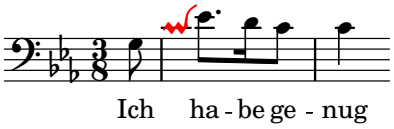
11. https://lilypond.org/doc/v2.24/Documentation/internals/notehead

12. https://lilypond.org/doc/v2.24/Documentation/internals/grob_002dinterface

13. https://lilypond.org/doc/v2.24/Documentation/notation/inside-the-staff#coloring-objects

```
\once\override NoteHead.color = #red
```

Running LilyPond now should produce the following result:



Hard-coding a change like this may be good for testing the waters, but we may want the *Schleifer* ornament in more than just one color. The beauty of extending LilyPond is that we can customize the Scheme code to allow for this flexibility.

Let us move on. We should now let the `define-music-function` know that we are adding an additional argument to specify the color. The first part of the code will look like this:

```
#(define-music-function (note schleiferColor) (ly:music? color?)
```

This adds the argument `schleiferColor`, which only accepts color, as indicated by the corresponding test function `color?`.

Then, implement this argument in the sequence of `\once\override` processes. The line `NoteHead.color` can now be changed to:

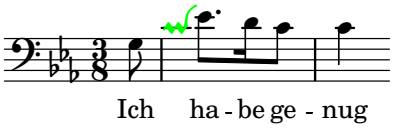
```
\once\override NoteHead.color = #schleiferColor
```

Now, the variable `\schleiferGrace` requires one more argument to specify the ornament's color. The entire code should look like this:

```
schleiferMarkup = \markup {
  \large \halign #.2 \raise #0.0
  \combine
  \halign #.8 \musicglyph "scripts.prall"
  \rotate #140 \normalsize \raise #2.4 \musicglyph "flags.u3"
}

schleiferGrace =
#(define-music-function (note schleiferColor) (ly:music? color?)
   #{
     \grace {
       \once\override NoteHead.stencil = #ly:text-interface::print
       \once\override NoteHead.X-extent = #'(-2 . 0)
       \once\override NoteHead.color = #schleiferColor
       \once\override NoteHead.text = \schleiferMarkup
```

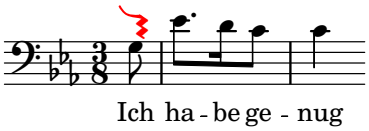
```
16          \once\omit Stem
17          \once\omit Flag
18          $note
19        }
20      #})
21  \relative {
22    \time 3/8
23    \partial 8
24    \clef bass
25    \key c \minor
26    g8
27    \schleiferGrace c #green es8. d16 c8
28    c4
29  }
30  \addlyrics {
31    Ich ha -- be ge -- nug
32  }
```

This produces the following output:



Notice that on Line 27, `#green` has been added. You can change this to any of the colors listed under "Normal Colors" in the *Notation Reference*,[14] such as `#'"lightsalmon"`, `#(x11-color "medium turquoise")`, or even `#'"#5e45ad"`.

As an exercise, try replicating the following excerpt:[15]



NB – In subsequent version updates of this document, I will add examples of Scheme code-heavy snippets.

Table of Contents

14. https://lilypond.org/doc/v2.24/Documentation/notation/list-of-colors
15. See LSR1185e3.ly for the answer.

Bibliography

Boulez, Pierre. *... explosante-fixe ... transitoire VII : (version 1991/93)*. Universal Edition, 1991.

———. *Sur incises : pour trois pianos, trois harpes et trois percussions-claviers (1996/1998)*. Universal Edition, 1998.

Friedman, Daniel P., and Matthias Felleisen. *The little Schemer (4th ed.)* Cambridge, MA, USA: MIT Press, 1996. ISBN: 0262560992.

Levine, Carin, and Christina Mitropoulos-Bott. *The techniques of flute playing = Die Spieltechnik der Flöte*. Kassel ; New York: Bärenreiter, 2003.

Liska, Urs. *Understanding Scheme In LilyPond*. Web Page, December 19, 2020. https://scheme-book.readthedocs.io/en/latest/.

Salzedo, Carlos. *L'étude moderne de la harpe... Modern study of the harp*. 3 p.l., 53 p. New York - Boston, G. Schirmer, 1921.

Samra, Jean Abou. *Extending LilyPond*. Web Page, December 19, 2021. https://extending-lilypond.gitlab.io/en/index.html.

Sparnaay, Harry. *The Bass Clarinet: A Personal History*. Periferia Sheet Music, 2012.

Takemitsu, Tōru. *Fantasma/cantos : for clarinet and orchestra*. Schott ; Schott Japan, 1993.

———. *Les yeux clos II : for piano*. Schott ; Schott Japan, 1990.

Appendices

Appendix A: Resources

As I taught LilyPond in a special topic course at the University of Delaware in Fall 2024, I compiled a list of links to useful websites and pages. It is in no way intended as a comprehensive list; instead, I list some essential pages that I have frequently looked up and found very useful. This page is subject to frequent revision.

On LilyPond

- Website: https://lilypond.org/
- Installing: https://lilypond.org/doc/v2.24/Documentation/learning/installing
- Manuals: https://lilypond.org/manuals.html

Text Editor for LilyPond

- Frescobaldi (Editor): https://frescobaldi.org/

Coding LilyPond

- Cheat Sheet: https://lilypond.org/doc/v2.24/Documentation/notation/cheat-sheet
- Snippets: https://lilypond.org/doc/v2.24/Documentation/web/snippets
- LilyPond Snippet Repository: https://lsr.di.unimi.it/

Mailing List

- Mailing list: https://lists.gnu.org/mailman/listinfo/lilypond-user
- Archives 1 https://lists.gnu.org/archive/html/lilypond-user/
- Archives 2 https://www.mail-archive.com/lilypond-user@gnu.org/

Advanced Topic on LilyPond

- LilyPond – Extending v2.24.4: https://lilypond.org/doc/v2.24/Documentation/extending/index#top

- Scheme (in LilyPond): https://scheme-book.readthedocs.io/en/latest/
- Extending LilyPond: https://extending-lilypond.gitlab.io/en/extending/index.html
- Scheme Resources https://www.gnu.org/software/guile/learn/#scheme-resources
- PostScript Manual: https://www.adobe.com/jp/print/postscript/pdfs/PLRM.pdf
- PostScript Tutorial: https://paulbourke.net/dataformats/postscript/

Troubleshooting

- The default text font for LilyPond doesn't seem to work (Mac)
- Frescobaldi freezes upon loading

Miscellaneous Items

- About Emmentaler font: https://lilypond.org/doc/v2.25/Documentation/notation/the-emmentaler-font

Table of Contents