Original LilyPond Snippets and Their Grammars

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

Version: December 5, 2024

Cite: Y. Onishi, "Original LilyPond Snippets and Their Grammars," (Version December 5, 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

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 Page 1 of this document.

If you have found any of the codes useful, please feel free to use them in your LilyPond files. Crediting me would be greatly appreciated especially if you use it verbatim or in its slightly modified version:

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

In the interest of making my codes available to as many people as possible, I have avoided using copyrighted musical examples. However, wherever appropriate, I have provided bibliographical 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 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.

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 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.

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.

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."[3] 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 have become increasingly disillusioned as I witness 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.[4] On 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. 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

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

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 addresses a specific element of music notation within the Western classical music tradition, such as noteheads, stems, beams, clefs, staff lines, and so on. A later chapter provides additional discussion and context.

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 email exchanges on LilyPond's 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.

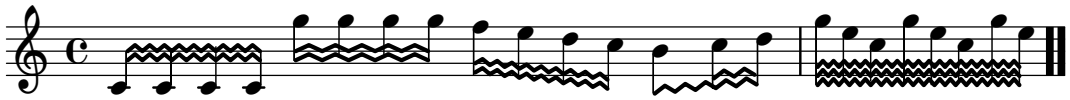
I also want to thank my friends who inspired me to start using LilyPond. It was Cole Ingraham who first introduced me to the program in 2016. My initial attempt at using it was not successful, but more recently, Santiago Beis composed and typeset his orchestral piece *Spletna* entirely in LilyPond, which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

Chapter 1

Beams

1.1 Wiggle Beam (zig-zag shaped beam)



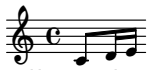
1.1.1 Description

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.

1.1.2 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.

1.1.3 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 ".."
}
```

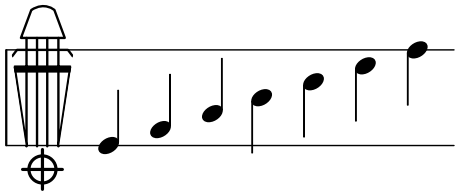
1.1.4 Discussion

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.

Chapter 2

Clefs

2.1 String Position Clef



2.1.1 Description

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

2.1.2 Grammar

```
\strPosClef
```

2.1.3 Code

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

```
newpath

-0.55 7.5 moveto
0 -3 rlineto
1 -6.5 rlineto
-1 -1 rlineto
0 -3 rlineto
4.1 0 rlineto
0 3 rlineto
-1 1 rlineto
1 6.5 rlineto
0 3 rlineto
closepath

} def

fingboardpath clip
newpath
0.15 setlinewidth
0.5 4.75 moveto
0 -6.8 rlineto
-0.75 5 rlineto
3.5 0 rlineto
-0.75 -5 rlineto
0. 6.8 rlineto
stroke
0.35 setlinewidth
-0.4 2.75 moveto
3.75 0 rlineto
stroke

%inner two line
newpath
0.15 setlinewidth
1.16 4.75 moveto
0. -6.8 rlineto
1.8 4.75 moveto
0. -6.8 rlineto
stroke

%bridge
```

```
newpath
-0.4 3.6 moveto
0.3 0.4 rlineto
3.2 0 rlineto
0.3 -0.4 rlineto
stroke

%tailpiece
0.15 4.75 moveto
1 setlinecap
1 setlinejoin
2.75 0 rlineto
-0.65 1.75 rlineto
-0 -0  -0.6 0.55 -1.45 0  rcurveto
closepath
stroke

newpath
0.2 setlinewidth
1 setlinecap
1.5 -2.25 moveto
0 -2.5 rlineto
0.25 -3.5 moveto
2.5 0 rlineto
stroke
newpath
1.5 -3.5 0.85 0 360 arc
stroke
grestore")
                                        (cons 0 3)
                                        (cons 0 1))

strPosClefSize =
#(lambda (grob)
   (let* ((sPCS (ly:grob-property grob 'font-size 0.0))
          (mult (magstep sPCS)))
     (ly:stencil-scale
      strPosClef
      mult mult)))

strPosClef = {
```

```
  \override Staff.Clef.stencil = \strPosClefDesign
}

normalClef = {
  \revert Staff.Clef.stencil
}

{
  \override Staff.StaffSymbol.line-positions = #'(6  -6)
  \override Staff.LedgerLineSpanner.stencil = ##f
  \override Staff.TimeSignature.stencil = ##f
  \override Staff.BarLine.stencil = ##f
  \strPosClef c'4 e' g' b' d'' f'' a''
}
```

2.1.4 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:

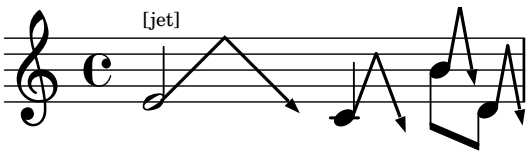
```
%mutesign
newpath
0.2 setlinewidth
1 setlinecap
1.5 -2.25 moveto
0 -2.5 rlineto
0.25 -3.5 moveto
2.5 0 rlineto
stroke
newpath
1.5 -3.5 0.85 0 360 arc
```

3. Once `\strPosClef` is used, in order to revert back to the normal clef, `\normalClef` must be used.
4. See **Combinations** chapter for a possible use of this clef.

Chapter 3

Noteheads

3.1 Jet Whistle (for flute)



3.1.1 Description

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

3.1.2 Grammar

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

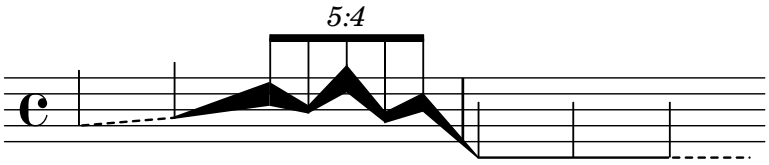
3.1.3 Code

```
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)))
```

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

```
          #{ #pitchthing ^\markup {
            \postscript
            #(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
    }
  }
}
```

3.2 Line as a Notehead



3.2.1 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.

3.2.2 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.

3.2.3 Code

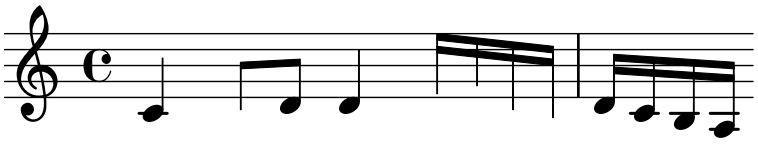
```
% See the entry on "Noteheadless" for its code;
% it is required for the snippet to run properly.

dashedLineNotehead =
#(define-music-function
  (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?)
```

```
(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 Voice.LedgerLineSpanner.transparent = ##t
     \once \override Voice.NoteHead.text = \markup  {
       \postscript
       #(string-append
         "newpath 1 setlinecap 0.1 setlinewidth -0.05 0 moveto 0 "
         (number->string (* beginningThickness 0.005)) " rlineto "
         (number->string x-distance) " "
         (number->string (+ (- (* endingThickness 0.005)
                               (* beginningThickness 0.005) )
                            (* steps 0.5)))
         " rlineto 0 "
         (number->string  (* endingThickness -0.01))  " rlineto "
         (number->string (* -1 x-distance))  " "
         (number->string (- (* endingThickness 0.005)
                            (* beginningThickness 0.005)
                            (* steps 0.5)))
         " rlineto
            closepath
            fill"
         )
     }
     #beginning
     \revert Voice.NoteHead.stencil
     \revert Staff.LedgerLineSpanner.stencil
   }
```

```
   #})
  )


\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
    }
  }
}
```

3.3 Noteheadless



3.3.1 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."

3.3.2 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.

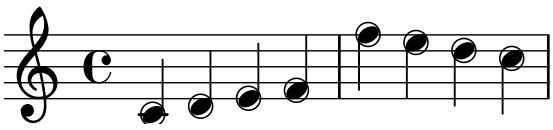
3.3.3 Code

```
%% Inspired by:
%% http://lsr.di.unimi.it/LSR/Item?id=796


noteheadless = {
  \once \override Voice.NoteHead.transparent = ##t
  \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
}
```

3.4 Slap Tongue, Type A



3.4.1 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.

3.4.2 Grammar

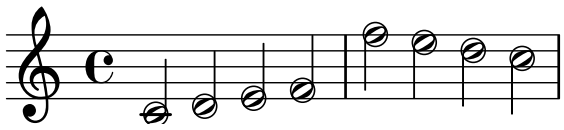
`\slapA NOTE`

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

3.4.3 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''
}
```

3.5 Slap Tongue, Type B



3.5.1 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.

3.5.2 Grammar

`\SlapB NOTE`

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

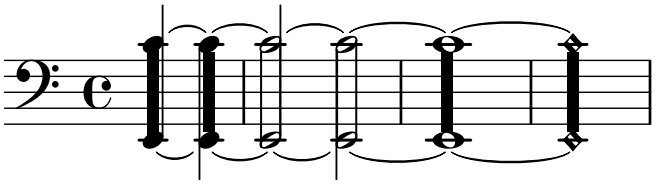
3.5.3 Code

```
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''
}
```

3.5.4 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.

3.6 Tone Cluster



3.6.1 Description

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

3.6.2 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**.

3.6.3 Code

```
toneClusterBar =
#(define-music-function (note1 note2 yOffset yLengthAdjust)
   (ly:music? ly:music? number? number?)
   (let* (
           (note1p (ly:music-property note1 'pitch))
           (note2p (ly:music-property note2 'pitch))
           (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
            0.3 " (number->string (- yOffset 0.5)) " moveto
            0.7 0 rlineto
            0 " (number->string (- (* -0.5 pitchDistance)
                                   (- yLengthAdjust 1))) " rlineto
            -0.7 0 rlineto
            closepath
            fill
            grestore")
       }
     #}
     )
   )
```

```
toneClusterBarHollow =
#(define-music-function (note1 note2 yOffset yLengthAdjust)
   (ly:music? ly:music? number? number?)
   (let* (
           (note1p (ly:music-property note1 'pitch))
           (note2p (ly:music-property note2 'pitch))
           (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
            0.1 " (number->string (- yOffset 0.5)) " moveto
            0 " (number->string (- (* -0.5 pitchDistance)
                                   (+ 0.5 yLengthAdjust))) " rlineto
            0.125 setlinewidth
            1.3 "(number->string (+ 0.75 (- yOffset 0.5))) " moveto
            0 " (number->string (- (* -0.5 pitchDistance)
           (+ 0.75 yLengthAdjust))) " rlineto
            stroke
            grestore")
       }
     #}
     )
   )


toneClusterBarWhole =
#(define-music-function (note1 note2 yOffset yLengthAdjust)
   (ly:music? ly:music? number? number?)
   (let* (
           (note1p (ly:music-property note1 'pitch))
           (note2p (ly:music-property note2 'pitch))
           (note1pnumber (+ (* 7 (ly:pitch-octave note1p))
```

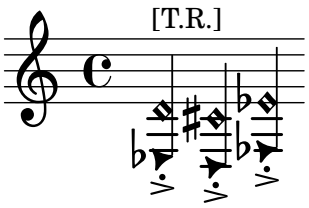
```
                          (ly:pitch-notename note1p)))
          (note2pnumber (+ (* 7 (ly:pitch-octave note2p))
                          (ly:pitch-notename note2p)))
          (pitchDistance (abs (- note1pnumber note2pnumber)))
          )
    #{
      < #note1
      #note2 > ^\markup {
        \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
}
```

3.6.4 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

3.7 Tongue Ram (for flute)



3.7.1 Description

Implementation of the tongue ram notation, as described in *The Techniques of Flute Playing* by Carin Levine and Christina Mitropoulos-Bott.[3]

3.7.2 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.

3.7.3 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.]} }  #}
```

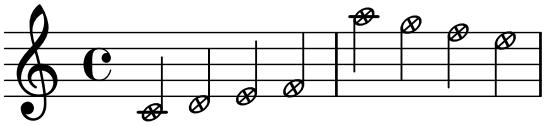
3. Levine and Mitropoulos-Bott, *The techniques of flute playing = Die Spieltechnik der Flöte*, 28.

```
         ))
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-.->}
```

3.7.4 Discussion

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

3.8 X In A Hollow Notehead



3.8.1 Description

While LilyPond Notation Reference provides an example of an X-in-a-circle notehead, its shape differs from the regular notehead.[4] This implementation simulates a hollow notehead with which the X notehead is combined.

3.8.2 Grammar

```
\cirX NOTE
```

3.8.3 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
```

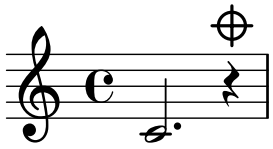
4. https://lilypond.org/doc/v2.24/Documentation/notation/modifying-stencils

```
            \temporary \override NoteHead.stem-attachment =
            #(lambda (grob)
               (let* ((stem (ly:grob-object grob 'stem))
                      (dir (ly:grob-property stem 'direction UP))
                      (is-up (eqv? dir UP)))
                 (cons dir (if is-up 0.2 -0.2))))
            #note
            \revert NoteHead.stencil
            \revert NoteHead.text
            \revert NoteHead.stem-attachment
          #})
{
  \cirX c'4 \cirX d' \cirX e' \cirX f'
  \cirX a''4 \cirX g'' \cirX f'' \cirX  e''
}
```

Chapter 4

Markups

4.1 Mute Sign



4.1.1 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.*[1]

4.1.2 Grammar

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

4.1.3 Code

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

  "newpath
```

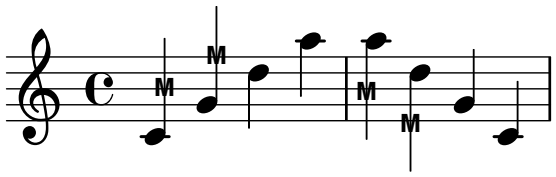
1. Carlos Salzedo, *L'étude moderne de la harpe... Modern study of the harp* (New York - Boston, G. Schirmer, 1921), 19.

```
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 }
```

Chapter 5

Stems

5.1 ”M” on Stem



5.1.1 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.

5.1.2 Grammar

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

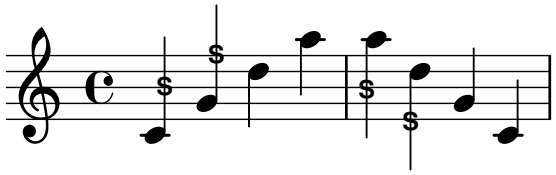
NB `\MOnStemOn` toggles the feature on, while `\MOnStemOff` toggles it off.

5.1.3 Code

```
MOnStemOn = {
  \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 "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'
}
```

5.2 "S" on Stem



5.2.1 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.

5.2.2 Grammar

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

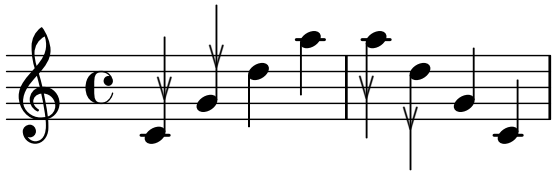
NB `\SOnStemOn` toggles the feature on, while `\SOnStemOff` toggles it off.

5.2.3 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))))
}
```

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

{
  \SOnStemOn c'4 g' \SOnStemOff d'' a''
  \SOnStemOn a'' d'' \SOnStemOff g' c'
}
```

5.3 "V" on Stem



5.3.1 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.

5.3.2 Grammar

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

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

5.3.3 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))))
```

```
}

VOnStemOff = {
  \revert Stem.length
  \revert Stem.stencil
  \revert Flag.stencil
}


{
  \VOnStemOn c'4 g' \VOnStemOff d'' a''
  \VOnStemOn a'' d'' \VOnStemOff g' c'
}
```

Bibliography

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

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

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

Appendices

Appendix A: Resources