

## to kill a monarch

*Preferably played in a dark or dim setting (e.g. with the least light needed by the performers).*

michael winter  
(berlin, de; 2021)

notes . . . . .	1
musical score . . . . .	4
appendix 2 - SuperCollider code and Lilypond templates . . . . .	70

---

### instrumentation and dedication

This piece was originally written for quartertone bass flute and three strings. There are also two electronic accompaniment parts (synthesized by a custom computer program written in SuperCollider).

However, the instrumentation is flexible such that the part labeled in the score by an asterisks (the part originally intended for flute; referred to here as part ‘star’) has a distinct timbre from the other three parts which are labeled by Roman numerals and should be rather homogeneous in timbre within themselves (the parts originally intended for strings, referred to here as the ‘candidate’ parts). These three parts may also be synthesized and played back with the electronic accompaniments (individually referred to as ‘electronic accompaniment’ I and II). Conversely, the part star can be synthesized, while the candidate parts are played on acoustic instruments.

That is, the piece may be played as a solo, trio, or quartet. The reason for this variability stems from a sort of economy-of-means. After moving to Berlin, I started discussing with Rebecca Lane writing a piece for her to play on microtonal-bass flute. At the same time, three mutual friends of ours, all cellists, Deborah Walker, Judith Hammann, and Lucy Railton, were entertaining the idea of starting a cello trio as the former two had also recently moved to Berlin. This flexible instrumentation gives the option for each of them to play the piece as a solo or together in different configurations. There is a certain pragmatism, however, that makes the part star playable on a microtonal bass flute and the candidate parts specifically suited for strings. These idiosyncrasies are given in more detail in the part descriptions below.

I would like to extend a special thanks to Rebecca Lane who compelled the piece. For encouraging me to write it and more specifically, for her suggestion early on to be fastidious about notating all the interrelations among and within the parts. Doing so led to a comprehensive analysis and better understanding of the underlying process which ultimately made the piece possible.

### process and structure (the first paragraph may be used as a short program note)

The first half of the piece is a sort of extended prelude. The second half is a series of modulations / interpolations where each modulation goes from one mode, always a gamut of 7 pitches built upon a given root / fundamental, to another. The interpolation is governed by an algorithm that models the phenomenon where the rich-get-richer such that the more pitches that have been selected from one of a set of potential ‘candidate’ modes, the more likely that candidate will eventually become the destination mode. Once a mode ‘wins’ (becomes overwhelming rich), its wealth is effectively stripped and it can no longer persist.

The current state of the interpolation is articulated by part star and electronic accompaniment I while the candidate parts articulate candidate modes that *may eventually become* the destination mode. That is, at any point in time, the pitch gamut of part star and electronic accompaniment I is comprised of subsets of pitches from the candidate modes articulated. As such, part star and electronic accompaniment I are typically multimodal (built upon 2 or more roots) while, individually, the candidate parts are always unimodal (each built upon one root). The exception being when a destination mode is reached and all parts play the same gamut of pitches built upon a single root.

The score is divided into sections and subsections. The first section is the extended prelude. Starting from section 2, each section is one full modulation of the above-described process where each subsection can be considered as a discrete point in the interpolation. The destination mode is always reached by the ultimate subsection of each section.

## notation

At the beginning of section 1 and each subsection from section 2 onward, a key is provided above the staff that indicates the current root for each mode articulated by each of the candidate parts, respectively; the relationships among the roots; and the relationship of the previous root to the current root within each part. The relationships among the roots are given by frequency ratios written above lines that connect the part numbers (in Roman numerals). The relationship of the previous root to the current root within each part is given by a frequency ratio written below the note of the current root. The note indicating the pitch of the current root is preceded by a note (given in gray) indicating the pitch of the previous root unless the root has not changed.

For the candidate parts, each note indicates the closest pitch in twelve-tone equal temperament with a cent-deviation (100th of a tempered semitone) written above and a frequency ratio from the current root written below. Part star is the same except that the frequency ratio may be written as a superscript of a Roman numeral that indicates which root (of the mode from candidate part I, II, or III) the frequency ratio is referencing. If no Roman numeral is given, the last one is assumed.

All frequency ratios are given in a *collapsed* form as if the pitches were within one octave above the same arbitrary C and always in the form where the numerator is greater than the denominator.

## candidate parts

These parts are designed such that within each subsection, each part only sounds tones with pitches from one mode. A mode is always a 7 pitch gamut in the following form (given by frequency ratios and cents from the 1/1):

		5/4 (386¢)	11/8 (551¢)		13/8 (841¢)	7/4 (969¢)
1/1	9/8 (204¢)			3/2 (702¢)		
		6/5 (316¢)	4/3 (498¢)		8/5 (813¢)	15/8 (1088¢)

Horizontally aligned frequency ratios indicate that one or the other pitch may be used in the mode (most likely the top one). Therefore, only a handful of modes are possible even though the root progresses / changes throughout. Since the mode structure is rather limited, the players need to be able to transpose the possible modes arbitrarily. This is why these parts are specifically suited for strings. Basically the intervals within the modes stay the same or similar, but the position on the instrument changes.

## part star

The pitch gamut of this part often comprises notes derived from several candidate modes at once. As mentioned above, a Roman numeral indicates the part that is articulating the candidate mode from which the note is drawn from. When the gamut is multimodal, the part generally has a lower temporal density and the sequence of pitches is always rising: each pitch is followed by the next highest pitch in the gamut until an upper limit is reached. This should make playing the part more feasible for a wind instrument like bass flute despite the complexity. Also, if the final note in a rising sequence is too high or feels too rushed, the written penultimate note can be extended till the sequence drops back down instead of playing the written ultimate note. Throughout the first section and in each ultimate subsection from section 2 onward, the part comprises arbitrary sequences of notes like the other parts. Therefore, there is a distinct shift starting at section 2, where the rising sequences begin. This should be made as clear as possible. Similarly, the interruption of the rising scale in the ultimate subsection of each section should also be made as clear as possible. If necessary, this part can be transposed up or down an octave. The part is written assuming bass flute hence the octavation marking below the treble clef; i.e. sounding an octave lower.

## electronic accompaniments I and II

Electronic Accompaniment I articulates the current state of the interpolation as mentioned previously. Electronic Accompaniment II articulates the overall section-per-section form by swelling throughout each section on a tone with a pitch that is a perfect 5th above or a perfect 4th below the root of the destination mode which then cadences to the root itself in the ultimate subsection of each section.

## dynamics

Each section should generally have a dynamic crescendo that peaks in the ultimate subsection. This can be executed by following the dynamic envelope of electronic accompaniment I where the candidate parts sound within / equal to the electronics and part star sounds slightly above / in the foreground. Generally, the sound should be rather present; filling the room more and more throughout each crescendo. However, the beginning of the crescendo need not be extremely quiet nor the peak excessively loud. With that said, the cadential peak of electronic accompaniment II in each ultimate subsection should briefly overwhelm all the other parts.

Within each subsection, each part often has a small flourish of activities which should be articulated as sub-swells within the larger dynamic profile with peaks based on the temporal density.

## tempo

The score is written in a cut time with a tempo where the half note equals 60 beats per minute. The piece may be played at a slower tempo, but no less than the half note at 50 beats per minute.

## SuperCollider program

A custom program written in the SuperCollider language can synthesize the parts that can be played by acoustic instruments and the strictly electronic accompaniments. A version of the application source code is appended at the end of this score. However, it is recommended to ensure that the most recent version of the code is being used which can be downloadable from a git repository at: [https://unboundedpress.org/code/mwinter/to\\_kill\\_a\\_monarch](https://unboundedpress.org/code/mwinter/to_kill_a_monarch)

The application provides a transport window to control playback and set variables as well as a basic mixing console to control the levels of the various sonic elements of the piece. The program also allows new versions of the piece to be generated, transcribed, and rendered to separate audio files for use with other playback systems. Note that while most of the code facilitates usability, playback, and transcription, the music of the piece is completely generated by the algorithm in `tkam_musical_data_generator.scd`. A help / readme file is included with the application documenting its functionality and use. To launch the application, execute `tkam_main.scd` in SuperCollider (on Linux, this is achieved by pressing `cmd+enter` with the cursor anywhere within the code block).

The generation of this document (using LaTeX) contains a version date at the bottom of this page in order to help track changes and the git repository will also detail commit changes. The piece was last generated using SuperCollider version 3.13 and Lilypond version 2.24.1.



application user interface

to kill a monarch

seed: 19800725

michael winter  
(berlin, germany; 2021)

**1.1**

III  $\frac{1}{1}$  II  $\frac{1}{1}$  I  $\frac{1}{1}$

+0 +0 +0

$\frac{1}{1}$   $\frac{1}{1}$   $\frac{1}{1}$

$\text{♩} = 60$

I  $\frac{+41}{13/8}$   $\frac{+16}{6/5}$   $\frac{+4}{9/8}$   $\frac{+0}{1/1}$

II

III

\*  $\frac{-31}{III^{7/4}}$   $\frac{+2}{3/2}$   $\frac{-31}{7/4}$   $\frac{+0}{1/1}$

**1.2**

I  $\frac{+2}{3/2}$   $\frac{-31}{7/4}$

II  $\frac{+2}{3/2}$

III  $\frac{+4}{11/8}$   $\frac{-49}{9/8}$   $\frac{+2}{3/2}$

\*  $\frac{+16}{6/5}$   $\frac{-49}{11/8}$   $\frac{-31}{7/4}$   $\frac{+41}{13/8}$   $\frac{+2}{3/2}$   $\frac{+4}{9/8}$

I  $\frac{-49}{11/8}$   $\frac{+0}{1/1}$

II  $\frac{+0}{1/1}$   $\frac{-49}{11/8}$

III  $\frac{-31}{7/4}$   $\frac{+16}{6/5}$   $\frac{+4}{9/8}$   $\frac{+2}{3/2}$   $\frac{+0}{1/1}$

\*  $\frac{+0}{III^{1/1}}$   $\frac{-49}{11/8}$

-1-

**1.3**

I: +4, +2, +0  
II: +0, -31, +4, +41, +2, -31, +41, +16, -49, +16  
III: +16, -31, +41, +0, +2, +4  
\*: -31, +0

9/8, 3/2, 1/1, 1/1, 7/4, 9/8, 13/8, 3/2, 7/4, 13/8, 6/5, 11/8, 6/5, 1/1, 6/5, 7/4, 13/8, 1/1, 11/8, 3/2, 9/8, III 7/4, 1/1

**1.4**

I: +41, +4, -49  
II: +41, +4  
III: -49  
\*: -49

13/8, 9/8, 11/8

**1.5**

I: -49, +16, -31  
II: +2, -49, +0, -31, +0, -49, +4  
III: -31, +41, +4, +0, +16, +0  
\*: +41, +2, -31

11/8, 6/5, 7/4, 3/2, 11/8, 1/1, 7/4, 1/1, 11/8, 9/8, 7/4, 13/8, 9/8, 1/1, 3/2, 6/5, 1/1, III 13/8, 3/2, 7/4

25

Staff I: +41, +2, +0, +4  
13/8, 3/2, 1/1, 9/8

Staff II: +16, +2, -31, +0, +41, -31, +16, -49  
6/5, 3/2, 7/4, 1/1, 13/8, 7/4, 6/5, 11/8  
1/1

Staff III: -49, +0, -31  
11/8, 1/1, 7/4

Staff \*: -49, +41, +16  
III<sup>11/8</sup>, 13/8, 6/5

29

1.6

Staff I: +0, +16, +0, +2, +0, -31, +0  
1/1, 6/5, 1/1, 3/2, 1/1, 7/4, 11/8, 1/1

Staff II: (Empty)

Staff III: +41, +0  
13/8, 1/1

Staff \*: +4, +2  
9/8, III<sup>3/2</sup>

33

1.7

Staff I: +41, +4, +41, -31, +16  
13/8, 9/8, 13/8, 7/4, 6/5

Staff II: +41, +0  
13/8, 1/1

Staff III: +2, +16  
3/2, 6/5

Staff \*: -49, -31, +0  
11/8, 7/4, III<sup>1/1</sup>

37

Score for measures 37-40. The system includes four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 37 has a circled number 37. Measure 38 has a circled number 38. Measure 39 has a circled number 39. Measure 40 has a circled number 40. Fingerings are indicated by numbers +0, +2, -31, +4, -31, +2, +0, -49, +4, +41. Rhythmic values are 3/2, 7/4, 1/1, 9/8, 7/4, 3/2, 1/1, 11/8, 9/8, 13/8.

41

1.8

Score for measures 41-44. The system includes four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 41 has a circled number 41. Measure 42 has a circled number 42. Measure 43 has a circled number 43. Measure 44 has a circled number 44. Fingerings are indicated by -31, +2, -49, +2, -31, +4. Rhythmic values are 7/4, 3/2, 11/8, 3/2, 7/4, 9/8.

45

Score for measures 45-48. The system includes four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 45 has a circled number 45. Measure 46 has a circled number 46. Measure 47 has a circled number 47. Measure 48 has a circled number 48. Fingerings are indicated by -49, +16, +41, +0, +2, -31, +4, +2, -49, +0, +41, +2, +41, +16, +16, -49, +2, +16. Rhythmic values are 11/8, 6/5, 13/8, 1/1, 9/8, 3/2, 7/4, 9/8, 3/2, 13/8, 11/8, 1/1, 13/8, 3/2, 13/8, 6/5, 6/5, 11/8, 3/2, 6/5.

1.9

1.10

1.11



61 1.12

Staff I: +0, -31, 7/4, 1/1

Staff II: +41, -49, +0, +16, 13/8, 11/8, 1/1, 6/5

Staff III: -31, +41, +16, +2, +4, +0, 7/4, 6/5, 3/2, 9/8, 1/1, 13/8

Staff \*: +16, +2, +0, -31, III 6/5, 3/2, 1/1, 7/4

65 1.13

Staff I: -49, +2, +0, +4, 11/8, 3/2, 1/1, 9/8

Staff II: -31, +41, +0, +41, -31, +16, 7/4, 13/8, 1/1, 13/8, 3/2, 7/4, 6/5

Staff III: +16, -31, -49, +41, +4, 6/5, 7/4, 11/8, 13/8, 9/8

Staff \*: -49, +4, +41, III 11/8, 9/8, 13/8

69

Staff I: +16, +41, -31, -49, +2, +16, +0, 6/5, 13/8, 7/4, 11/8, 3/2, 6/5, 1/1

Staff II: +2, +4, +0, +2, 3/2, 9/8, 1/1, 3/2

Staff III: +2, +16, 3/2, 6/5

Staff \*: +0, III 1/1

73 **1.14**

I: +2  
II: +16, 6/5, -31, -49, +41  
III: 6/5, 11/8, 13/8, 7/4, 11/8

77 **1.15**

I: +0  
II: +16 +2 +4, -31 +0, +41 +4 -31 +0, +2, +16  
III: 6/5 3/2 7/4 11/8, 1/1 9/8 1/1, 13/8 9/8 7/4 1/1, 3/2, +2, 3/2

\*: -31, +2, -49, +0, +41, +0, +4, 1/1

81

I: +41 +4, -31, -49  
II: 9/8 13/8, 7/4, 9/8 3/2 1/1 9/8  
III: -49 +41 +0 -31, 13/8 11/8, 1/1 7/4, 9/8 3/2 1/1 9/8

\*: -31, -49, +2, 11/8, 3/2



97 1.18

System 97-100: Four staves (I, II, III, \*) with fret numbers and accidentals. Staff I: Treble clef, 13/8 time signature, +41. Staff II: Bass clef, +2, +0, +4, 1/1, 3/2, 9/8. Staff III: Bass clef, +41, +0, +4, 13/8, 1/1, III<sup>9/8</sup>. Staff \*: Treble clef, +41, +0, +4, 13/8, 1/1, III<sup>9/8</sup>.

101 1.19

System 101-104: Four staves (I, II, III, \*) with fret numbers and accidentals. Staff I: Treble clef, +0, -31, +2, 1/1, 7/4, 3/2. Staff II: Bass clef, +2, -49, +41, -31, +0, 3/2, 11/8, 9/8, 13/8, 7/4, 1/1. Staff III: Bass clef, +2, -49, +41, -31, +0, 3/2, 11/8, 9/8, 13/8, 7/4, 1/1. Staff \*: Treble clef, +16, +4, +0, -31, -49, 6/5, 9/8, 1/1, 7/4, 11/8.

105

System 105-108: Four staves (I, II, III, \*) with fret numbers and accidentals. Staff I: Treble clef, +4, +16, +0, +2, -31, -49, 9/8, 6/5, 1/1, 3/2, 11/8, 7/4. Staff II: Bass clef, -31, +41, 7/4, 13/8. Staff III: Bass clef, +16, -49, +4, 6/5, 11/8, 9/8. Staff \*: Treble clef, +2, III<sup>3/2</sup>.

109 **1.20**

I  
II  
III  
\*

113 **1.21**

I  
II  
III  
\*

117 **1.22**

I  
II  
III  
\*

121

Musical score for measures 121-124. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (bass clef). Measure 121 starts with a circled '121'. Measure 122 has a circled '123'. Measure 123 has a circled '124'. Measure 124 has a circled '125'. Fingerings are indicated by numbers 1-4 above notes. Interval numbers are written below notes. A double bar line is present at the end of measure 124.

I: +2 +0 -49 +16 +0 +2  
3/2 1/1 6/5 1/1 3/2  
11/8

II: +16 -31 +41 +2  
6/5 7/4 13/8 3/2

III: +2 -49 +0 +4 -31 +0 +16  
3/2 11/8 1/1 9/8 13/8 6/5  
7/4 1/1

\*: -31 +16 +41  
III 7/4 6/5 13/8

125

Musical score for measures 125-128. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (bass clef). Measure 125 starts with a circled '125'. Measure 126 has a circled '123'. Measure 127 has a circled '124'. Measure 128 has a circled '125'. Fingerings are indicated by numbers 1-4 above notes. Interval numbers are written below notes. A double bar line is present at the end of measure 128.

I: -31 +4 +0 +41 +16 +16  
9/8 1/1 6/5 6/5  
7/4 13/8

II: +4 +0 +41 -49  
9/8 1/1 13/8 11/8

III: -31 +2  
7/4 III 3/2

129

Musical score for measures 129-132. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (bass clef). Measure 129 starts with a circled '129'. Measure 130 has a circled '124'. Measure 131 has a circled '125'. Measure 132 has a circled '126'. Fingerings are indicated by numbers 1-4 above notes. Interval numbers are written below notes. A double bar line is present at the end of measure 132.

I: -49 +4 +0 +41 +2 -31  
11/8 9/8 1/1 13/8 3/2 7/4  
13/8

II: +0 -31 +4 -49  
1/1 7/4 9/8 11/8

III: +4 +41 +2 -49  
9/8 13/8 3/2 11/8

\*: +16 -31 +4  
6/5 7/4 9/8

133

Musical score for measures 133-140. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 133 is marked with a circled '133'. Above the staves are fret numbers: +16 +41, -49 +4, +0, +0 +2 -31 +0, -31, +41 +16 +0 +2 +4 -49 -31, +2 +0, +41 +0, -49 +16. Below the staves are time signatures: 6/5 13/8, 11/8 9/8, 1/1, 1/1 3/2 7/4 1/1, 7/4, 6/5 1/1 3/2 11/8 7/4 13/8 9/8, III<sup>3/2</sup> 1/1 13/8 1/1 11/8 6/5.

1.25

137

Musical score for measures 137-140. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 137 is marked with a circled '137'. Above the staves are fret numbers: +41 +2, +41, +41 +2 +0, +16. Below the staves are time signatures: 13/8 3/2, 13/8, 13/8 3/2 1/1 6/5, III<sup>7/4</sup>.

141

1.26

Musical score for measures 141-144. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 141 is marked with a circled '141'. Above the staves are fret numbers: +16 -31 +4, +2, +41 +2 -49 +4. Below the staves are time signatures: 6/5 7/4 9/8 1/1 3/2, -31 7/4 9/8, 13/8 3/2 11/8 III<sup>9/8</sup>.

145

Staff I: Treble clef, whole notes, rests.

Staff II: Alto clef, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +4, +16, +2. Rhythmic values: 9/8, 6/5, 3/2.

Staff III: Bass clef, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +41, +0, +2, +4, +16, +0, -49, +2, +41. Rhythmic values: 1/1, 3/2, 9/8, 6/5, 1/1, 11/8, 13/8, 13/8, 3/2.

Staff \*: Soprano clef, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +0, +4, +2, -31. Rhythmic values: 1/1, 9/8, 3/2, 7/4.

1.27

149

Staff I: Treble clef, quarter notes, quarter notes, quarter notes, quarter notes. Accents: -49, +41, +16. Rhythmic values: 11/8, 13/8, 6/5.

Staff II: Alto clef, quarter notes, quarter notes, quarter notes, quarter notes. Accents: -49, +0, +2. Rhythmic values: 11/8, 1/1, 3/2.

Staff III: Bass clef, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +0. Rhythmic values: 1/1.

Staff \*: Soprano clef, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +0, +4, +16. Rhythmic values: III<sup>1/1</sup>, 9/8, 6/5.

153

1.28

Staff I: Treble clef, whole notes, rests.

Staff II: Alto clef, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +16, -31, +2, +41. Rhythmic values: 7/4, 6/5, 3/2, 13/8.

Staff III: Bass clef, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes. Accents: +2, +4, +0, -31. Rhythmic values: 3/2, 9/8, 1/1, 7/4.

Staff \*: Soprano clef, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes, quarter notes. Accents: -49, +41, +0, -31, +0. Rhythmic values: 11/8, III<sup>13/8</sup>, 1/1, 7/4, 1/1.



157

1.29

I

II

III

\*

161

I

II

III

\*

1.30

165

I

II

III

\*

169 1.31

Staff I: -31 +16  
7/4 6/5

Staff II: -31 +41 +4  
7/4 13/8 9/8

Staff III: +2 -31 +0 +41 +16 -49  
3/2 1/1 13/8 7/4 11/8 6/5

Staff \*: -31 -49 +2 +16 +4 +0  
7/4 11/8 3/2 6/5 9/8 1/1

173

Staff I: +0 -31  
1/1 7/4

Staff II: +0 -49 +4 +2 +41 +0 -31 +16 +16 +2  
1/1 11/8 3/2 13/8 1/1 7/4 6/5 3/2 6/5

Staff III: +0 +2 -49 +0 +41 +4 -49 +0 +2 +0  
1/1 3/2 11/8 1/1 13/8 9/8 11/8 1/1 3/2 1/1

Staff \*: -31 -49 +4 +41 +2  
11/8 9/8 13/8 3/2

1.32

177

Staff I: -49 +16 +41 +0 +16  
11/8 6/5 13/8 1/1 6/5

Staff II: +41 +4 +0 +41 -49 +0  
13/8 9/8 1/1 13/8 11/8 1/1

Staff III: -31 +16 +41  
7/4 6/5 13/8

Staff \*: +4 -31 +0 +4 +2 +16 -31 +41 +2 +16  
9/8 7/4 1/1 9/8 3/2 6/5 7/4 13/8 3/2 6/5

1.33

181

II +16 +2 +4 -31 -49 +4 +16  
6/5 3/2 9/8 7/4 11/8 9/8 6/5

III -49 +4 +0  
11/8 9/8 1/1

\* +0 -49 +4 +41 +16  
III 1/1 11/8 9/8 13/8 6/5

1.34

185

I -49 +0 +2 +41 +2 +0 +16  
1/1 3/2 13/8 3/2 11/8 7/4 9/8 1/1

II -31 +4 +0  
7/4 9/8 1/1

III -49  
11/8

\* -49  
III 11/8

1.35

189

I +0 +2 +41 -49 +4 +0 +16 +2  
1/1 3/2 13/8 11/8 9/8 7/4 1/1 6/5 3/2

II +2 +41 -49 +4  
3/2 13/8 11/8 9/8

III -31  
7/4

\* +0 -31 +16 +2  
1/1 7/4 6/5 3/2

193 1.36

Staff I: -31 +16 -49 +0 +4 +2 +16 -49 +0 +41 +2 +0  
 7/4 6/5 11/8 1/1 9/8 3/2 6/5 11/8 1/1 13/8 3/2 1/1

Staff II: +41 +2 -31 +0 +16  
 13/8 3/2 7/4 1/1 6/5

Staff III: +41 +2 +16 -49 -31 +4 +2 +0 +16 +4 +16 +41 +2  
 3/2 6/5 11/8 7/4 9/8 3/2 1/1 6/5 9/8 6/5 13/8 3/2

Staff \*: +4 +16 +0  
 III<sup>9/8</sup> 6/5 1/1

197

Staff I: -49 -31 +4 +0 +41 +2 +16  
 7/4 9/8 11/8 1/1 13/8 3/2 6/5

Staff II: +2 -49  
 3/2 11/8

Staff III: -31 +16 +0 +4  
 7/4 6/5 1/1 9/8

Staff \*: +2  
 III<sup>3/2</sup>

201 1.37

Staff I: -31 +0 +16 +4 -49  
 7/4 6/5 1/1 9/8 11/8

Staff II: -31 +2 +0 +4  
 7/4 3/2 1/1 9/8

Staff III: -49 +0  
 11/8 1/1

Staff \*: +0  
 III<sup>1/1</sup>

205 1.38

Staff I: +2 +4 +0 -31 +0  
3/2 9/8 1/1 7/4 1/1

Staff II: -31 +16 +2 +4 -31 -49 +2  
7/4 6/5 13/8 3/2 7/4 9/8 11/8 3/2

Staff III: -31 +2 -49 +0 +2 +16  
7/4 3/2 11/8 1/1 3/2 6/5

Staff \*: +41 +2 -49 +4 +0  
13/8 3/2 11/8 9/8 1/1

209

Staff I: +16 +41 -49 +0 +2 +4  
6/5 13/8 11/8 3/2 1/1 9/8

Staff II: +0 -49 +16  
1/1 11/8 6/5

Staff III: +4 +41 +0 -49 +4 +16  
13/8 1/1 9/8 11/8 9/8 6/5

Staff \*: +16 -31  
III<sup>6/5</sup> 7/4

213 1.39

Staff I: (Empty)

Staff II: (Empty)

Staff III: +2  
3/2

Staff \*: +2 +0 +2  
3/2 1/1 III<sup>3/2</sup>

217 1.40

Musical score for measures 217-220. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 217 has a +2 fretting instruction above the first staff. Measure 218 has fretting instructions +2, +4, -31, +0, +2, +4 above the second staff. Measure 219 has fretting instructions +4, -31, +0, -49 above the third staff. Measure 220 has a +16 fretting instruction above the fourth staff. Rhythmic values are indicated below the notes: 3/2, 9/8, 7/4, 1/1, 3/2, 9/8, 9/8, 7/4, 1/1, 11/8, 6/5.

221

Musical score for measures 221-224. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 221 has fretting instructions -49, +0, -31, +41, +16, -49 above the first staff. Measure 222 has fretting instructions +41, +0, +16 above the second staff. Measure 223 has fretting instructions +41, +0, +2 above the third staff. Measure 224 has a -49 fretting instruction above the fourth staff. Rhythmic values are indicated below the notes: 11/8, 1/1, 7/4, 13/8, 6/5, 11/8, 13/8, 1/1, 6/5, 13/8, 1/1, 3/2, III<sup>11/8</sup>.

225 1.41

Musical score for measures 225-228. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 225 has a +16 fretting instruction above the first staff. Measure 226 has a -49 fretting instruction above the second staff. Measure 227 has fretting instructions +41, -31, +0, +4 above the third staff. Measure 228 has a +2 fretting instruction above the fourth staff. Rhythmic values are indicated below the notes: 6/5, 11/8, 7/4, 1/1, 9/8, 13/8, 3/2.

229 1.42

Staff I: -31, +0, +41, -31  
7/4, 1/1, 9/8, 13/8, 7/4

Staff II: +41, +16, +0  
13/8, 6/5, 1/1

Staff III: -31, +2  
7/4, 3/2

Staff \*: +41, -31, +4, +0, -31, +4, +0, +4, -49, +2  
III 7/4, 9/8, 1/1, 7/4, 9/8, 1/1, 9/8, 11/8, 3/2

233 1.43

Staff I: +2, +16  
3/2, 6/5

Staff II: +16, -31  
6/5, 7/4

Staff III: +16, -49  
6/5, 11/8

Staff \*: +0, +41, +16, +41, +2  
III 1/1, 13/8, 6/5, 13/8, 3/2

237

Staff I: +0, +41, +4  
1/1, 13/8, 9/8

Staff II: +2, +4  
3/2, 9/8

Staff III: +2, +0, +41, -31, +16, -49, +0, +4, +16  
3/2, 1/1, 13/8, 7/4, 6/5, 11/8, 1/1, 9/8, 6/5

Staff \*: +0, -31, -49, +0  
III 1/1, 7/4, 11/8, 1/1

2.1

III 9/8 II 3/2 I

+0 +4 +0 +0 +2

241

7/4 11/8 3/2 1/1 9/8 3/2 9/8 1/1

-29 -47 +4 +2 +6 +4 +2

+41 13/8

-27 -45 +44

7/4 11/8 13/8

+0 +4 +8

II 1/1 I 3/2 III 9/8

2.2

III 9/8 II 3/2 I

+4 +0 +2 +2 +0

245

+18 +42 +4

6/5 13/8 3/2

1/1 3/2 3/2

+41 -31 +16

7/4 6/5 13/8

+4 +8 +6 -27

1/1 9/8 3/2 7/4

-49 +41

II 11/8 I 13/8

2.3

III 3/2 II 9/8 I

+4 +2 +2 +0 +0 +4

249

+0 +2 -49 -31 +4 +0

1/1 3/2 11/8 7/4 1/1 9/8

+6

3/2

+2

1/1

+6

3/2

+6 +0 +4 +8 -49

II 9/8 I 1/1 II 3/2 III 9/8 I 11/8



**2.4**

III  $\frac{3}{2}$  II  $\frac{3}{2}$  I

+2 +0 +4 +2

1/1 1/1 4/3

253

I  $\frac{13}{8}$   $\frac{9}{8}$

II  $\frac{11}{8}$  1/1  $\frac{7}{4}$  1/1  $\frac{9}{8}$   $\frac{11}{8}$   $\frac{3}{2}$

III  $\frac{6}{5}$   $\frac{9}{8}$   $\frac{13}{8}$

\* III  $\frac{9}{8}$

+44 +8

-49 +0 -31 +0 +4 -49 +2

+18 +6 +42

+6

257

I  $\frac{11}{8}$   $\frac{3}{2}$

II  $\frac{6}{5}$   $\frac{3}{2}$

III  $\frac{7}{4}$   $\frac{3}{2}$  1/1

\* III  $\frac{6}{5}$   $\frac{11}{8}$

-47 +4

+16 +2

-29 +4 +2

+18 -47

**2.5**

III  $\frac{3}{2}$  II  $\frac{12}{11}$  I

+2 +0 +0 +2 +2 -49

$\frac{3}{2}$   $\frac{3}{2}$   $\frac{12}{11}$

261

I  $\frac{1}{1}$   $\frac{6}{5}$   $\frac{9}{8}$   $\frac{1}{1}$

II

III

\*  $\frac{11}{8}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{11}{8}$

-49 -33 -45 -49

-47 +4 -29 +6

2.6

III 1/1 II 1/1 I  
+0 -49 +2 -49 -49  
11/8 12/11 1/1

265

I -47 -8 -33 -49 -45  
3/2 13/8 6/5 1/1 9/8

II -29 +42 +18  
7/4 13/8 6/5

III -49 +2  
11/8 3/2

\* +18 +3  
6/5 11/8

269

I -33 +20 -49 -47 -49 +3 -47  
7/4 6/5 1/1 3/2 11/8 1/1 11/8 3/2

II -49 -45 -33 +3 -49 +3 -49 -47  
1/1 9/8 6/5 11/8 1/1 11/8 1/1 3/2  
13/8

III -49 -33 -45 -49 -47 +20 -33 -47 +20 -49 +3  
1/1 6/5 9/8 1/1 3/2 7/4 6/5 3/2 7/4 1/1 11/8

\* -45 -49  
III 9/8 1/1

273

I -33 -49  
6/5 1/1

II -33 +20 +3 -8 -45 -49  
6/5 7/4 11/8 13/8 9/8 1/1

III +20 -45  
7/4 9/8

\* -33 -8 -47 +3 -8 -45 +20 -47  
6/5 13/8 3/2 11/8 13/8 9/8 7/4 3/2

**3.1**  $\frac{4}{3}$

III  $\frac{16}{13}$  II  $\frac{13}{12}$  I  
-49 -49 -8 -49 -47

277

I  $\frac{1}{1}$   $\frac{16}{13}$   $\frac{4}{3}$  +22 -47 -43 -45 -47 +5  
 $\frac{7}{4}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{1}{1}$   $\frac{11}{8}$

II +43  
 $\frac{11}{8}$

III -33 -47  
 $\frac{6}{5}$   $\frac{3}{2}$

\* 7

**3.2**  $\frac{1}{1}$

III  $\frac{13}{12}$  II  $\frac{13}{12}$  I  
-49 -47 -8 -47

281

I -6  
 $\frac{13}{8}$

II -8  
 $\frac{1}{1}$

III -8 +20  
 $\frac{13}{8}$   $\frac{7}{4}$

\* -8 +20 +40 +5 7  
III  $\frac{13}{8}$   $\frac{7}{4}$  I  $\frac{5}{4}$   $\frac{11}{8}$

285

I -47 +40 +5 -47 +22 -45  
 $\frac{1}{1}$   $\frac{5}{4}$   $\frac{11}{8}$   $\frac{1}{1}$   $\frac{7}{4}$   $\frac{3}{2}$

II -6 -8 -39  
 $\frac{3}{2}$   $\frac{1}{1}$   $\frac{7}{4}$

III -47 +22  
 $\frac{1}{1}$   $\frac{7}{4}$

\* -6 +32 +40 +40 +5 -45  
III  $\frac{13}{8}$  II  $\frac{13}{8}$  III  $\frac{5}{4}$   $\frac{5}{4}$   $\frac{11}{8}$   $\frac{3}{2}$

-24-

**3.3**

13/12

III 1/1 II 13/12 I

-47 -8 -47 -47 -8

1/1 13/12 13/12

289

I

11/8

II

-47 -6 -43 +5 -47 -43 -45 +22

1/1 9/8 11/8 1/1 9/8 3/2 7/4

13/8

III

+5

11/8

\*

-6 +22 +32 -4

III 13/8 7/4 I 13/8 9/8

**3.4**

1/1

III 13/12 II 13/12 I

-47 -47 -8 -8 -47

1/1 13/12 13/12

293

I

-4 +32 -6 -22 -8

9/8 13/8 3/2 5/4 1/1

9/8

II

III

-6 +40 -43 -45

13/8 5/4 9/8 3/2

\*

+5 -45 -6

III 11/8 3/2 III 13/8

297

I

-45

3/2

II

-8 -22 +32 -39 -8

1/1 13/8 5/4 7/4 1/1

III

+22 -47

7/4 1/1

\*

+22 +32 +40 +40

7/4 II 13/8 III 5/4 5/4

**3.5**

III 13/12 1/1 II 13/12 I  
-47 -8 -47 -47 -8

301

I  
+32 -22 -39 +43 +32 -22  
13/8 3/2 9/8 1/1 11/8 13/8 5/4  
5/4 7/4

II  
+40 -45 -45  
3/2 5/4 3/2

III  
+22 -47  
7/4 1/1

\*  
-22 +5 -45  
I<sup>5/4</sup> III<sup>11/8</sup>  
3/2

**3.6**

III 13/12 1/1 II 1/1 I  
-47 -8 -47 -8 -47

305

I

II

III  
+40  
5/4

\*  
-6 +32 +40 -22  
13/8 I<sup>13/8</sup> III<sup>5/4</sup> I<sup>5/4</sup>

309

I  
-47 -6 -45 -47 +5 -47  
1/1 13/8 9/8 1/1 5/4 1/1  
3/2 11/8

II  
-43 -47 -6 +22 -45  
11/8 3/2 11/8 13/8 7/4 3/2  
9/8 1/1

III  
+32 -6  
13/8 3/2

\*  
+5 -45  
II<sup>11/8</sup> 3/2

**3.7**

III 1/1 II 1/1 I

-8 -47 -47 -47

13/12 1/1 1/1

313

314

315

316

+40 -47 -43 -6 +22 -43 +5 -45

1/1 9/8 13/8 7/4 11/8 3/2

5/4

-43

III 9/8

317

+22 -43 -6 +40 -47 -45 +22

7/4 9/8 13/8 5/4 1/1 3/2 7/4

-47 +40 +22

1/1 5/4 7/4

-43 +22 +40 -47 -43 -6 +5

9/8 7/4 5/4 1/1 9/8 13/8 11/8

+5 +22 -47 +40 -6 -47 -6 -45 +22 -6 -47

11/8 7/4 1/1 5/4 13/8 1/1 13/8 7/4 13/8 1/1

**4.1**

III 5/4 II 5/4 I

-47 -47 +40 -47

1/1 5/4 1/1

321

+40 -47 +40

5/4 1/1 5/4

+22

7/4

+5

11/8

4.2

	5/4	
III	13/10	II
-47	+40	+40
		-6
		I
		-47
	5/4	13/10
		1/1

925

Musical score for system 925, measures 925-928. It features four staves: I (treble), II (alto), III (bass), and \* (piano). Measure 925 has a +5 adjustment and a 11/8 time signature. Measure 926 has +40 and -20 adjustments and 1/1 and 13/8 time signatures. Measure 927 has -6 and +5 adjustments and 13/8 and 11/8 time signatures. Measure 928 has +8, -6, +22, and +8 adjustments and 13/8, 7/4, and 7/4 time signatures. A key signature change to one flat is indicated between measures 927 and 928.

929

Musical score for system 929, measures 929-932. Measure 929 has +22, -47, and -43 adjustments and 7/4, 1/1, and 9/8 time signatures. Measure 930 has -4 and +9 adjustments and 3/2 and 6/5 time signatures. Measure 931 has +45 adjustment and 11/8 time signature. Measure 932 has +42, +26, +40, -20, +43, +42, and +8 adjustments and 5/4, 3/2, 1/1, 13/8, 9/8, 7/4, 3/2, and 7/4 time signatures.

4.3

	13/10	
III	5/4	II
+40	-6	-47
		-47
		-6
		I
	1/1	13/8
		13/8

933

Musical score for system 933, measures 933-936. Measure 933 has a -4 adjustment and a 3/2 time signature. Measure 934 has -43 and -47 adjustments and 9/8 and 1/1 time signatures. Measure 935 has +8 adjustment and 7/4 time signature. A key signature change to one flat is indicated between measures 934 and 935.

337

341

345



349

5.1

353

III<sup>6/5</sup>

357

**5.2**

III 8/7 II 16/11 I  
-6 -37 +45 -6 -37 +45

361 8/7 16/11 14/11

I +47 +31 +49  
3/2 5/4 9/8

II -2 -37 -4  
9/8 7/4 3/2

III +31 -37 +3 +14 -33 -22 +31 -37  
7/4 3/2 1/1 13/8 11/8 9/8 6/5 7/4 1/1

\* -4 -2  
I<sup>11/8</sup> I<sup>9/8</sup>

**5.3**

III 8/7 II 1/1 I  
-37 -6 -6 -37 +45 -37

365 8/7 8/7 14/11

I +45  
1/1

II -37 -33  
1/1 9/8

III -2  
9/8

\* -35 +3 +31  
III<sup>3/2</sup> 13/8 II<sup>7/4</sup>

369 -22 -35 +14 -35 -37 +31 +3 +14  
6/5 7/4 3/2 1/1 7/4 11/8  
3/2 11/8 13/8

II

III -37 -4 -37 -2 -4  
7/4 3/2 7/4 9/8 3/2

\* -33 -2  
9/8 III<sup>9/8</sup>

**5.4**

III 1/1 II 1/1 I  
-6 -37 -37 -37

8/7 1/1 1/1

I

II -22 +14 +31 -35 +3 -37 -22 -33 +14  
6/5 11/8 7/4 3/2 13/8 1/1 6/5 9/8 11/8

III +3 -37 -33 -22 -37  
13/8 1/1 9/8 6/5 1/1

\* -35 -37 +31 +31 -33  
III<sup>3</sup>/2 1/1 7/4 7/4 13/8 9/8

377

I -33 +31 -37 +14 -22 -33 +3 -37 -22 -35  
9/8 7/4 1/1 6/5 9/8 13/8 1/1 3/2 6/5

II -35 +3  
3/2 13/8

III +31  
7/4

\* -37 +3 -22 -35 +14 -33 +31 +3  
1/1 13/8 6/5 11/8 9/8 7/4 13/8

**6.1**

III 4/3 II 1/1 I  
-37 -35 -37 -37

4/3 1/1 1/1

381

I -37 +14  
1/1 11/8

II -22  
6/5

III +16  
11/8

\* -37

6.2

III 8/5 II 4/3 I  
-35 -22 -37 -35 -37  
8/5 4/3 1/1

385  
I -22 -35 +14 -37 -33 -35  
6/5 3/2 11/8 1/1 9/8 3/2

II -35 -37 -33 +14 +3 -37 -35  
3/2 1/1 9/8 11/8 13/8 1/1 3/2

III -49 -33 -35  
5/4 3/2 1/1

\* +14 +14  
II 11/8 11/8

389

I -33 -22  
9/8 6/5

II +33  
7/4

III +47 -35 -20  
7/4 5/4 3/2

\* +3 -32 +47 -33 -22 +14  
I 13/8 II 9/8 III 7/4 II 3/2 I 6/5 11/8

6.3

III 4/3 II 6/5 I  
-22 -35 -35 -37 -37 -22  
8/5 4/3 6/5

393  
I -22  
1/1

II -22 -37 -35 +14 +3 -22 +31  
1/1 3/2 11/8 13/8 6/5 7/4  
6/5

III -22 +30 -32 +5  
1/1 11/8 9/8 13/8

\* +3 -32  
13/8 III 9/8

6.4

III 6/5 II 8/5 I

-35 -37 -37 -22 -22 -35

4/3 6/5 8/5

397

+47 -22

7/4 1/1

-35 -22 -30 +47

1/1 11/8 5/4 7/4

-33 +14 -33

3/2 1/1 9/8 9/8 11/8

+47 -33

II 7/4 3/2

6.5

III 1/1 II 4/3 I

-37 -35 -22 -35 -35 -37

4/3 8/5 4/3

401

+5 +16 -35 -32 +5

13/8 11/8 1/1 9/8 13/8

+47 -35 -18

5/4 7/4 9/8

3/2 -35

-22

6/5 13/8 1/1

+5 +14 +14 +30 -49

13/8 III 11/8 11/8 II 11/8 III 5/4

6.6

III 16/11 II 4/3 I

-35 +14 -35 -37 -37 -35

12/11 4/3 4/3

405

-37 -33 +14 +31 -35

1/1 9/8 11/8 7/4 3/2 13/8

-22 +14 -35 -37

6/5 11/8 3/2 1/1

+16 -33 +5 -49

11/8 3/2 13/8 5/4

-33 -32 +33 -49 -35 +5 -35

3/2 9/8 7/4 5/4 1/1 5/4 13/8 1/1

+16 -33

11/8 3/2

6.7

409

+5 +16 +33 -49 +33 -35 +16

13/8 11/8 5/4 7/4 5/4 7/4 1/1 11/8

+14

11/8

-17

7/4

+16 +14

3/2 1/1

+33 -17 +14 +0 -49

III 7/4 I 7/4 III 5/4 I 5/4

III 12/11 II 1/1 I 1/1

+14 -35 -37 +14 -35 +14

12/11 16/11 12/11

413

+16 +0 +18 -46 +16

5/4 9/8 13/8 3/2 1/1

+0

-46 +14 +18

13/8 5/4 1/1 9/8

-32 +33 +16 +5 +33 -32

9/8 11/8 13/8 7/4 9/8

-49 +16 -32

5/4 9/8 11/8

-35 -46 -17 +33 +18

II 11/8 13/8 7/4 III 7/4 II 9/8

9/8 5/4

III 1/1 II 1/1 I 1/1

-35 +14 +14 +14

12/11 1/1 1/1

6.8

417

-17

7/4

-35

13/8

+16 +14 -17 +0 -35

1/1 3/2 7/4 11/8 5/4

-35 +16 -17 -35 +14 +18 -46

11/8 III 3/2 7/4 11/8 1/1 9/8 13/8

III 1/1 II 1/1 I 1/1

-35 +14 +14 +14

12/11 1/1 1/1

421

Staff I: -35, +0 -46, -17, +18, +16, -17 +14  
 11/8, 5/4 13/8, 7/4, 9/8, 3/2, 7/4 1/1

Staff II: +16 +18, +14 -35 +0, +14, +18  
 1/1 11/8 5/4, 1/1, 9/8

Staff III: +14 +18 -17, +16, +14, -35  
 1/1 9/8 7/4 3/2, 1/1, 11/8

Staff \*: -35  
 11/8

7.1

III 3/2 II 3/2 I  
 +14 +14 +16 +14

Staff I: 1/1, 3/2, 1/1, -35, +16  
 11/8, 3/2

Staff II: -33, +16 -44 -15, +2  
 11/8, 1/1 13/8 5/4, 7/4

Staff III: -46, +16, +0 +18 +0  
 13/8, 3/2 5/4 9/8 5/4, 1/1

Staff \*: -46, -17  
 III 13/8, 7/4

7.2

III 13/8 II 13/12 I  
 +14 +16 -46 +14 +16

Staff I: 1/1, 13/12, 3/2, +2, 5/4

Staff II: 1/1, 13/12, 3/2, +2, 5/4

Staff III: -17, 7/4

Staff \*: -33, +6, II 11/8, II 11/8

433

III  $\frac{5}{4}$  11/8

7.3

III  $\frac{3}{2}$  II  $\frac{13}{8}$  I

437

III  $\frac{5}{4}$

7.4

III  $\frac{12}{11}$  II  $\frac{11}{8}$  I

441



445

11/8

-35

+6 -35 +17 -35

-35 -31 -31

-48 -33

1/1 9/8 9/8 5/4 3/2

13/8 11/8 1/1

+18 +2

3/2 5/4

-33

III 11/8

Detailed description: This system contains measures 445-448. It features four staves: I (treble), II (alto), III (bass), and \* (piano). Measure 445 has a treble staff with a whole note and a piano staff with a whole note. Measure 446 has a treble staff with a whole note and a piano staff with a whole note. Measure 447 has a treble staff with a whole note and a piano staff with a whole note. Measure 448 has a treble staff with a whole note and a piano staff with a whole note. Rhythmic values are indicated below the staves. Fingerings and accidentals are shown above the notes.

449

7.5

12/11 1/1 12/11

III 1/1 II 12/11 I

+16 -35 -35 +14 +16

-17

7/4

+16 +2 -33

5/4 1/1 11/8

+6

13/8

-35 +34 -31

1/1 7/4 9/8

-33

11/8

Detailed description: This system contains measures 449-452. It features four staves: I (treble), II (alto), III (bass), and \* (piano). Measure 449 has a treble staff with a whole note and a piano staff with a whole note. Measure 450 has a treble staff with a whole note and a piano staff with a whole note. Measure 451 has a treble staff with a whole note and a piano staff with a whole note. Measure 452 has a treble staff with a whole note and a piano staff with a whole note. Rhythmic values are indicated below the staves. Fingerings and accidentals are shown above the notes.

453

7.6

1/1 1/1 1/1

III 1/1 II 1/1 I

-35 -35 +16 -35

1/1 1/1 12/11

+18 +16

3/2 1/1

-33 -31 -48 +34 -31 -31

3/2 9/8 1/1 3/2 9/8

5/4 7/4

-44 -15 -31 -48

13/8 7/4 III 9/8 5/4

Detailed description: This system contains measures 453-456. It features four staves: I (treble), II (alto), III (bass), and \* (piano). Measure 453 has a treble staff with a whole note and a piano staff with a whole note. Measure 454 has a treble staff with a whole note and a piano staff with a whole note. Measure 455 has a treble staff with a whole note and a piano staff with a whole note. Measure 456 has a treble staff with a whole note and a piano staff with a whole note. Rhythmic values are indicated below the staves. Fingerings and accidentals are shown above the notes.

457

I: +34, -48, -33, +6, -31, -35, -48  
II: +17, -48, -33, +34, -35, -33  
III: +17, +34, -48  
\*: -31, -33, -35, +17, -35, -33, +6

Rhythmic values: 7/4, 5/4, 3/2, 13/8, 9/8, 1/1, 5/4, 11/8, 5/4, 3/2, 7/4, 1/1, 3/2, 11/8, 7/4, 5/4, 3/2, 1/1, 11/8, 1/1, 7/4, 3/2, 13/8

8.1

III 11/8 II 16/13 I  
-35 +17 -35 -35 +6  
11/8 1/1 16/13

461

I: +17  
II: 11/8  
III: -35, +6, -35, -31, +17  
\*: -35, -31, -48, -33

Rhythmic values: 1/1, 13/8, 1/1, 9/8, 11/8, 7/4, 1/1, 9/8, 5/4, 3/2

465

I: +8, +10, -25, +6, +46  
II: +6, +17, -31, +34, -33, -35, -48, +6, -33  
III: +17  
\*: +46, +17, -33

Rhythmic values: 3/2, 9/8, 7/4, 1/1, 13/8, 13/8, 11/8, 7/4, 9/8, 3/2, 1/1, 5/4, 13/8, 3/2, 7/4, 1/1, 13/8, II 11/8, 3/2

8.2

11/8

III 22/13 II 16/13 I

+17 -35 +6 +6 -35

469

I +8 1/1 16/13 16/13 +17 -31

3/2 11/8 7/4 9/8

II -33 -31 -31

9/8 9/8 3/2

III

\* +34

7/4

473

I -33

3/2

II +6 -43 +6 +21 +8

1/1 11/8 1/1 6/5 3/2

III +19 -43 +32 +19

3/2 13/8 6/5 3/2

\* +19 -43

III<sup>3/2</sup> II<sup>11/8</sup>

8.3

16/13

III 1/1 II 16/13 I

+17 +6 +6 -35

477

I +6 -33 +17

13/8 3/2 11/8

II +46 +6 -25

13/8 1/1 7/4

III +6

7/4 1/1

\* +46 -25

III<sup>13/8</sup> 7/4

8.4

III  $\frac{1}{1}$  II  $\frac{1}{1}$  I  
 +6 +6 -35 +6  
 1/1 1/1 16/13

481

I -35 -33 1/1 3/2 6/5 7/4 13/8 +21 -25 +46

II

III -43 11/8 9/8 7/4 11/8 +6 -25 -43 1/1

\*  $\frac{1}{3/2}$  III  $\frac{1}{1}$  +6

485

I -43 11/8 +10 9/8

II +21 6/5 -43 11/8 +6 +10 1/1 9/8 +8 +6 -25 3/2 7/4 1/1

III +21 6/5 +10 9/8 +8 3/2 +21 6/5 +6 1/1 -25 7/4 +8 3/2

\* +21 6/5 -25 7/4 +8 3/2 +21 6/5

9.1

III  $\frac{3}{2}$  II  $\frac{3}{2}$  I  
 +6 +6 +8 +6  
 1/1 3/2 1/1

489

I +8 3/2 +6 1/1

II +21 6/5 -43 11/8 -25 7/4 +6 1/1 +10 9/8 +21 6/5 +46 13/8

III

\* +46 13/8 +10 9/8

493

9.2

497

9.3

501

9.4

III 1/1 II 1/1 I  
+10 +8 +10 +6 +10  
1/1 4/3 9/8

505

-25 +46 -43

7/4 13/8 11/8

-23 +23 +12 -41 +8 +12

6/5 7/4 9/8 11/8 1/1 9/8

-4 +10 -21

5/4 1/1 7/4

-39 +46

11/8 13/8

509

+10 -4 -21 +12 +10 +23 -21 +14 +10 -39

1/1 5/4 7/4 1/1 8/5 7/4 3/2 1/1 9/8 1/1 11/8

-39

11/8 1/1 3/2 9/8

+10 +12 +14

-39 +12 +10 +23 +10 +12 -21 +14 -4 -39

11/8 3/2 1/1 8/5 1/1 3/2 7/4 9/8 5/4 11/8

-4 -21 +12 +14 -39 +10 -21 -4 +10 +23 -21 +14

III 5/4 7/4 3/2 9/8 11/8 1/1 7/4 5/4 1/1 8/5 7/4 9/8 3/2

10.1

III 3/2 II 12/11 I  
+10 +10 +12 +10 -39  
1/1 3/2 11/8

513

-4

5/4 11/8

+10 +23

1/1 8/5

+10

1/1

-4 +10 +12 -39 -35

5/4 1/1 3/2 11/8 9/8

**10.2** 12/11

III 3/2 II 11/8 I 1/1

+10 +12 +12 +10 -39

3/2 3/2 1/1

517

-37 -35 -39 -37 +30 -39

3/2 9/8 1/1 7/4 1/1

3/2

+14 -2 -48 +14

5/4 13/8 3/2

+14 +23

9/8 8/5

+23 -21 +23 -2

III 8/5 8/5 II 5/4 7/4

521

+2 +12 +47 -35 -39 +30 +47 -39 -35 -37 +12

13/8 11/8 9/8 1/1 7/4 5/4 1/1 9/8 3/2 11/8

13/8 11/8 9/8 1/1 5/4 7/4 5/4 1/1 9/8 3/2 11/8

+14 -21 +23

9/8 7/4 1/1 11/8 8/5

+10 -39

-20 +12 -2

7/4 1/1 5/4

+12 +14 +30

I 11/8 II 9/8 I 7/4

**10.3** 11/8

III 1/1 II 11/8 I

+12 -39 +10 -39 -39 +10

12/11 11/8 11/8

525

12/11 11/8 11/8 +23 +10 +14

8/5 1/1 9/8

-37 +47

3/2 5/4

-35 +12 -37

9/8 11/8 3/2

-35 +23 +47

III 9/8 I 8/5 III 5/4

10.4

III 55/32 1/1 II 55/32 I  
 -39 +23 -39 +23 +10 -39  
 1/1 55/32 11/8

529

533

10.5

III 55/32 1/1 II 55/32 I  
 -39 +23 +23 -39  
 55/32 1/1 1/1

537



10.6

541

545

11.1

549

11.2

III 13/8 II 13/8 I  
 -36 +23 -36 +23  
 1/1 13/8 1/1

553

I: +10, -25, 11/8, 5/4, +23, +25, 1/1, 3/2

II: +10, -8, 5/4, 7/4, -36, +4, 1/1, 13/8

III: +33, 7/4

\*: +25, -36, 3/2, 13/8

557

I: +10, +27, -36, -8, +23, +25, -25, 5/4, 9/8, 13/8, 7/4, 1/1, 3/2, 11/8, 3/2

II: -50, -36, -32, -36, +4, 1/1, 5/4, 9/8, 1/1, 13/8

III: -32, -36, 9/8, 1/1

\*: -32, -50, III 9/8, 5/4

11.3

III 13/8 II 5/4 I  
 -36 -36 +23 +23 +10  
 1/1 13/8 5/4

561

I: -50, 13/8

II: +25, +10, -8, -25, +23, +27, +10, 3/2, 5/4, 7/4, 11/8, 5/4, 9/8, 5/4, 1/1, 3/2

III: -34, -32, -50, -34, +33, +4, -36, 3/2, 9/8, 5/4, 3/2, 7/4, 13/8, 1/1

\*: -50, +27, +10, +4, III 5/4, II 9/8, 5/4, III 13/8

11.4

13/10

III 5/4 II 13/8 I

-36 +10 +23 +10 -36

13/10 1/1 13/10

565

I

II

III

\*

11/8 9/8 7/4

II 3/2 I 11/8

11.5

13/10

III 13/10 II 1/1 I

+10 -36 +23 +10 -36 +10

13/10 5/4 13/10

569

I

II

III

\*

3/2 5/4 1/1 9/8 3/2

3/2 7/4 1/1 11/8 9/8 5/4 7/4 1/1

9/8 5/4 III 7/4

573

I

II

III

\*

11/8 1/1

13/8

3/2 13/8 7/4 5/4

II 9/8 5/4 11/8 III 9/8 5/4

**11.6** 1/1

III 13/10 II 13/10 I  
-36 +10 +10 -36 +10

677 13/10 13/10 1/1

I  
II  
III  
\*  
-50  
II<sup>5/4</sup>

-36 -50 -32  
1/1 5/4 9/8  
+12 +10 -4  
1/1 5/4 3/2

**11.7** 1/1

III 1/1 II 1/1 I  
+10 -36 +10 +10

581 1/1 13/10 1/1

I  
II  
III  
\*  
-50  
13/8  
III<sup>7/4</sup> 9/8

+14 -4 +12 -4  
9/8 5/4 1/1 3/2 5/4  
+10 +12 -4  
1/1 13/10 1/1

**11.8** 1/1

III 1/1 II 1/1 I  
+10 -36 +10 +10

585 1/1 13/10 1/1

I  
II  
III  
\*  
-50  
13/8 7/4 11/8 5/4 7/4 13/8 9/8 11/8 3/2 5/4  
III<sup>11/8</sup> 1/1 5/4 7/4 11/8 9/8 13/8 5/4 1/1 13/8 7/4  
-49-  
+14 +10 +12 -4 -21 -39 +14 -50  
7/4 1/1 11/8 3/2 1/1 5/4 11/8 9/8 13/8  
+10 -39 -21 +12 +10 +12  
1/1 11/8 7/4 3/2 1/1 3/2 9/8  
+12 +10 -4 -21 -39 +14 -50 -4 +10 -50 -21  
3/2 1/1 5/4 7/4 11/8 9/8 13/8 5/4 1/1 13/8 7/4

589

I

II

III

\*

13/8

1/1

-50

+12 -39 -4 +14

3/2 11/8 5/4 9/8

12.1

III 9/8 II 16/13 I

+10 +14 +10 +10 -50

593 9/8 1/1 16/13

I

II

III

\*

9/8

1/1

16/13

7/4

7/4

5/4 7/4 1/1 3/2

+19

-4 -21 +10 +12

+14

1/1

-4

II 5/4

12.2

III 18/13 II 16/13 I

+14 +10 -50 -50 +10

597 1/1 16/13 16/13

I

II

III

\*

1/1

16/13

16/13

9/8

1/1

13/8

3/2

+18

9/8

+18 -9 +12

III 9/8 I 13/8 II 3/2

601

I +10 1/1 +14 9/8

II +19 7/4 +2 11/8 7/4

III +0 5/4 +16 3/2

\* -21 7/4 +2 11/8 -4 5/4 +18 13/8 -9

12.3

III 18/13 II 18/13 I

+14 -50 -50 +14 +10 -50

18/13 18/13 16/13

605

I +10 1/1 +12 3/2 +19 7/4 +2 11/8

II -46 13/8

III +2 11/8

\* +19 7/4 -46 11/8

12.4

III 1/1 II 1/1 I

-50 +14 -50 -50

1/1 18/13 1/1

609

I +37 5/4 -9 13/8 -46 9/8 -50 1/1 -48 3/2 -46 9/8 +19 7/4 -48 3/2 +19 7/4 -46 9/8

II +18 1/1 +14 5/4 +0 9/8

III +19 7/4 -9 13/8 -50 1/1 -48 3/2

\* -46 9/8 +37 5/4 +2 11/8

613

Staff I: -50, +37, -48, +19, -9  
1/1, 5/4, 3/2, 7/4, 13/8

Staff II: -50, +19, -50, -48, +37  
1/1, 7/4, 1/1, 3/2, 5/4

Staff III: -46, +19, -48, +37, -50, +2, +19, -50, -9, -48, -50  
9/8, 7/4, 3/2, 5/4, 1/1, 11/8, 7/4, 1/1, 13/8, 3/2, 1/1

Staff \*: -50, +37, -48, -46, -50, -9, +37, +19, -46  
III<sup>1/1</sup>, 5/4, 3/2, 9/8, 1/1, 5/4, 11/8, 13/8, 7/4, 9/8

13.1

5/4, 5/4, 1/1, 1/1

-50, +37, -50, -50

617

Staff I: +37  
5/4

Staff II: +2, +19  
11/8, 7/4

Staff III: +19, +2, +38, +40  
7/4, 11/8, 3/2, 9/8

Staff \*: -48, -50, -46, +2, -48, -50, +37, -48  
3/2, 1/1, 9/8, 11/8, 3/2, 1/1, 5/4, II<sup>3/2</sup>

13.2

10/9, 5/4, 9/8, 1/1

+37, -50, -50, -46

621

Staff I: -48, -50, +37, -9, -50  
3/2, 1/1, 5/4, 13/8, 1/1

Staff II: +5, +37, -23  
7/4, 1/1, 13/8

Staff \*: -9, +38, -46, +37, 7  
13/8, III<sup>3/2</sup>, II<sup>9/8</sup>, 5/4

**13.3**

III 9/8 II 5/4 I  
+37 -46 -50 -46 +37  
10/9 1/1 10/9

625

I: -44 -46 -46 -42 +23 -46 +40 -44  
3/2 1/1 3/2 1/1 9/8 7/4 1/1 5/4 3/2

II: +37 -48 -46 -48 -50 +2  
5/4 3/2 9/8 3/2 1/1 11/8

III: -48 +37 +38  
6/5 1/1 3/2

\*: +2 -48 -44 +38  
II 11/8 3/2 I 3/2 III 3/2

629

I: -12 +38 -23 -48 +37 +37 +5 +38 -23 -12 -48 +37  
11/8 3/2 13/8 6/5 1/1 1/1 7/4 3/2 13/8 11/8 6/5 1/1

II: (empty staff)

III: +40 -46 -44  
5/4 1/1 3/2

\*: -23  
I 13/8

**13.4**

III 10/9 II 10/9 I  
-46 -50 +37 +37 -46  
1/1 5/4 10/9

633

I: +5  
7/4

II: +37 -48 -46 +5  
5/4 3/2 9/8 7/4

III: +23 -44  
7/4 3/2

\*: -23 -46 +37  
13/8 II 9/8 5/4



13.5

III 10/9 II 10/9 I  
 -46 +37 -46  
 1/1 1/1 1/1

637

9/8 3/2 9/8 1/1  
 -44 -42 -46  
 -23 +37 +38 -48  
 13/8 1/1 6/5 3/2  
 -42 -46 -5 +40  
 9/8 1/1 13/8 5/4  
 +40 +5 -44 +38 -23  
 II<sup>9/8</sup> III<sup>11/8</sup> 3/2 II<sup>3/2</sup> 13/8

13.6

III 1/1 II 1/1 I  
 -46 +37 -46 -46  
 1/1 10/9 1/1

641

13/8 5/4 9/8 7/4  
 -5 +40 -42 +23  
 +37  
 1/1  
 +5 -44 -46 +23 -5 +5 -46 +5 -44  
 11/8 3/2 1/1 7/4 13/8 11/8 13/8 3/2  
 1/1 11/8  
 -42 +40 +5 -44 -5 -23  
 III<sup>9/8</sup> III<sup>11/8</sup> 3/2 13/8 II<sup>13/8</sup>  
 II<sup>9/8</sup>

645

3/2 1/1 1/1 13/8 7/4 11/8 5/4  
 -44 -46 -46 -5 +23 +5 +40  
 11/8 3/2  
 +40 +23 -46 +23 -46 +5 -44  
 5/4 7/4 9/8 9/8 13/8 9/8 1/1 5/4 7/4  
 1/1 7/4 13/8 11/8 3/2  
 -44 -5 +40 -42 +5 +5 -44 +23  
 III<sup>3/2</sup> 13/8 7/4 1/1 11/8 11/8 3/2 7/4  
 5/4 9/8

649

9/8

-42

-5

+5 -46

+23 -42

-44 -5 -44 +40

13/8 11/8 1/1 7/4 3/2 13/8 1/1 3/2 5/4

-46 -5 +5

1/1 13/8 11/8

-46 +40 -5

1/1 5/4 13/8

14.1

1/1

III 3/2 II 3/2 I

-46 -46 -44 -46

1/1 3/2 1/1

653

-46

1/1

+7 -3 -42 -44 +42 +25 -40 -42

11/8 3/2 1/1 5/4 7/4 9/8 13/8 3/2

-42

9/8

-44 +23

3/2 III 7/4

14.2

9/8

III 4/3 II 3/2 I

-46 -42 -44 -46

9/8 1/1 1/1

657

-44 +5 +40

3/2 11/8 5/4

+7 +25 +42

11/8 7/4 5/4

-44

3/2

+42 -42

II 5/4 III 9/8

**14.3**

III 4/3 II 1/1 I  
-42 -44 -46 -44  
1/1 1/1 3/2

661

I -46 -5 +5  
1/1 13/8 11/8 3/2

II -44  
1/1

III +44 -38 +9 -1 -42 -40  
5/4 9/8 11/8 13/8 1/1 3/2

\* +25 +44  
II 7/4 III 5/4

665

I -44 +42 -40 -44 -42  
1/1 5/4 9/8 1/1 3/2

II +7 -40  
11/8 9/8

III +27  
7/4

\* -40 +42 +7 -42  
II 9/8 5/4 11/8 3/2

**14.4**

III 1/1 II 1/1 I  
-42 -44 -44 -44  
4/3 1/1 1/1

669

I -3 +25 +7 +42 -44 -40 -44 -3 +42  
13/8 7/4 11/8 5/4 1/1 9/8 1/1 13/8 5/4

II -42  
3/2

III +9  
11/8

\* +25 -3  
7/4 III 13/8

-56-

673

Staff I: Treble clef, notes with accidentals +25 and -44. Rhythmic values: 7/4, 1/1.

Staff II: Bass clef, notes with accidentals -44, -3, +42, -42, -44. Rhythmic values: 1/1, 13/8, 5/4, 3/2, 1/1.

Staff III: Bass clef, notes with accidentals +42, -40. Rhythmic values: 5/4, 9/8.

Staff \*: Treble clef, notes with accidentals +7, -44, +42, -3, -42, -44, +25, +42. Rhythmic values: 11/8, 1/1, 5/4, 3/2, 1/1, 7/4, 5/4, 13/8.

15.1

III 4/3 II 12/11 I  
-44 -44 -42 -44 +7

677

Staff I: Treble clef, notes with accidentals -42, +7. Rhythmic values: 3/2, 11/8, 1/1, 4/3, 16/11, 7/8.

Staff II: Bass clef, notes with accidentals -44. Rhythmic value: 1/1.

Staff III: Bass clef, notes with accidentals -44. Rhythmic value: 1/1.

Staff \*: Treble clef, notes with accidentals -44, +25, -42, -3, -42. Rhythmic values: 1/1, 7/4, 3/2, 13/8, 3/2.

681

Staff I: Treble clef, notes with accidentals +7, +11, +7, -41, +9, +7. Rhythmic values: 1/1, 9/8, 1/1, 7/4, 11/8, 3/2, 1/1.

Staff II: Bass clef, notes with accidentals +44, -1, +27, -40, -42, -1. Rhythmic values: 13/8, 5/4, 7/4, 3/2, 1/1, 13/8.

Staff III: Bass clef, notes with accidentals +25, +7. Rhythmic values: 7/4, 11/8.

Staff \*: Treble clef, notes with accidentals +44, -40, +42, +42, +7. Rhythmic values: II 5/4, III 9/8, 5/4, 5/4, 11/8.

**15.2** 16/11

III 12/11 II 4/3 I  
-44 +7 -42 +7 -44

685 16/11 1/1 16/11

7/4 5/4 9/8 11/8 7/4 3/2 5/4 3/2 7/4 11/8  
13/8 1/1 9/8 13/8

5/4 11/8 3/2 1/1 7/4 5/4

3/2 13/8 1/1 3/2 7/4 6/5 1/1 9/8 13/8  
3/2

III 6/5 I 7/4 II 5/4

**15.3** 14/11

III 1/1 II 14/11 I  
+7 -42 +7 -44 +25

689 1/1 12/11 8/7

5/4 11/8 1/1

1/1 3/2 3/2 6/5

11/8 13/8 6/5

III 11/8 3/2 7/4 I 7/4

**15.4** 1/1

III 14/11 II 14/11 I  
+7 +7 +25 +25 +7

693 1/1 14/11 14/11

9/8 6/5 13/8 9/8

7/4 9/8 13/8 1/1

6/5 3/2 1/1

III 9/8 6/5 II 9/8

15.5

III 14/11 II 14/11 I

+7 +25 +25 +7

14/11 1/1 1/1

697

I

II +27 +25 -35 -6 -24 +25 +29  
3/2 1/1 13/8 7/4 11/8 1/1 9/8

III -24 +48 +23 +9 +11  
7/4 13/8 3/2 9/8  
6/5

\* +48 -24  
III 13/8 7/4

701

I +9 -24 +48 +7  
3/2 7/4 13/8 1/1  
6/5

II +27 -35 +25 +11 +29  
3/2 13/8 1/1 5/4 9/8

III -24 -35 +25  
11/8 13/8 1/1

\* -24 +27 -35 -6 +23 +29  
7/4 13/8 7/4 9/8  
III 3/2 1/6/5

15.6

III 14/11 II 1/1 I

+25 +7 +25 +7 +25

14/11 1/1 14/11

705

I +27 +29 -35  
9/8 13/8 3/2

II -6 -35 -35 +27  
7/4 11/8 13/8 3/2  
13/8

III +9  
3/2

\* +48  
I 13/8

**15.7**

III 1/1 II 14/11 I  
+7 +25 +25 +25 +7

14/11 1/1 14/11

709

I -24 +27  
11/8 3/2

II +11 +25  
5/4 1/1

III -24 +23 +7 +48 +11 +23 -24  
7/4 6/5 1/1 13/8 9/8 7/4  
7/4 6/5

\* -24 +27  
III 7/4 II 3/2

713

I +7 -24 +48 +11 -24 +7 +9 -41  
1/1 7/4 9/8 13/8 7/4 1/1 6/5 3/2 11/8

II +27  
3/2

III -24  
11/8

\* -35 -6 +29 +11  
III 13/8 7/4 9/8 5/4

**15.8**

III 1/1 II 1/1 I  
+25 +25 +7 +25

1/1 1/1 14/11

717

I +11 +25 -35 +29 -24 +11 +29  
5/4 1/1 13/8 9/8 11/8 3/2 9/8  
11/8 5/4

II

III -35 +25 +27 +11 +29 +27 -6 +11 +25 +29 -24 +27 +25 -35  
1/1 3/2 5/4 9/8 3/2 7/4 5/4 1/1 9/8 11/8 3/2 1/1 13/8  
13/8 1/1 3/2

\* -35 +11  
III 13/8 5/4

721

Score for measures 721-724. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 721 has a +11 fret shift in staff I and a 5/4 time signature. Measure 722 has fret shifts of +25, -6, +27, -24, +25 in staff II and time signatures of 1/1, 7/4, 3/2, 11/8, 1/1. Measure 723 has a +11 fret shift in staff III and a 5/4 time signature. Measure 724 has fret shifts of +29, -24, -6, +29 in staff \* and time signatures of 9/8, 11/8, 1/1, 7/4, 9/8.

16.1

Diagram 16.1 shows three fretted notes on a staff. The first note is on the 9th fret with a 9/8 time signature and a +25 fret shift. The second note is on the 1st fret with a 1/1 time signature and a +29 fret shift. The third note is on the 1st fret with a 1/1 time signature and a +25 fret shift.

725

Score for measures 725-728. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 725 has a -35 fret shift in staff I and a 13/8 time signature. Measure 726 has a +11 fret shift in staff II and a 5/4 time signature. Measure 727 has a +25 fret shift in staff III and a 1/1 time signature. Measure 728 has a -2 fret shift in staff III and a 7/4 time signature. Staff \* has fret shifts of +27, +11, +44, +44, +27 and time signatures of 3/2, 5/4, III<sup>6/5</sup>, 6/5, II<sup>3/2</sup>.

16.2

Diagram 16.2 shows three fretted notes on a staff. The first note is on the 9th fret with a 9/8 time signature and a +29 fret shift. The second note is on the 9th fret with a 9/8 time signature and a +25 fret shift. The third note is on the 1st fret with a 1/1 time signature and a +29 fret shift.

729

Score for measures 729-732. It consists of four staves: I (treble clef), II (alto clef), III (bass clef), and \* (treble clef). Measure 729 has fret shifts of +27, -6, +25 in staff I and time signatures of 3/2, 7/4, 1/1. Measure 730 has fret shifts of -6, +29, +25, -35, +11 in staff II and time signatures of 7/4, 9/8, 1/1, 13/8, 5/4. Measure 731 has fret shifts of -31, +33, +29 in staff III and time signatures of 13/8, 1/1, 9/8. Measure 732 has a -6 fret shift in staff \* and time signatures of 13/8, 7/4.



733

5/4 7/4 3/2 11/8 1/1 13/8 7/4

13/8 9/8

11/8 3/2 1/1 7/4 13/8 11/8 7/4 3/2

11/8

III 9/8 5/4 11/8 III 13/8 7/4 9/8 5/4 II 6/5

II 6/5

16.3

9/8 9/8 1/1 I

+25 +29 +29 +25 +25

737

9/8 9/8 1/1 7

9/8

7/4

III 6/5

741

9/8 5/4 1/1

5/4 13/8 1/1

6/5 1/1

11/8 II 13/8

16.4

18/13

III 9/8 II 16/13 I  
 +29 +25 +25 -35  
 1/1 1/1 16/13

745

749

16.5

9/8

III 18/13 II 16/13 I  
 +29 +25 -35 -35 +25

1/1 16/13 16/13

753

16.6

III 18/13 II 1/1 I

+29 -35 -35 +29 +25 +29

757

I 18/13 18/13 9/8 +29 -20

II -31 -20 +31 +44 -31

III +44 -2 -29 -33 -31 -33

6/5 11/8 1/1 3/2 9/8 3/2

7/4 11/8 7/4 3/2 6/5 13/8

\* +6 +44

III<sup>13/8</sup> II<sup>6/5</sup>

16.7

III 18/13 II 1/1 I

-35 +29 +29 -35 +29 -35

761

I 18/13 18/13 18/13

II -2 +17 -35 -48 -35

7/4 1/1 11/8 1/1 5/4

III +6 +31 +44 -20 +33 +29

13/8 3/2 6/5 11/8 9/8 1/1

7/4

\* -20 +31 -2

11/8 3/2 III<sup>7/4</sup>

765

I +34 -31 -35 +6

7/4 9/8 1/1 13/8

II

III

\* +17 -33 +6

II<sup>11/8</sup> 3/2 13/8

**16.8** 1/1

III 18/13 II 18/13 I  
+29 -35 -35 +29 -35

769 18/13 18/13 1/1

I  
-33 -31 +17 -35  
3/2 9/8 11/8 1/1

II  
-20 +33  
+31 +31 +29  
3/2 3/2 1/1  
11/8 9/8

III  
+6 +17 -33 +34 -48  
13/8 11/8 3/2 7/4 5/4

\* +34  
III 7/4

**16.9** 1/1

III 18/13 II 18/13 I  
-35 +29 -35

773 1/1 1/1 1/1

I

II  
-20 +31 -2 -20 +44 -31  
11/8 7/4 11/8 6/5 13/8  
3/2

III  
-33 +17 -31 -35  
3/2 11/8 9/8 1/1

\* -20 +31 -48  
II 11/8 3/2 III 5/4

**16.10** 1/1

III 1/1 II 1/1 I  
-35 +29 -35 -35

777 1/1 18/13 1/1

I  
+34 +6 -33 -31  
13/8 7/4 3/2 9/8

II  
+31 +29 -2 +33 +31 +29 -20  
3/2 1/1 7/4 9/8 3/2 1/1 11/8

III

\* -33 +17  
11/8 3/2

to kill a monarch (seed:19800725)

781

1/1 5/4 11/8 3/2 9/8  
7/4 1/1

-35 -48 +17 -33 -31

II -31 -33 -35  
9/8 3/2 1/1

III +6 +34 +34 +17 -48 -33 +6  
7/4 13/8 7/4 3/2 11/8 5/4 3/2 13/8

\* -31 +17  
III 9/8 11/8

785

13/8 7/4

I +6 +34

II -48 -35 +6 -31 +17  
5/4 1/1 13/8 9/8 11/8

III

\* -35 -48 +6 -33 -31  
1/1 5/4 13/8 1/1 9/8  
3/2

789

I

II

III

\*

## appendix - SuperCollider code and Lilypond template

### tkam\_readme.scd

```
1 /*
2 ----execute
3 Execute tkam_main.scd to run.
4
5
6 ----transport tab
7 The play button will always start from the beginning of the current section.
8
9 The transport buttons allow you to advance by subsection (<,>) and section (<<>>).
10
11 Tempo change will only go into effect once the enter key or "set tempo" button is pressed.
12
13 The default seed given in the application will generate the first version of the music and score (as provided). Changing the seed will generate a new version with that
14 seed once the "generate" button is pressed. After the new version is generated, new Lilypond files can be generated by pressing the "transcribe" button. This will
15 create a tkam_score.ly file in a folder labeled "seed_[number]" which can be rendered by Lilypond. Note that the file must be rendered from that location as it
16 depends on files in that folder and the "includes" subfolder.
17
18 ----mixer tab
19 This allow individual control of each of the sonic elements. The three parts that can be played on acoustic instruments are automatically muted. The outputs will go out to
20 whatever sound card is being used by the system.
21 */
```

### tkam\_main.scd

```
1 (
2 // MAIN LAUNCH (loads necessary files and definitions)
3
4 var appEnvironment;
5
6 //push new environment
7 appEnvironment = Environment.make;
8 appEnvironment.push;
9
10 s.waitForBoot({
11     var preampBusses, accompBusses, postampBusses;
12
13     `hash = Date.getDate.hash.asString;
14     `cRes = 1;
15
16     // load all files
17     "tkam_musical_data_generator.scd".loadRelative;
18     "tkam_sonifier.scd".loadRelative;
19     "tkam_gui.scd".loadRelative;
20     "tkam_transcriber.scd".loadRelative;
21
22     # preampBusses, accompBusses, postampBusses = `allocBusses.value(s);
23     `defineSynths.value(s, preampBusses, accompBusses, postampBusses);
24
25     `genAll = {arg seed;
26         `dUnit = 0.reciprocal;
27         `musicData = `genMusicData.value(seed);
28         `scoreData = `genScoreData.value("musicData[0]");
29         `sectionData = "musicData[4]";
30         `patterns = `genPatterns.value("musicData[0]", "musicData[1]", "musicData[2]", "musicData[3]", `sectionData,
31             preampBusses, accompBusses, postampBusses);
32         `sectionNavDict = "musicData[5]";
33         `isPlaying = false;
34     };
35
36     `patternProxy = EventPatternProxy.new;
37
38     `tempoClock = TempoClock.new(1);
39     `dir = thisProcess.nowExecutingPath.dirname;
40     "loading app".postln;
41     `genAll.value(19800725);
42     `play = Synth.new(\masterPlayerControl ++ `hash);
43     4.collect({arg p; Synth.new(\clip ++ `hash, {\bin, accompBusses[p].index, \bus, postampBusses[5].index}});
44     `generateGUI.value(preampBusses, accompBusses, postampBusses);
45     "ready".postln;
46 });
47 appEnvironment.pop;
48 )
```

### tkam\_musical\_data\_generator.scd

```
1 (
2 // DATA GENERATOR - this file IS the piece
3 var frAdd, frDiff, frToFloat, frNearestInList, frCollapse, harmonicDistance,
4 genMode, hdChoose, wchooseDict, collectRoots,
5 initModeState, advanceMode,
6 initTemporalState, genTemporalData,
7 initPartStates, distributeRoots,
8 genEnsemblePart, genAccompPart, genBassPart, genAmpCurve, genMusicData, genScoreData, genPatterns;
9
10
11 //-----FREQUENCY RATIO MATH FUNCTIONS-----
12 //for frequency ratios in the form [numerator.factors, denominator.factors]
13 //we use arrays of factors in order to represent very complex ratios
14
15 //add
16 frAdd = {arg fr0, fr1;
17     var num, den;
18     num = fr0[0] ++ fr1[0];
19     den = fr0[1] ++ fr1[1];
20     [[1] ++ num.difference(den).sort, [1] ++ den.difference(num).sort]
21 };
22
23 //difference
24 frDiff = {arg fr0, fr1;
25     var res;
26     res = frAdd.value(fr0, fr1.reverse);
27     if(frToFloat.value(res) < 1, {res = res.reverse});
28     res
29 };
30
31 //convert to float
32 frToFloat = {arg fr; fr[0].asFloat.product / fr[1].asFloat.product};
33
34 //find nearest in list (not sharing the same root or itself)
35 frNearestInList = {arg frComp, frDict;
36     var frNearest, diffNearest, sub;
37     frNearest = nil;
38     diffNearest = 1000;
39     frDict.reject({arg item;
```

```

40     (item[\root][0] == frComp[1]) ||
41     (item[\fr] == frComp[1])
42   }).keys.asList.sort({arg a, b; harmonicDistance.value(a) < harmonicDistance.value(b)}).do({arg fr;
43     var diff = abs(frToFloat.value(fr) - frToFloat.value(frComp[0]));
44     if(diff < diffNearest, {diffNearest = diff; frNearest = fr});
45   });
46   frNearest
47 };
48
49 //collapse into one octave
50 frCollapse = {arg fr;
51   var res = fr;
52   while({frToFloat.value(res) >= 2}, {res = frAdd.value(res, [1, 2])});
53   while({frToFloat.value(res) < 1}, {res = frAdd.value(res, [2, 1])});
54   res
55 };
56
57 //harmonic distance
58 harmonicDistance = {arg fr; log2(fr[0].asFloat.product * fr[1].asFloat.product)};
59
60
61 //-----GENERATE MODE-----
62 genMode = {arg forceHS = false;
63   var mode, alternateProb;
64   alternateProb = [1, 0].wchoose({if(forceHS, {0}, {1}), 4}.normalizeSum);
65   mode = [
66     [1, 1],
67     [9, 8],
68     [[5, 4], [6, 5]].wchoose([3, if(forceHS, {0}, {1})].normalizeSum),
69     [[4, 3], [11, 8]].wchoose([alternateProb, 1].normalizeSum),
70     [3, 2],
71     [[8, 5], [13, 8]].wchoose([alternateProb, 1].normalizeSum),
72     [[15, 8], [7, 4]].wchoose([alternateProb, 1].normalizeSum)
73   ];
74   mode.collect({arg fr; [[1] ++ fr[0].factors, [1] ++ fr[1].factors]});
75 };
76
77
78 //-----CHOOSE AND COLLECT FUNCTIONS-----
79 hdChoose = {arg mode, exp = 1, weights = [1, 1, 1, 1, 1, 1, 1];
80   var probs;
81   probs = pow(1 / mode.collect({arg fr;
82     harmonicDistance.value(if(fr == [[1], [1]], {[[2], [1]]}, {fr}))), exp) * weights;
83   mode.wchoose(probs.normalizeSum)
84 };
85
86 wchooseDict = {arg dict, exp = 1, limit = 0, isFR = true;
87   var keyList, probs;
88   keyList = if(isFR, {
89     dict.keys.asList.sort({arg a, b; harmonicDistance.value(a) < harmonicDistance.value(b)});
90     {
91       dict.keys.asList.sort({arg a, b; a.convertDigits(2) < b.convertDigits(2)});
92     }
93   });
94   probs = keyList.collect({arg key;
95     var count = dict[key][\count];
96     if(count < limit, {0}, {count})
97   });
98   probs = pow(probs, exp);
99   keyList.wchoose(probs.normalizeSum)
100 };
101
102 collectRoots = {arg dict; dict.keys.collect({arg fr;
103   dict[fr][\root][0]).asList.sort({arg a, b; harmonicDistance.value(a) < harmonicDistance.value(b)});
104 };
105
106 //-----GENERATE MODE SEQUENCE-----
107 initModeState = {
108   var curModeState, frSet;
109   curModeState = Dictionary.new;
110   frSet = [[1, 1], [9, 8], [6, 5], [11, 8], [3, 2], [13, 8], [7, 4]].collect({arg fr;
111     [1] ++ fr[0].factors, [1] ++ fr[1].factors});
112   frSet.do({arg fr;
113     var mode, count;
114     mode = genMode.value;
115     count = if(fr == [[1], [1]], {10}, {1});
116     curModeState.add(fr->
117       Dictionary.with(*{\count->count, \mode->frSet, \root->[[[1], [1]], frSet], \mult->fr, \fr->fr}))
118   });
119   curModeState;
120 };
121
122 advanceMode = {arg lastModeState, lastCadenceState, forceHS = false;
123   var curModeState, curRoots, lastRoots, lastCadenceRoot, changeCount, modSpeed;
124
125   curModeState = lastModeState.deepCopy;
126   curRoots = collectRoots.value(curModeState);
127   lastRoots = collectRoots.value(lastModeState);
128   lastCadenceRoot = collectRoots.value(lastCadenceState).asList[0];
129   changeCount = 0;
130   modSpeed = if(forceHS, {1}, {[1, 2, 3].wchoose([2, 3, 1].normalizeSum)});
131
132   while({
133     (curRoots == lastRoots) && (changeCount < modSpeed) ||
134     (changeCount < modSpeed) && (curRoots.size > 1)
135   }, {
136     var roots, rootSel, mults, multProbs, multSel, new;
137
138     //bump for length of time its been around
139     curModeState.keysValuesDo({arg key, val;
140       val[\count] = val[\count] + 1;
141       if(val[\count] > 100, {val[\count] = 1});
142     });
143
144     // max 3 roots that were not the last cadencial root and only in the mode of the last cadence
145     roots = curModeState.reject({arg val;
146       (val[\fr] == lastCadenceRoot) ||
147       lastCadenceState.includesKey(val[\fr]).not || //consider two steps out?
148       (curRoots.size >= 3) && curRoots.includes(val[\fr]).not
149     });
150
151     rootSel = wchooseDict.value(roots, 1, 2);
152     mults = curModeState[rootSel][\mode];
153     multProbs = mults.collect({arg fr;
154       if(curModeState.keys.includes(frCollapse.value(frAdd.value(rootSel, fr))), {1}, {2})});
155     multSel = hdChoose.value(mults, 0.5, multProbs);
156     new = frCollapse.value(frAdd.value(rootSel, multSel));
157
158     curModeState[rootSel][\count] = curModeState[rootSel][\count] + 2; //bump if gets chosen as a root
159
160     if(curModeState.includesKey(new), {
161       //bump if it gets chosen again
162       curModeState[new][\count] = curModeState[new][\count] + 1;
163       if((curModeState[new][\count] >= 20), {
164         curModeState[new][\root] = [rootSel, curModeState[rootSel][\mode]]

```

```

165     });
166   }, {
167     var old;
168     //calculate nearest in list that does not share the same root
169     old = frNearestInList.value([new, rootSel], curModeState);
170     if(curModeState[old][\count] >= 20, {
171       var mode, root;
172       root = [rootSel, curModeState[rootSel][\mode]];
173       curModeState.add(new ->
174         Dictionary.with(*[\count->1,\mode->genMode.value(forceHS), \root->root, \mult->multSel, \fr->new]));
175       curModeState.removeAt(old);
176       changeCount = changeCount + 1;
177     })
178   });
179   curRoots = collectRoots.value(curModeState);
180 });
181 curModeState
182 };
183
184
185 //-----GENERATE TEMPORAL FRAMEWORK-----
186 initTemporalState = {
187   Dictionary.with(*([\count], [0, 1], [0, 1], [0, 1]).allTuples.collect({arg tuple;
188     tuple->Dictionary.with(*([\count->1]))))
189 };
190
191 genTemporalData = {arg lastTupleState, modeState, cadenceOverride, noParts = 4;
192   var cadence, curTupleState, timeToNextEvent, tuple, temporalData;
193
194   cadence = if(collectRoots.value(modeState).size == 1, {cadenceOverride.not}, {false});
195   curTupleState = lastTupleState.deepCopy;
196   timeToNextEvent = (64 + 50.rand + if(cadence, {50}, {0})).round(16);
197   tuple = wchooseDict.value(curTupleState, isFR: false);
198   if(cadence, {tuple = [1, 1, 1]});
199   curTupleState = curTupleState.keysValuesDo({arg key, val;
200     curTupleState[key][\count] = val[\count] + 1};
201   curTupleState[tuple][\count] = 0;
202   tuple = if(cadence, {[1, 1, 1]}, {[0] ++ tuple});
203
204   temporalData = noParts.collect({arg p;
205     var flourishDensity, genDensity, flourish, beforeLen, before, after, buffer;
206     flourishDensity = if(tuple[p] == 1, {0.125 + 0.5.rand}, {3});
207     if(p == 0) && cadence.not, {flourishDensity = 3};
208     genDensity = if(p == 0, {5}, {20});
209
210     flourish = (if(cadence, {16}, {8}) + 32.rand).collect({[0, 1].wchoose({flourishDensity, 1}.normalizeSum)});
211     buffer = 16.collect({0});
212     beforeLen = ((timeToNextEvent - flourish.size - buffer.size) / if(cadence, {1.25}, {1})).asInteger.rand;
213     before = beforeLen.collect({arg i; [0, 1].wchoose({genDensity, 0.25}.normalizeSum)});
214     after = (timeToNextEvent - before.size - flourish.size - buffer.size).collect({[0, 1].wchoose({genDensity, 1}.normalizeSum)});
215     flourish = before ++ flourish ++ after;
216     if(flourish.sum == 0, {flourish[flourish.size.rand] = 1});
217     flourish = buffer ++ flourish;
218   });
219   [temporalData, curTupleState]
220 };
221
222
223
224 //-----GENERATE ENSEMBLE PARTS-----
225 initPartStates = {
226   var allRatios;
227   allRatios = [[1, 1], [9, 8], [5, 4], [6, 5], [4, 3], [11, 8], [3, 2], [8, 5], [13, 8], [15, 8], [7, 4]];
228   Dictionary.with(*
229     4.collect({arg part;
230       part->Dictionary.with(*[\multCounts->
231         Dictionary.with(*allRatios.collect({arg fr;
232           [[1] ++ fr[0].factors, [1] ++ fr[1].factors->1]})),
233         \noteCount->0, \index->part, \lastFreq->0, \lastFreqRatio->[[1], [1]], \lastDur->0
234       })
235     })
236   );
237 };
238
239 //this is how roots are distributed to the parts
240 distributeRoots = {arg modeState, lastRoots;
241   var roots;
242   roots = modeState.keys.asList.collect({arg fr; modeState[fr][\root]});
243   roots = roots.asBag.contents.asPairs.reverse.clump(2);
244   roots = roots = roots.sort({arg a, b;
245     if(a[0] != b[0], {a[0] > b[0]}, {frToFloat.value(a[1][0]) > frToFloat.value(b[1][0])}).collect({arg item; item[1]}).wrapExtend(4);
246
247   roots = [roots[0]] ++ roots[..2].scramble;
248   roots = 4.collect({arg part;
249     var root, rootMod, rootFreq, mode;
250     root = roots[part];
251     rootMod = frDiff.value(root[0], lastRoots[part]);
252     rootFreq = 40.midicps * pow(2, [1, 0, 1, 2][part]) * frToFloat.value(root[0]);
253     [root[0], root[1], rootMod, rootFreq]
254   });
255   roots = roots.collect({arg root, r;
256     var rootRels;
257     rootRels = 4.collect({arg p; frDiff.value(root[0], roots[p][0])});
258     rootRels.removeAt(r);
259     root.add(rootRels)
260   });
261   roots
262 };
263
264 genEnsemblePart = {arg partState, modeState, temporalData, roots, part, offset;
265   var trans, root, mults, rootMod, amp, firstChange, cadence, lastInsRef, ensData;
266
267   trans = pow(2, [1, 0, 1, 2][partState[\index]]);
268   # root, mults, rootMod = roots[part];
269   amp = [0, 1, 2, 3].wchoose([0, 2, 2, 2].normalizeSum);
270   firstChange = false;
271   cadence = if(collectRoots.value(modeState).size == 1, {true}, {false});
272   lastInsRef = nil;
273
274   ensData = [];
275   temporalData.do({arg val, ts;
276     var timeStamp, comp, change;
277
278     partState[\lastDur] = partState[\lastDur] + 1;
279     timeStamp = ts + offset;
280     change = [val == 1, (val == 1) && firstChange.not].wchoose([1, 2].normalizeSum);
281     if(
282       (partState[\index] == 0) &&
283       (frToFloat.value(partState[\lastFreqRatio]) >= 4.0) &&
284       (partState[\lastDur] < 16) && cadence.not,
285       {change = false}
286     );
287
288     if(change, {
289       var mult, multWeights, freq, rootFreq, insRef;

```



```

290 //this weights notes that are richer and mixes with the DCA algorithm
291 multWeights = mults.collect({arg fr;
292     var comp = frCollapse.value(frAdd.value(root, fr));
293     if(modeState.keys.includes(comp), {3}, {1}) * pow(partState[\multCounts][fr], 1);
294 });
295
296 mult = hdChoose.value(mults, 0.5, multWeights);
297 mults.do({arg fr; partState[\multCounts][fr] = partState[\multCounts][fr] + 1});
298 partState[\multCounts][mult] = 0;
299
300 freq = 40.midicps * trans * frToFloat.value(frAdd.value(root, mult));
301
302 //flute special case
303 if((partState[\index] == 0) && cadence.not, {
304     var mode, continue, freqRatio;
305     mode = modeState.keys.asList.collect({arg fr;
306         [
307             frCollapse.value(frAdd.value(modeState[fr][\root][0], modeState[fr][\mult])),
308             modeState[fr][\root][0], modeState[fr][\mult]
309         ]
310     });
311     mode = mode.sort({arg a, b;
312         case
313             {frToFloat.value(a[0]) != frToFloat.value(b[0])} {frToFloat.value(a[0]) < frToFloat.value(b[0])}
314             {frToFloat.value(a[1]) != frToFloat.value(b[1])} {frToFloat.value(a[1]) < frToFloat.value(b[1])}
315             {true} {frToFloat.value(a[2]) < frToFloat.value(b[2])};
316     });
317     mode = mode ++ mode.collect({arg fr; [frAdd.value(fr[0], [[2], [1]]), fr[1], fr[2]]});
318     mode = mode ++ mode.collect({arg fr; [frAdd.value(fr[0], [[4], [1]]), fr[1], fr[2]]});
319     continue = true;
320     while({continue}, {
321         # freqRatio, root, mult = mode[partState[\noteCount] % 15];
322         freq = 40.midicps * trans * frToFloat.value(frAdd.value([1], [1], freqRatio));
323         continue = (freq <= partState[\lastFreq]) && (partState[\noteCount] % 15) != 0;
324         partState[\noteCount] = partState[\noteCount] + 1;
325     });
326     partState[\lastFreq] = freq;
327     partState[\lastFreqRatio] = freqRatio;
328     insRef = roots.slice(nil, 0).deepCopy.drop(1).indexOfEqual(root) + 1;
329     insRef = if(lastInsRef != insRef, {lastInsRef, insRef}, {lastInsRef, nil});
330 });
331 if((partState[\index] == 0) && cadence, {
332     insRef = if(firstChange.not, {1}, {nil});
333 });
334
335 rootFreq = 40.midicps * trans * frToFloat.value(root);
336
337 if((partState[\index] == 0) && (partState[\noteCount] % 15) == 1) && cadence.not, {ensData = ensData.add([0, timeStamp - 8, 0, 0, 0])};
338 ensData = ensData.add([freq, timeStamp, amp, mult, insRef]);
339 firstChange = true;
340 partState[\lastDur] = 0;
341 if((partState[\index] == 0) && cadence, {partState[\lastDur] = 32});
342 });
343
344 ensData = [[0, ensData[0][1] - 4, 0, 0, 0]] ++ ensData;
345 [ensData, partState]
346 };
347
348
349
350 //-----GENERATE ELECTRONIC ACCOMPANIMENT-----
351 genAccompPart = {arg modeState, temporalData, offset, trans, part, register;
352     var firstChange, accompData;
353     firstChange = false;
354     accompData = [];
355     temporalData.do({arg val, tS;
356         var change;
357         change = [val == 1, (val == 1) && firstChange.not].wchoose([1, if(part == 0, {5}, {3})].normalizeSum);
358         if(change, {
359             var sel, freq, amp;
360             sel = wchooseDict.value(modeState, 0.1);
361             freq = 48.midicps * trans * frToFloat.value(sel);
362             amp = [0, 1, 2, 3].wchoose([2, 2, 1, 1].normalizeSum);
363
364             accompData = accompData.add([freq, tS + offset, amp, part]);
365             firstChange = true;
366         });
367     });
368     accompData
369 };
370
371 //-----GENERATE ELECTRONIC BASS-----
372 genBassPart = {arg root, ampCurve, hi;
373     var freq;
374     freq = if(hi,
375         {40.midicps * frToFloat.value(frCollapse.value(frAdd.value(root, [[3], [2]]))},
376         {40.midicps * frToFloat.value(root)});
377     ampCurve.collect({arg sec, iter; [freq, sec[1]]})
378 };
379
380
381 //-----GENERATE AMP CURVES-----
382 genAmpCurve = {arg temporalData1, temporalData2, offset1, offset2, type;
383     var firsts1, firsts2, delay, attack, decay, release, min, max, env;
384     firsts1 = temporalData1.collect({arg ptd; ptd.indexOf(1)});
385     firsts2 = temporalData2.collect({arg ptd; ptd.indexOf(1)});
386     delay = switch(type)
387     {0} {0}
388     {1} {0}
389     {2} {firsts1.minItem};
390     attack = switch(type)
391     {0} {offset2 - offset1}
392     {1} {offset2 - temporalData2[0].size + firsts2.minItem - offset1}
393     {2} {firsts1.maxItem - firsts1.minItem};
394     decay = switch(type)
395     {0} {0}
396     {1} {firsts2.maxItem - firsts2.minItem}
397     {2} {temporalData1[0].size - firsts1.maxItem};
398     release = switch(type)
399     {0} {0}
400     {1} {temporalData2[0].size - firsts2.maxItem}
401     {2} {(offset2 - temporalData2[0].size) - offset1};
402     min = switch(type)
403     {0} {0.15}
404     {1} {0}
405     {2} {0};
406     max = switch(type)
407     {0} {0.5}
408     {1} {0.65}
409     {2} {1};
410
411     env = Env.dadsr(delay, attack, decay, 0.25, release, curve: \cub).range(min, max);
412     (delay + attack + decay + release) / 1.asInteger.collect({arg iter; [env.at(iter * 1), offset1 + (iter * 1)]})
413 };
414

```

```

415
416
417 //-----GENERATE ALL MUSIC DATA-----
418 ^genMusicData = {arg seed;
419   var minTotalDur, minSection1Dur, dUnit, curLen, cadence,
420   ultimateSubsection, ultimateSection, ultimateCadenceCount,
421   minTotalLen, minSection1Len,
422   modeState, temporalState, partStates,
423   lastCadenceTemporalData, lastCadenceState, lastSectionPoint,
424   ensData, accompData, bassData, ampData,
425   sectionData, sectionNavDict,
426   sectionCount, subsectionCount,
427   lastRoots, roots, ampDataTmp;
428
429   thisThread.randSeed = seed;
430
431   # minTotalDur, minSection1Dur, dUnit, curLen, cadence = [23 * 60, 8 * 60, 8.reciprocal, 0, false];
432   # ultimateSubsection, ultimateSection, ultimateCadenceCount = [false, false, 0];
433   # minTotalLen, minSection1Len = [(minTotalDur / dUnit).round(16), (minSection1Dur / dUnit).round(16)];
434   # modeState, temporalState, partStates = [initModeState.value, initTemporalState.value, initPartStates.value];
435   # lastCadenceTemporalData, lastCadenceState, lastSectionPoint = [nil, modeState.deepCopy, 0];
436   # ensData, accompData, bassData, ampData = [4.collect({[]}), 4.collect({6.collect({[]})}), 2.collect({[]}), 3.collect({[]})];
437   # sectionData, sectionNavDict = [Dictionary.new, Dictionary.new];
438   # sectionCount, subsectionCount = [1, 1];
439
440   while({(curLen < minTotalLen) || ((curLen >= minTotalLen) && cadence.not) || ultimateSection.not}, {
441     var temporalData;
442     # temporalData, temporalState = genTemporalData.value(temporalState, modeState, curLen <= minSection1Len);
443
444     collectRoots.value(modeState).collect({arg fr; [fr[0].asFloat.product, fr[1].asFloat.product]}).postln;
445     //modeState.keys.postln;
446     ("-- " ++ sectionCount ++ " " ++ subsectionCount ++ " ----").postln;
447
448     lastRoots = if(curLen == 0, {4.collect({[1], [1]})}, {roots.slice(nil, 0)});
449     roots = distributeRoots.value(modeState, lastRoots);
450
451     sectionData.add((curLen / 4).asInteger->[roots, lastRoots.collect({arg fr, part;
452       [fr, 40.midicps * pow(2, [1, 0, 1, 2][part]) * fr.toFloat.value(fr)]]}, sectionCount, subsectionCount, cadence, ultimateSubsection));
453     sectionNavDict.add([sectionCount, subsectionCount]->[(curLen / 16 + 1).asInteger]);
454
455     4.do({arg part;
456       var musicData, partState, accompRoutine;
457       # musicData, partState = genEnsemblePart.value(partStates[part], modeState, temporalData[part], roots, part, curLen);
458       ensData[part] = ensData[part] ++ musicData;
459       partStates[part] = partState;
460
461       //use an independent random number generator for the accompaniment
462       accompRoutine = Routine({
463         thisThread.randSeed = Date.seed;
464         6.do({arg register;
465           musicData = genAccompPart.value(modeState, temporalData[part], curLen, pow(2, part + register), part, register);
466           accompData[part][register] = accompData[part][register] ++ musicData;
467         });
468       });
469       accompRoutine.value;
470
471     });
472
473     subsectionCount = subsectionCount + 1;
474
475     if(curLen == 0, {
476       lastCadenceTemporalData = temporalData;
477       lastSectionPoint = curLen;
478     });
479
480     curLen = curLen + temporalData[0].size;
481
482     if(curLen > minSection1Len, {
483       if(collectRoots.value(modeState).size == 1, {
484
485         ampData[0] = ampData[0] ++ genAmpCurve.value(lastCadenceTemporalData, temporalData, lastSectionPoint, curLen, 0);
486         ampDataTmp = genAmpCurve.value(lastCadenceTemporalData, temporalData, lastSectionPoint, curLen, 1);
487         ampData[1] = ampData[1] ++ ampDataTmp;
488         bassData[0] = bassData[0] ++ genBassPart.value(collectRoots.value(modeState).asList[0], ampDataTmp, true);
489
490         if(sectionCount == 1, {
491           ampData[2] = ampData[2] ++ ((curLen - temporalData[0].size) / 1).asInteger.collect({arg iter; [0, iter * 1]});
492           bassData[1] = bassData[1] ++ ((curLen - temporalData[0].size) / 1).asInteger.collect({arg iter; [0, iter * 1]});
493         }, {
494           ampDataTmp = genAmpCurve.value(lastCadenceTemporalData, temporalData, lastSectionPoint, curLen, 2);
495           ampData[2] = ampData[2] ++ ampDataTmp;
496           bassData[1] = bassData[1] ++ genBassPart.value(collectRoots.value(lastCadenceState).asList[0], ampDataTmp, false);
497         });
498
499         sectionData.add((curLen - temporalData[0].size) / 4).asInteger->
500         sectionData[(curLen - temporalData[0].size) / 4].asInteger.put(5, true);
501         (subsectionCount - 1).do({arg subsectionIndex;
502           sectionNavDict.add([sectionCount, subsectionIndex + 1]->[sectionNavDict[[sectionCount, subsectionIndex + 1]][0], subsectionCount - 1]);
503         });
504         # sectionCount, subsectionCount = [sectionCount + 1, 1];
505         # lastCadenceTemporalData, lastCadenceState, lastSectionPoint = [temporalData, modeState, curLen];
506         cadence = true;
507
508         //this should ensure that the final cadence is a HS
509         if(curLen >= minTotalLen, {ultimateCadenceCount = ultimateCadenceCount + 1});
510         ultimateSection = ultimateCadenceCount > 1;
511       }, {
512         cadence = false
513       });
514       modeState = advanceMode.value(modeState, lastCadenceState, curLen >= minTotalLen);
515     });
516   });
517
518   ampDataTmp = genAmpCurve.value(lastCadenceTemporalData, [[1], [1], [1], [1]], lastSectionPoint, lastSectionPoint + 360, 2);
519   ampData[2] = ampData[2] ++ ampDataTmp;
520   bassData[1] = bassData[1] ++ genBassPart.value(collectRoots.value(lastCadenceState).asList[0], ampDataTmp, false);
521
522   [ensData, accompData, bassData, ampData, sectionData, sectionNavDict]
523 };
524 )

```

## tkam\_sonifier.scd

```

1 (
2   var formatPatternData;
3
4   //-----ALLOCATE BUSSES-----
5   ^allocBuses = {arg server;
6     var preampBuses, accompBuses, postampBuses;
7     preampBuses = 3.collect({Bus.audio(server, 1)});
8     accompBuses = 4.collect({Bus.audio(server, 1)});
9     postampBuses = 7.collect({Bus.audio(server, 1)});
10    [preampBuses, accompBuses, postampBuses];

```

```

11 };
12
13 //-----DEFINE SYNTHS-----
14 ^defineSynths = {arg server, preampBusses, accompBusses, postampBusses;
15   var sdPlayer, sdTransport, sdClick, sdAmpCurve, sdEns, sdAccomp, sdClip, sdBass, sdDiskOut, allSds;
16
17   sdPlayer = SynthDef(\masterPlayerControl ++ ^hash, {
18     var router, sigs, sigsPanned, masterSig, imp;
19
20     sigs = postampBusses.collect({arg bus, i; In.ar(bus) * NamedControl.kr(\vol ++ i, 1, 0.1) * NamedControl.kr(\mute ++ i, 1, 0.1)});
21     router = sigs.collect({arg sig, i; NamedControl.kr(\out ++ i, 0, 0)});
22     sigs.collect({arg sig, i; Out.ar(router[i] - 1, sig * router[i].sign)});
23
24     sigsPanned = sigs.collect({arg sig, i; Pan2.ar(sig, NamedControl.kr(\pan ++ i, 0, 0.1)});
25     masterSig = Mix.ar(sigsPanned.collect({arg sig, i; sig * abs(router[i].sign - 1)}));
26     masterSig = masterSig * NamedControl.kr(\masterVol, 1, 0.1) * NamedControl.kr(\masterMute, 1, 0.1);
27
28     Out.ar(NamedControl.kr(\masterOut, 0, 0), masterSig);
29
30     imp = Impulse.kr(10);
31     SendReply.kr(imp, 'masterLevels.' ++ ^hash, values: [Amplitude.kr(masterSig)]);
32     sigs.collect({arg sig, i; SendReply.kr(imp, 'trackLevel.' ++ i ++ "-" ++ ^hash, values: [Amplitude.kr(sig)])});
33   });
34
35   sdTransport = SynthDef(\transport ++ ^hash, {arg measure = 0, beat = 0, section = 0, subsection = 0, gate = 1, dur = 1;
36     SendReply.kr(Impulse.kr(0) * (measure > 0) * (beat > 0), 'measureClock.' ++ ^hash, values: [measure, beat, section, subsection]);
37     EnvGen.kr(Env.sine(dur), gate, doneAction: 2);
38   });
39
40   sdClick = SynthDef(\click ++ ^hash, {arg beat = 0, gate = 1, dur = 1;
41     Out.ar(postampBusses[6], 10 * BPF.ar(WhiteNoise.ar * EnvGen.kr(Env.perc(0.01, 0.1), gate), 440 * ((beat <= 1) + 1), 0.02));
42     EnvGen.kr(Env.sine(dur), gate, doneAction: 2);
43   });
44
45   sdAmpCurve = SynthDef(\amp.curve ++ ^hash, {arg amp = 1, dur = 0.1, bus = 0;
46     Out.kr(bus, amp.lag)
47   });
48
49   sdEns = SynthDef(\ens ++ ^hash, {arg freq = 440, amp = 1, dur = 1, gate = 1, bus = 0, ampBus = 0, rel = 0.1;
50     Out.ar(bus, SinOsc.ar(freq, 2pi.rand, 0.1) * amp * Latch.kr(In.kr(ampBus), Impulse.kr(0)) * EnvGen.kr(Env.asr(0.1, 1, rel), gate, doneAction: 2));
51   });
52
53   sdAccomp = SynthDef(\accomp ++ ^hash, {arg freq = 440, amp = 1, sustain = 1, dur = 1, gate = 1, bout = 0, ampBus = 0, rel = 0.01;
54     Out.ar(bout, SinOsc.ar(freq, 2pi.rand, 1) * 0.01 * amp * Latch.kr(In.kr(ampBus), Impulse.kr(0)) * EnvGen.kr(Env.asr(sustain, 1, rel), gate, doneAction: 2));
55   });
56
57   sdClip = SynthDef(\clip ++ ^hash, {arg dur = 1, gate = 1, bin = 0, bus = 0;
58     Out.ar(bus, (In.ar(bin)).clip(0, 1) * 50)
59   });
60
61   sdBass = SynthDef(\bass_mono ++ ^hash, {arg freq = 440, ampBus = 0, bus = 0;
62     Out.ar(bus, (SinOsc.ar(freq) * 0.5 * In.kr(ampBus)))
63   });
64
65   sdDiskOut = SynthDef(\disk.out ++ ^hash, {arg bufnum, inbus;
66     DiskOut.ar(bufnum, In.ar(inbus));
67   });
68
69   allSds = [sdPlayer, sdTransport, sdClick, sdAmpCurve, sdEns, sdAccomp, sdClip, sdBass, sdDiskOut];
70   allSds.do({arg sd; sd.add});
71   allSds
72 };
73
74
75 // group data by measures for navigation
76 formatPatternData = {arg musData, measureLen, rel, print = false;
77   var dataLen;
78   dataLen = musData[0][0].size + 1;
79   musData.collect({arg partData;
80     var res;
81     res = partData;
82     res = res.collect({arg mData, index; mData.add(if(index != (res.size - 1), {rel}, {5.rand + 5})});
83     res = res.flop;
84     res = res.add(res[1]);
85     res[1] = (res[1].differentiate.drop(1) ++ [10]);
86     res = res.flop ++ measureLen.collect({arg measure; dataLen.collect({0}) ++ [measure * 16]});
87     res = res.sort({arg a, b; a.last < b.last}).flop;
88     res = res.insert(1, (res.last.differentiate.drop(1) ++ [10])).flop;
89     res = res.separate({arg a, b; (a.last / 16).trunc != (b.last / 16).trunc});
90     res.collect({arg measureData; measureData.flop})
91   }).flop
92 };
93
94
95 //-----GENERATE PATTERNS-----
96 //this generates patterns grouped by measures except for bass data and amp curve data which are much higher resolution
97 //these are used to make playable patterns
98 ^genPatterns = {arg ensData, accompData, bassData, ampData, sectionData, preampBusses, accompBusses, postampBusses;
99   var measureLen, ensDataFormatted, accompDataFormatted, bassDataFormatted, ampDataFormatted,
100     dUnit, section, subsection, patterns;
101
102   measureLen = ((
103     ensData.collect({arg partData; partData.last[1]}) ++
104     accompData.flatten.collect({arg partData; partData.last[1]})
105   ).maxItem.ceil(16) / 16).asInteger + 1;
106
107   ensDataFormatted = formatPatternData.value(ensData, measureLen, 0.1, true);
108   accompDataFormatted = formatPatternData.value(accompData.flatten, measureLen, 0.01);
109   dUnit = 8.reciprocal;
110
111   patterns = measureLen.collect({arg measure;
112     if(sectionData[measure * 4] != nil, {
113       section = sectionData[measure * 4][2];
114       subsection = sectionData[measure * 4][3];
115     });
116     Ppar(
117       //check how amplitude is being handled
118       ensDataFormatted[measure].collect({arg musData, p;
119         Pbind(
120           \instrument, \ens ++ ^hash,
121           \freq, Pseq(musData[0].replace(0, Rest(0))),
122           \dur, Pseq(musData[1] * dUnit),
123           \sustain, Pseq(musData[2] * dUnit),
124           \amp, [1, 0.7, 0.5, 0.3][p],
125           \ampBus, preampBusses[0].index,
126           \bus, postampBusses[p].index,
127           \rel, Pseq(musData[6])
128         )
129       }) ++
130       //check how amplitude and attack are being handled
131       accompDataFormatted[measure].collect({arg musData;
132         Pbind(
133           \instrument, \accomp ++ ^hash,
134           \freq, Pseq(musData[0].replace(0, Rest(0))),
135           \dur, Pseq(musData[1] * dUnit),

```

```

136         \sustain, Pseq(musData[2] * dUnit),
137         \amp, Pseq(musData[3].collect({arg item; [0, 2, 4, 8][item]} * 0.0125 * 1),
138         \ampBus, preampBusses[0].index,
139         \bout, Pseq(musData[4].collect({arg index; accompBusses[index].index})),
140         \rel, Pseq(musData[5])
141     )
142 } ++
143 [
144     Pbind(
145         \instrument, \transport. ++ `hash,
146         \measure, measure + 1,
147         \beat, Pseq([1, 2]),
148         \section, section,
149         \subsection, subsection,
150         \dur, 8 * dUnit
151     ),
152     Pbind(
153         \instrument, \click. ++ `hash,
154         \beat, Pseq([1, 2]),
155         \dur, 8 * dUnit
156     )
157 ]
158 );
159 [patterns, bassData, ampData]
160 };
161
162
163
164 //this is a playable pattern based on start measure
165 ^genPlayablePatterns = {arg startMeasure, patterns, preampBusses, accompBusses, postampBusses;
166     Ppar(
167         [Pseq(patterns[0][startMeasure..], 1)] ++
168         patterns[2].collect({arg pattern, p;
169             Pmono(\amp_curve. ++ `hash,
170                 \amp, Pseq(Pseq(pattern.slice(nil, 0)[(startMeasure * 16)..], 1), 1 * ^dUnit, \cub),
171                 \dur, 1 * ^dUnit,
172                 \bus, preampBusses[p].index
173             )
174         } ++
175         patterns[1].collect({arg pattern, p;
176             Pmono(\bass_mono. ++ `hash,
177                 \freq, Pseq(Pseq(pattern.slice(nil, 0)[(startMeasure * 16)..], 1),
178                 \dur, 1 * ^dUnit,
179                 \ampBus, preampBusses[p + 1].index,
180                 \bus, postampBusses[4].index
181             )
182         }
183     )
184 };
185
186
187
188 //-----BOUNCE AUDIO-----
189 //this bounces the audio for use in another DAW or for practice
190 ^bounceAudio = {arg seed;
191     var trackNames, basePath, server, buffers, recDur,
192     preampBusses, nextNode, accompBusses, postampBusses,
193     synths, prePatterns, playablePatterns, score;
194
195     trackNames = ["part.star", "part.III", "part.II", "part.I", "accomp.II", "accomp.I", "click"];
196
197     basePath = ^dir "+/" + "." + "/" + "audio" + "/" + "seed." ++ seed;
198     basePath.mkdir;
199
200     server = Server(\nrt. ++ `hash,
201         options: ServerOptions.new
202         .numOutputBusChannels(7)
203         .numInputBusChannels(0)
204     );
205
206     # preampBusses, accompBusses, postampBusses = ^allocBusses.value(s);
207     postampBusses = 7.collect({arg index; Bus.new(rate: 'audio', index: index, numChannels: 1, server: server)});
208     synths = ^defineSynths.value(s, preampBusses, accompBusses, postampBusses);
209
210     prePatterns = ^genPatterns.value("musicData[0]", "musicData[1]", "musicData[2]", "musicData[3]", ^sectionData,
211         preampBusses, accompBusses, postampBusses);
212
213     playablePatterns = ^genPlayablePatterns.value(0, prePatterns, preampBusses, accompBusses, postampBusses);
214
215     recDur = (prePatterns[2][0].size / 8) + 45;
216     score = playablePatterns.asScore(duration: recDur, timeOffset: 0.001);
217     nextNode = score.score.slice(nil, 1).select({arg msg; msg[0] == 9}).slice(nil, 2).maxItem + 1;
218
219     synths.do({arg synth; score.add([0.0, [\d.recv, synth.asBytes]]);});
220
221     4.collect({arg p;
222         score.add([0.0, [\s.new, \clip. ++ `hash, nextNode, 1, 1, \bin, accompBusses[p].index, \bus, postampBusses[5].index]]);
223         nextNode = nextNode + 1;
224     });
225
226     buffers = 7.do({arg track;
227         score.add([0.0, [\b.alloc, track, 65536, 1]]);
228         score.add([0.0, [\b.write, track, basePath + "/" + "tkam." ++ trackNames[track] + ".wav".standardizePath, "WAV", "int16", 0, 0, 1]]);
229         score.add([0.0, [\s.new, \diskout. ++ `hash, nextNode, 1, 1, \bufnum, track, \inbus, track]]);
230         score.add([recDur, [\n.free, nextNode]]);
231         score.add([recDur, [\b.close, track]]);
232         score.add([recDur, [\b.free, track]]);
233         nextNode = nextNode + 1;
234     });
235
236     score.sort;
237
238     score.recordNRT(
239         outputPath: basePath + "/" + "tkam.all" ++ ".wav".standardizePath,
240         sampleRate: 44100,
241         headerFormat: "WAV",
242         sampleFormat: "int16",
243         options: server.options,
244         duration: recDur
245     );
246
247     server.remove;
248 }
249 )

```

## tkam\_transcriber.scd

```

1 (
2 var formatMusicData, spellingDict, lyNoteNameStr, lyOctStr, lyFinalizeMusic, lyMeasureDef,
3 lyRelMark, lyRelMarkNote, lyHBracket, lyStaffDef, lyTie,
4 lyNoteName, lyCentDev, lyFreqRatio, lyDur, lyNote, lyBeamOpen, lyBeamClosed,
5 consolidateNotes, consolidateRests;
6

```

```

7 // formats the data for the transcriber
8 formatMusicData = {arg rawMusicData;
9   var maxSize, musicData;
10  maxSize = 0;
11  musicData = rawMusicData.collect({arg partData, p;
12    var res;
13    res = partData.collect({arg item, i;
14      var freq, dur, amp, mult, insRef, sus, note, rest;
15      # freq, dur, amp, mult, insRef = item;
16      sus = dur * sign(amp);
17      note = sus.collect({[freq, mult, insRef, i]});
18      rest = if(p < rawMusicData.size, {(dur - sus).collect({[-1, -1, -1, i]}), {[]}};
19      note ++ rest
20    }).flatten;
21    if(res.size > maxSize, {maxSize = res.size});
22    res
23  });
24
25 //make them all the same length
26 maxSize = maxSize.trunc(64) + 64;
27 musicData = musicData.collect({arg partData, p; partData.extend(maxSize, partData.last)});
28 musicData
29 };
30
31 // constants (spelling dictionary note names and octaves)
32 spellingDict = Dictionary.with(*
33 [
34   \major -> Dictionary.with(*
35     [0, 7, 2, 9, 4, 11].collect({arg pc; pc->\sharps}) ++
36     [5, 10, 3, 8, 1, 6].collect({arg pc; pc->\flats})
37   ),
38   \minor -> Dictionary.with(*
39     [9, 4, 11, 6, 1, 8].collect({arg pc; pc->\sharps}) ++
40     [2, 7, 0, 5, 10, 3].collect({arg pc; pc->\flats})
41   )
42 ]
43 );
44
45 lyNoteNameStr = Dictionary.with(*
46 [
47   \sharps -> ["c", "cis", "d", "dis", "e", "f", "fis", "g", "gis", "a", "ais", "b"],
48   \flats -> ["c", "des", "d", "ees", "e", "f", "ges", "g", "aes", "a", "bes", "b"],
49 ]
50 );
51
52 lyOctStr = ["", "1", "2", "3", "4", "5", "6", "7", "8"];
53
54 //define staff
55 lyStaffDef = {arg name, nameShort, nameMidi;
56   "\new Staff = \"\" ++ name ++ \"\n\" \\\with { \n\" ++
57   "instrumentName = \"\" ++ name ++ \"\n\" ++
58   "shortInstrumentName = \"\" ++ nameShort ++ \"\n\" ++
59   "midiInstrument = #\"\" ++ nameMidi ++ \"\n\" ++
60   "\n\"
61 };
62
63 // add music preamble
64 lyFinalizeMusic = {arg lyStr, part, name, nameShort, nameMidi, clef;
65   "\new StaffGroup \\\with {\\remove \\system.startDelimiterEngraver}\"\\n<<\n\" ++
66   lyStaffDef.value(name, nameShort, nameMidi) ++
67   <<<\n\"
68   //\"\\set Score.markFormatter = #format-mark-box-numbers \" +
69   "\tempo 2 = 60\n\"
70   "\numericTimeSignature \\time 2/2\n\" +
71   "\clef \" ++ clef ++ \"\n\" ++ lyStr ++ \"\\fermata\" +
72   "\>> \\bar |.|.\" \n\" \n\">>\" ++
73   "\n\">>\"
74 };
75
76 lyRelMarkNote = {arg root, lastRoot, part, clef;
77   if(root[part][2] != [1], [1]), {
78     "\stopStaff s8. \\startStaff \\clef\" + clef + "s16 \n\" ++
79     "\once \\override TextScript.color = #(rgb-color 0.6 0.6 0.6) \n\" ++
80     "\tweak Accidental.color #(rgb-color 0.6 0.6 0.6) \n\" ++
81     "\tweak NoteHead.color #(rgb-color 0.6 0.6 0.6) \n\" ++
82     lyNote.value(lastRoot[part][1], 1, lastRoot[part][0], nil, \sharps, true, true, false) +
83     "\hide c\" ++ [nil, "", "", "''", "''''"][part] ++ "8 \n\"
84   }, {
85     "\stopStaff s4. \\startStaff \\clef\" + clef + "s16 \n\"
86   } ++
87   lyNote.value(root[part][3], 1, root[part][2], nil, \sharps, true, false, true)
88 };
89
90 lyHBracket = {arg fr, yOffset, sPair1, sPair2, edgeH1, edgeH2;
91   "-\\tweak HorizontalBracket.Y-offset #\"\" ++ yOffset ++ \"\n\" ++
92   "-\\tweak HorizontalBracket.shorten-pair #'(\" ++ sPair1 + \".\" + sPair2 ++ \"\") \n\" ++
93   "-\\tweak HorizontalBracket.edge-height #'(\" ++ edgeH1 + \".\" + edgeH2 ++ \"\") \n\" ++
94   "-\\tweak HorizontalBracketText.text\" + fr + \"\\startGroup \n\"
95 };
96
97 lyRelMark = {arg root, lastRoot, section, subsection;
98   var sectionMark;
99   sectionMark = "\mark \\markup { \\bold \\override #'(box-padding . 0.5) \\box \" ++ section ++ \".\" ++ subsection ++ \" } \n\";
100  if((section == 1) && (subsection > 1),
101    {
102      "\once \\override Score.RehearsalMark.self-alignment-X = #0 \n\" ++
103      "\once \\override Score.RehearsalMark.Y-offset = #5 \n\" ++
104      "\once \\override Score.RehearsalMark.X-offset = #1 \n\" ++
105      sectionMark
106    }, {
107      "\mark \\markup { \n\" ++
108      "\halign #-1 \n\" ++
109      "\relMark ##{ { \n\" ++
110      "\time 15/8 \n\" ++
111      "\once \\override Staff.Clef.stencil = ##f \n\" ++
112      sectionMark ++
113
114      lyRelMarkNote.value(root, lastRoot, 1, "bass") ++ "\markup{\\large \\raise #2 \"III\"} ++
115
116      lyHBracket.value(lyFreqRatio.value(root[1][4][2], nil, true, 0, false), 8.5, 1, 2, 1, 1) ++
117      lyHBracket.value(lyFreqRatio.value(root[2][4][1], nil, true, 0, false), 5.5, 3, 3, 0, 0) ++
118
119      "\hide c16 \n\" ++
120
121      lyRelMarkNote.value(root, lastRoot, 2, "alto") ++ "\markup{\\large \\raise #2 \"II\"} +
122      "\stopGroup \\hide c'16 \n\" ++
123
124      lyHBracket.value(lyFreqRatio.value(root[2][4][2], nil, true, 0, false), 5.5, 1, 3, 0, 0) ++
125
126      lyRelMarkNote.value(root, lastRoot, 3, "treble") ++ "\markup{\\large \\raise #2 \"I\"} +
127      "\stopGroup \\stopGroup \n\" ++
128      "\hide c'16 \n\" ++
129      "}}}"
130  });
131 };

```

```

132 // barline and ossia definition
133
134 lyMeasureDef = {arg sectionData, insName, part, beat;
135   var ossia = "", barline = "|", break = "";
136   if(sectionData != nil, {
137     var root, lastRoot, section, subsection;
138     # root, lastRoot, section, subsection = sectionData;
139     ossia = lyRelMark.value(root, lastRoot, section, subsection);
140     barline = "\\bar \\|\\|";
141     if(sectionData[4], {barline = "\\bar \\|\\|\\|";});
142     if(sectionData[5], {barline = "\\bar \\|\\|\\|\\|";});
143   });
144   if(beat % 16 == 0, {break = "\\break \\noPageBreak"});
145   //for full score
146   //if(beat % (16 * 3) == 0, {break = "\\pageBreak"});
147   //for parts
148   if(beat % (16 * 8) == 0, {break = "\\pageBreak"});
149   if(beat != 0, {"\\n>>>\\n" + barline + break}, {""}) + "\\n<<<\\n" + ossia + ""
150 };
151
152 // add tie
153 lyTie = {"-"};
154
155 lyNoteName = {arg freq, spellingPref = \sharps;
156   if(freq != -1, {
157     lyNoteNameStr[spellingPref][((freq.cpsmidi).round(1) % 12)] ++
158     lyOctStr[(((freq.cpsmidi).round(1) / 12).asInteger - 2)];
159   }, {"x"});
160 };
161
162 lyCentDev = {arg freq, padding = true;
163   var centDev;
164   centDev = ((freq.cpsmidi - (freq.cpsmidi).round(1)) * 100).round(1).asInteger;
165   "\\markup { " ++ if(padding, {"\\pad-markup #0.2 \\"}, {""}) ++
166   if(centDev >= 0, {"+"}, {""}) ++ centDev.asString ++ "\\}";
167 };
168
169 lyFreqRatio = {arg freqRatioMult, ref, padding = true, lower = 3, attachedToNote = true;
170   var res, ratio;
171   res = "\\markup { " + if(attachedToNote, {""}, {"\\normal-size"}) +
172   "\\lower # " + lower + if(padding, {"\\pad-markup #0.2 "}, {" "});
173   ratio = "\\ " ++ freqRatioMult[0].product.asInteger ++ "/" ++ freqRatioMult[1].product.asInteger ++ "\\}";
174   res = if(ref != nil,
175     {
176       res ++ "\\concat{ \\ " ++ [nil, "III", "II", "I"][ref] ++ "\\normal-size-super " ++ ratio ++ " "
177     }, {
178       res ++ ratio
179     }
180   );
181   if(attachedToNote, {"." ++ res}, {res});
182 };
183
184
185 lyNote = {arg freq, noteLength, freqRatioMult, ref, spellingPref = \sharps, addMarkup = true, frHide = false, padding = true;
186   lyNoteName.value(freq, spellingPref) ++
187   lyDur.value(noteLength) ++
188   if(addMarkup, {
189     "<MARKUP" ++
190     lyCentDev.value(freq, padding) ++
191     if(frHide, {""}, {lyFreqRatio.value(freqRatioMult, ref, padding)}) ++
192     ">MARKUP>"
193   }, {""})
194 };
195
196 lyDur = {arg noteLength;
197   switch(noteLength, 1, {"16"}, 2, {"8"}, 3, {"8."}, 4, {"4"});
198 };
199
200 lyBeamOpen = {"["};
201
202 lyBeamClosed = {"]"};
203
204 consolidateNotes = {arg lyStr, part;
205   var noteRegex, markupRegex, fullNoteRegex, restRegex, fullRestRegex, res;
206   noteRegex = "(?<n>[a-g]{?es}is)?(?:[.']*?)24)";
207   markupRegex = if(part != 0, {"<MARKUP.{75,85}MARKUP>?"}, {"<MARKUP.{75,115}MARKUP>?"});
208   fullNoteRegex = noteRegex ++ markupRegex ++ "(?:\\h+\\|\\k<n>?";
209   restRegex = "(?<?>4)";
210   fullRestRegex = "(?<?>4) (?:\\h+\\|\\k<n>?";
211   res = lyStr;
212   [6, 4, 3, 2].do({arg len;
213     [fullNoteRegex, fullRestRegex].do({arg regex;
214       res.findRegexp(regex ++ "{ " ++ (len-1) ++ "}") .clump(3) .do({arg match;
215         var word, note, markup, lyDur;
216         word = match[0][1];
217         note = match[1][1];
218         markup = match[2][1];
219         lyDur = switch(len, 6, {"1."}, 4, {"1"}, 3, {"2."}, 2, {"2"});
220         res = res.replace(word, note.replace("4", lyDur) ++ markup);
221       });
222     });
223   res.replace("<MARKUP", "").replace(">MARKUP>", "");
224 };
225
226 ^transcribe = {arg rawMusicData, sectionData, seed;
227   var basePath, scoreFile, musicData, insData, insNames, insNamesShort, insMidi, insClef;
228
229   basePath = "dir +/" + "." + "/" + "lilypond" + "/" + "seed." ++ seed;
230   basePath.mkdir;
231   (basePath +/"includes").mkdir;
232
233   scoreFile = File(basePath +/"tkam_score.ly".standardizePath, "w");
234   scoreFile.write(File.readAllString(basePath +/".."/ +/"template"/ +/"tkam_score.template.ly").replace("seed: xxx", "seed: " ++ seed));
235   scoreFile.close;
236
237   musicData = formatMusicData.value(rawMusicData);
238
239   insData = [
240     ["*", "*", "clarinet", "\\treble_8\\|"],
241     ["III", "III", "clarinet", "bass"],
242     ["II", "II", "clarinet", "alto"],
243     ["I", "I", "clarinet", "treble"]
244   ];
245
246   insNames = insData.slice(nil, 0);
247   insNamesShort = insData.slice(nil, 1);
248   insMidi = insData.slice(nil, 2);
249   insClef = insData.slice(nil, 3);
250
251   musicData.do({arg part, p;
252     var lyFile, lyStr, lastMusAtom, measureCount, spellingPref,
253     tmpSectionData, pcRoot, partLookup, quality;
254
255     //create file
256     //for full score

```

```

257 //lyFile = File(basePath + "/" + "includes" + "/" + "part." + ["star", "III", "II", "I"][p] ++ ".ly".standardizePath, "w");
258 //for parts
259 lyFile = File(basePath + "/" + "includes" + "/" + "part." + ["star", "III", "II", "I"][p] ++ ".8systemsperpage.ly".standardizePath, "w");
260
261 //start lypond directives
262 lyStr = "";
263 lastMusAtom = nil;
264 measureCount = 0;
265 spellingPref = \sharps;
266 tmpSectionData = nil;
267 part.clump(4).do({arg beat, i;
268     var gSum;
269     gSum = 0;
270     beat.separate({arg a, b;
271         ((a[0] != -1) || (b[0] != -1)) && (a != b)}.do({arg group, g;
272             var noteLength, curMusAtom, freq, freqRatioMult, ref, isSame, isRest, isFirst, isLast,
273             isTied, isMeasureBound, isBeamStart, isBeamEnd;
274
275             noteLength = group.size;
276             gSum = gSum + noteLength;
277             curMusAtom = group[0];
278             freq = curMusAtom[0];
279             freqRatioMult = curMusAtom[1];
280             ref = curMusAtom[2];
281             # isSame, isRest, isFirst, isLast = [curMusAtom == lastMusAtom, freq == -1, g == 0, gSum == 4];
282             # isTied, isMeasureBound = [isSame && isRest.not, isFirst && ((i % 4) == 0)];
283             # isBeamStart, isBeamEnd = [(noteLength != 4) && isFirst, (noteLength != 4) && isLast];
284
285             //add ties
286             if(isTied, {lyStr = lyStr + lyTie.value});
287
288             //add barline and ossia definition
289             if(isMeasureBound, {lyStr = lyStr + lyMeasureDef.value(sectionData[i], insNames[p], p, i)});
290
291             //add note data
292             if(sectionData[i] != nil, {
293                 tmpSectionData = sectionData[i];
294             });
295             if(isTied.not, {
296                 partLookup = if((p != 0) || [1, 2, 3].includes(ref).not, {p}, {ref});
297                 pcRoot = ((tmpSectionData[0][partLookup][3].cpsmidi).round(1) % 12).asInteger;
298                 quality = if(tmpSectionData[0][partLookup][1][2] == [1, 5], [1, 2, 2], {\major}, {\minor});
299                 spellingPref = spellingDict[quality][pcRoot];
300                 if(p == 0, {(i / 4).asInteger, partLookup, pcRoot, quality});
301             });
302
303             lyStr = lyStr + lyNote.value(freq, noteLength, freqRatioMult, ref, spellingPref, isSame.not && isRest.not);
304
305             //beam group
306             if(isBeamStart, {lyStr = lyStr + lyBeamOpen.value});
307             if(isBeamEnd, {lyStr = lyStr + lyBeamClosed.value});
308
309             lastMusAtom = curMusAtom;
310         });
311     });
312
313     //wrap music and add staff definitions
314     lyStr = lyFinalizeMusic.value(lyStr, p, insNames[p], insNamesShort[p], insMidi[p], insClef[p]);
315
316     //consolidate notes and rests
317     lyStr = consolidateNotes.value(lyStr, p);
318
319     //write file
320     lyFile.write(lyStr);
321     lyFile.close;
322 };
323 };
324 };
325
326 //-----GENERATE SCORE DATA-----
327 ^genScoreData = {arg ensData;
328     var res;
329     res = ensData.collect({arg partData;
330         partData.flop.collect({arg data, d; if(d == 1, {data.differentiate ++ [10]}, {[0] ++ data})});
331     });
332     res.collect({arg part; part.flop});
333 };
334 )

```

## tkam\_gui.scd

```

1 (
2   var clockStringFunc, metronomeStringFunc, metronomeColorFunc, updateTransport, updateSection, updateSubsection,
3   buildGenerator, buildMetronome, buildTransport, buildTempoControl, buildMasterFader, buildTrackFader,
4   buildMasterView, buildFaderView, buildHelpView, currentSection = 1, currentSubsection = 1;
5
6   // these funcs update the elements of the transport panel
7   clockStringFunc = {
8     arg measure, beat;
9     var measureString, beatString, leadSpace;
10    measureString = measure.asInteger.asString;
11    beatString = beat.asInteger.asString;
12    leadSpace = (3 - measureString.size).collect({" "}).join;
13    leadSpace ++ measureString ++ "." ++ beatString
14  };
15
16  // [-30, -105, -104] and [-30, -105, -113] are unicode inverse bullet and normal bullet, respectively
17  metronomeStringFunc = { arg beat; if(beat == 1,
18    {[-30, -105, -104].collect({arg int; int.asAscii}).as(String)},
19    {[-30, -105, -113].collect({arg int; int.asAscii}).as(String)});
20  metronomeColorFunc = { arg beat; if(beat == 1, {Color.red}, {Color.black});
21
22  updateTransport = {arg clock, metronome, sectionDisplay, measure, beat, section, subsection;
23    sectionDisplay.string = "section: " ++ section.asInteger ++ "." ++ subsection.asInteger;
24    clock.string = clockStringFunc.value(measure, beat);
25    metronome.stringColor = metronomeColorFunc.value(beat);
26    metronome.string = metronomeStringFunc.value(beat);
27    {0.75.wait; {metronome.string = ""}.defer}.fork(~tempoClock, quant: 0);
28  }.inEnvir;
29
30  buildGenerator = {arg view;
31    var ranSeed;
32    HLayout (
33      ranSeed = TextField(view).string("19800725"),
34      Button(view).states(["reset seed"]).action({ ranSeed.string = "19800725"}.inEnvir),
35      Button(view).states(["random seed"]).action({ ranSeed.string = 50000000.rand.asString}.inEnvir),
36      Button(view).states(["generate"]).action({
37        ^genAll.value(ranSeed.string.asInteger); ^appStatus.string = "status: ready"}.fork(AppClock);
38        ^appStatus.string = "status: generating"}.inEnvir),
39      [^appStatus = StaticText(view).string("status: ready"), stretch: 1],
40      Button(view).states(["transcribe"]).action({
41        ^transcribe.value("scoreData", "sectionData", ranSeed.value); ^appStatus.string = "status: ready"}.fork(AppClock);
42        ^appStatus.string = "status: transcribing"}.inEnvir),

```

```

43     Button(view).states([[["bounce audio"]]).action({
44         {"bounceAudio.value(ranSeed.value); `appStatus.string = "status: ready".fork(AppClock);
45         `appStatus.string = "status: bouncing audio".inEnvir,
46         nil)
47     });
48
49 buildMetronome = {arg win;
50     var clock, metronome, layout;
51
52     clock = StaticText(win).string(" 1.1").font(Font("Liberation Mono", 200));
53     metronome = StaticText(win).string("[-30, -105, -104].collect({arg int; int.asAscii})
54         .as(String)).font(Font("Liberation Mono", 300)).stringColor(Color.red);
55
56     layout = HLayout(
57         clock,
58         StaticText(win).string("|").font(Font("Liberation Mono", 200)),
59         metronome
60     );
61
62     [clock, metronome, layout]
63 };
64
65 updateSection = {arg mod, clock, metronome, sectionDisplay, refresh = true, indirect = false;
66     var changeSection;
67     case
68     {currentSubsection > 1} && (mod < 0) {
69         currentSubsection = 1;
70     }
71     {currentSubsection <= 1} && (mod < 0) && (currentSection > 1) {
72         currentSection = currentSection + mod;
73         if(indirect, {
74             currentSubsection = `sectionNavDict[[currentSection, 1]][1]
75         }, {
76             currentSubsection = 1;
77         })
78     }
79     {mod > 0} && (!`sectionNavDict[[currentSection + mod, 1]] != nil) {
80         currentSection = currentSection + mod;
81         currentSubsection = 1;
82     };
83
84     if(refresh, {
85         updateTransport.value(clock, metronome, sectionDisplay,
86             `sectionNavDict[[currentSection, currentSubsection]][0], 1,
87             currentSection, currentSubsection
88         );
89     });
90 };
91
92 updateSubsection = {arg mod, clock, metronome, sectionDisplay, refresh = true;
93     if(`sectionNavDict[[currentSection, currentSubsection + mod]] != nil, {
94         currentSubsection = currentSubsection + mod;
95         if(refresh, {
96             updateTransport.value(clock, metronome, sectionDisplay,
97                 `sectionNavDict[[currentSection, currentSubsection]][0], 1,
98                 currentSection, currentSubsection
99             );
100         });
101     }, {
102         updateSection.value(mod, clock, metronome, sectionDisplay, refresh, true)
103     });
104 };
105
106 buildTransport = {arg win, view, clock, metronome, preampBusses, accompBusses, postampBusses;
107     var sec, subsec, sectionDisplay, layout, player;
108
109     sectionDisplay = StaticText(win).string("section: 1.1").font(Font("Liberation Mono", 70));
110
111     OSCFunc({ arg msg, time;
112         {
113             var measure, beat, section, subsection;
114             # measure, beat, section, subsection = msg[3..];
115             currentSection = sec = section.asInteger;
116             currentSubsection = subsec = subsection.asInteger;
117             updateTransport.value(clock, metronome, sectionDisplay, measure, beat, section, subsection);
118         }.inEnvir.defer;
119     }, 'measureClock.' ++ `hash, s.addr);
120
121     layout = HLayout(
122         Button(view).states([[["<<"], Color.black]]).action({arg pState; updateSection.value(-1, clock, metronome, sectionDisplay)}.inEnvir),
123         Button(view).states([[["<"], Color.black]]).action({arg pState; updateSubsection.value(-1, clock, metronome, sectionDisplay)}.inEnvir),
124         Button(view).states([[["play"], Color.black], [{"stop"}, Color.black, Color.grey]]).action({arg pState;
125             if(pState.value == 1, {
126                 player = {
127                     var startMeasure = `sectionNavDict[[currentSection, currentSubsection]][0] - 1;
128                     `patternProxy.source = `genPlayablePatterns.value(startMeasure, `patterns, preampBusses, accompBusses, postampBusses);
129                     Pbind(\instrument, \click ++ `hash, \beat, Pseq([1, 2, 1, 2]), \dur, 1).play(`tempoClock, quant: 0);
130                     [1, 2, 1, 2].do({arg beat;
131                         {
132                             metronome.stringColor = metronomeColorFunc.value(beat);
133                             metronome.string = metronomeStringFunc.value(beat);
134                         }.defer;
135                         0.75.wait;
136                         {metronome.string = ""}.defer;
137                         0.25.wait;
138                     });
139                     `patternProxy.play(`tempoClock, quant: 0)
140                 }.fork(`tempoClock, quant: 0)
141             }, {
142                 `patternProxy.pause;
143                 //player.stop;
144                 updateTransport.value(clock, metronome, sectionDisplay,
145                     `sectionNavDict[[currentSection, currentSubsection]][0], 1,
146                     currentSection, currentSubsection);
147             });
148         }.inEnvir,
149         Button(view).states([[[">"], Color.black]]).action({arg pState; updateSubsection.value(1, clock, metronome, sectionDisplay)}.inEnvir),
150         Button(view).states([[[">>"], Color.black]]).action({arg pState; updateSection.value(1, clock, metronome, sectionDisplay)}.inEnvir), nil,
151         sectionDisplay, nil);
152     [sectionDisplay, layout]
153 };
154
155 buildTempoControl = {arg view;
156     var layout, tempoField, address, updateSection;
157     layout = HLayout(
158         tempoField = TextField(view).string("60").action({arg v;
159             var tempo = v.value.asInteger; `tempoClock.tempo = tempo / 60}.inEnvir),
160         Button(view).states([[["set tempo"]]).action({arg v; `tempoClock.tempo = tempoField.string.asInteger / 60}.inEnvir),
161         [StaticText(view).string(" "), stretch: 1]);
162     [layout, tempoField]
163 };
164
165 buildMasterFader = {arg view;
166     var trackIndicators, layout, volSlider, muteButton, outMenu;
167

```



```

168 trackIndicators = {LevelIndicator()} ! 2;
169
170 OSCFunc.new({arg msg;
171   {trackIndicators[0].value = msg[3].ampdb.linlin(-50, 0, 0, 1)}.defer;
172   {trackIndicators[1].value = msg[4].ampdb.linlin(-50, 0, 0, 1)}.defer
173 }, '/masterLevels.' ++ `hash, s.addr);
174
175 layout = HLayout([
176   VLayout(
177     HLayout(
178       volSlider = Slider(view).value(0.8).action(
179         {arg v; var masterVol = v.value * 1.25; `play.set(\masterVol, masterVol)}.inEnvir),
180       trackIndicators[0],
181       trackIndicators[1]),
182       muteButton = Button(view).states([[["mute", Color.black], ["mute", Color.black, Color.grey]]].action(
183         {arg v; var masterMute = (1 - v.value).abs; `play.set(\masterMute, masterMute)}.inEnvir),
184       StaticText(view).string("out").align(\center),
185       outMenu = PopUpMenu(view).items((1..16).collect({arg o; o + "-" + (o + 1)})).action(
186         {arg v; var out = v.value.postln; `play.set(\masterOut, out)}.inEnvir),
187       StaticText(view).string("master").align(\center)
188     ), stretch: 2], nil);
189   [layout, volSlider, muteButton, outMenu]
190 ];
191
192 buildTrackFader = {arg view, name, index;
193   var trackIndicator, netAddr, layout, volSlider, soloButton, muteButton, panKnob, outMenu;
194
195   netAddr = NetAddr("127.0.0.1", NetAddr.langPort);
196   trackIndicator = LevelIndicator();
197
198   OSCFunc.new({arg msg; {trackIndicator.value = msg[3].ampdb.linlin(-50, 0, 0, 1)}.defer},
199     '/trackLevel.' ++ index ++ "-" ++ `hash, s.addr);
200
201   layout = HLayout(
202     VLayout(
203       HLayout(
204         volSlider = Slider(view).value(0.8).action(
205           {arg v; var vol = v.value * 1.25; `play.set(\vol. ++ index, vol)}.inEnvir),
206         trackIndicator),
207         soloButton = Button(view).states([[["solo", Color.black], ["solo", Color.black, Color.grey]]].action(
208           {netAddr.sendMsg("/soloer." ++ `hash, index)}.inEnvir).value(0),
209         muteButton = Button(view).states([[["mute", Color.black], ["mute", Color.black, Color.grey]]].action(
210           {arg v; var mute = (1 - v.value).abs;
211             `play.set(\mute. ++ index, mute)}.inEnvir).valueAction.(if(index < 4, {1}, {0})),
212         VLayout(
213           StaticText(view).string("pan").align(\center),
214           panKnob = Knob(view).action({arg v; var pan = v.value * 2 - 1; `play.set(\pan. ++ index, pan)}.inEnvir).valueAction.(0.5)
215         ),
216         StaticText(view).string("out").align(\center),
217         outMenu = PopUpMenu(view).items(["master"] ++ (1..16)).action(
218           {arg v; var out = v.value; `play.set(\out. ++ index, out)}.inEnvir).valueAction.(if(index < 6, {0}, {3})),
219         StaticText(view).string(name).align(\center)
220         //StaticText(view).string("output").align(\center),
221       ),
222       nil);
223     [layout, volSlider, soloButton, muteButton, panKnob, outMenu]
224 ];
225
226 buildMasterView = {arg win, preampBusses, accompBusses, postampBusses;
227   var view, generatorLayout, clock, metronome, metronomeLayout, transportLayout,
228     tempoControl, auxControlsLayout, countOffZ, ranSeed, order, tempo, sectionDisplay, address;
229
230   view = View(win);
231   generatorLayout = buildGenerator.value(view);
232   # clock, metronome, metronomeLayout = buildMetronome.value(win);
233   # sectionDisplay, transportLayout = buildTransport.value(win, view, clock, metronome, preampBusses, accompBusses, postampBusses);
234   tempoControl = buildTempoControl.value(view);
235   auxControlsLayout = tempoControl[0];
236
237   view.layout.(
238     HLayout(
239     [
240       VLayout(
241         metronomeLayout,
242         [StaticText(view).string(" "), stretch: 1],
243         transportLayout,
244         [StaticText(view).string(" "), stretch: 1],
245         auxControlsLayout,
246         [StaticText(view).string(" "), stretch: 1],
247         generatorLayout),
248         alignment: \top
249       ]
250     )
251   );
252   [view, tempoControl[1]]
253 ];
254
255 buildFaderView = {arg win, tempoField;
256   var view, masterIndicators, trackIndicators, master, tracks, openButton, basePath, saveButton;
257   var partAbbr = ["+", "III", "II", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I"];
258   var trackNames = ["+", "III", "II", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I", "I"];
259   var partVols, partMutes, partPans;
260   var masterMute, masterVol;
261   var netAddr = NetAddr("127.0.0.1", NetAddr.langPort);
262   var player = `play;
263
264   // set initial mixer values
265   partVols = [1, 1, 1, 1, 1, 1, 1];
266   partMutes = [0, 1, 1, 1, 1, 0];
267   partPans = [0, 0, 0, 0, 0, 0];
268   masterMute = 1;
269   masterVol = 1;
270
271   view = View(win);
272   masterIndicators = {LevelIndicator()} ! 2;
273   trackIndicators = {LevelIndicator()} ! 6;
274
275   master = buildMasterFader.value(view);
276   tracks = {arg part;
277     buildTrackFader.value(view, trackNames[part], part);
278   } ! 7;
279
280   OSCFunc.new({arg msg; {
281     tracks.slice(nil, 3).do({arg mute, m;
282       if(tracks[msg[1]][2].value == 1, {
283         mute.valueAction = if(msg[1] == m, {0}, {1});
284         tracks[m][2].value = if(msg[1] != m, {0}, {1})
285       }, {
286         mute.valueAction = 0
287       });
288     });
289   }, '/soloer.' ++ `hash, netAddr);
290
291   basePath = `dir ++ "/" ++ `.. ++ "/" ++ "mixer.settings";
292

```

```

293 openButton = Button(view).states(["open", Color.black]).action({
294   Dialog.openPanel({ arg path;
295     var settings;
296     settings = File.readAllString(path).parseJSON;
297     tempoField.valueAction = settings["tempo"];
298     master[1].valueAction = settings["master_volume"];
299     master[2].valueAction = settings["master_pan"];
300     master[3].valueAction = settings["master_out"];
301     settings["track_volumes"].do({arg val, v; tracks[v][1].valueAction = val});
302     settings["track_solos"].do({arg val, v; tracks[v][2].valueAction = val});
303     settings["track_mutes"].do({arg val, v; tracks[v][3].valueAction = val});
304     settings["track_pans"].do({arg val, v; tracks[v][4].valueAction = val});
305     settings["track_outs"].do({arg val, v; tracks[v][5].valueAction = val});
306   }, {}, false, basePath);
307 });
308
309 saveButton = Button(view).states(["save", Color.black]).action({
310   Dialog.savePanel({ arg path;
311     var settings, file;
312     settings = "{}\n";
313     settings = settings + "\"tempo\": " ++ tempoField.string ++ ",\n";
314     settings = settings + "\"master_volume\": " ++ master[1].value ++ ",\n";
315     settings = settings + "\"master_mute\": " ++ master[2].value ++ ",\n";
316     settings = settings + "\"master_out\": " ++ master[3].value ++ ",\n";
317     settings = settings + "\"track_volumes\": [" ++ tracks.collect({arg track; track[1].value}).join(",") ++ "],\n";
318     settings = settings + "\"track_solos\": [" ++ tracks.collect({arg track; track[2].value}).join(",") ++ "],\n";
319     settings = settings + "\"track_mutes\": [" ++ tracks.collect({arg track; track[3].value}).join(",") ++ "],\n";
320     settings = settings + "\"track_pans\": [" ++ tracks.collect({arg track; track[4].value}).join(",") ++ "],\n";
321     settings = settings + "\"track_outs\": [" ++ tracks.collect({arg track; track[5].value}).join(",") ++ "],\n";
322     settings = settings + "};\n";
323     file = File(path, "w");
324     file.write(settings);
325     file.close;
326   }, {}, basePath);
327 });
328
329 view.layout(HLayout(HLayout(master[0], nil, *tracks.slice(nil, 0)), VLayout(nil, saveButton, openButton)))
330 };
331
332 buildHelpView = {arg win;
333   TextView(win).string(File.readAllString("~/dir +/ tkam-readme.scd").editable(false);
334 };
335
336 ~generateGUI = {arg preampBusses, accompBusses, postampBusses;
337   var win, tabButtonReset, transportButton, mixerButton, helpButton, masterControl, tempoControl, masterView, faderView, helpView, tabs;
338   win = Window("to kill a monarch", Rect(500, 500, 1100, 575), false).front;
339   tabButtonReset = {transportButton.value = 1; mixerButton.value = 1; helpButton.value = 1};
340   masterControl = buildMasterView.value(win, preampBusses, accompBusses, postampBusses);
341   masterView = masterControl[0];
342   tempoControl = masterControl[1];
343   faderView = buildFaderView.value(win, tempoControl);
344   helpView = buildHelpView.value(win);
345
346   win.layout = VLayout(
347     HLayout(
348       HLayout(
349         [
350           transportButton = Button().states(["transport", Color.white, Color.grey], ["transport", Color.black]).action(
351             {tabButtonReset.value; transportButton.value = 0; tabs.index = 0 }.inEnvir).value(0), stretch: 1
352         ],
353         [
354           mixerButton = Button().states(["mixer", Color.white, Color.grey], ["mixer", Color.black]).action(
355             {tabButtonReset.value; mixerButton.value = 0; tabs.index = 1 }.inEnvir).value(1), stretch: 1
356         ]
357       ),
358       helpButton = Button().states(["help", Color.white, Color.grey], ["help", Color.black]).action(
359         {tabButtonReset.value; helpButton.value = 0; tabs.index = 2 }.inEnvir).value(1)
360     ),
361     tabs = StackLayout(masterView, faderView, helpView);
362 };
363 )

```

## tkam\_score\_template.ly

```

1 %\version "2.19.83"
2 \version "2.24.1"
3
4 # (define (override-color-for-all-grobs color)
5   (lambda (context)
6     (let loop ((x all-grob-descriptions))
7       (if (not (null? x))
8         (let ((grob-name (caar x)))
9           (ly:context-pushpop-property context grob-name 'color color)
10          (loop (cdr x))))))
11
12 # (define-markup-command (relMark layout props mus) (ly:music?)
13   #:properties ((size -2))
14   (interpret-markup layout props
15     # {
16       \markup {
17         \score {
18           \new Staff { $mus }
19           \layout {
20             \context {
21               \Staff
22               \remove Time_signature_engraver
23               fontSize = #-2
24               \hide Stem
25               \override TextScript.outside-staff-priority = ##f
26               \override StaffSymbol.staff-space = #(magstep -2)
27               \override StaffSymbol.thickness = #(magstep -2)
28               \override TextScript.self-alignment-X = #-0.4
29               \override TextScript.staff-padding = #1
30             }
31             \context {
32               \Score
33               proportionalNotationDuration = #(ly:make-moment 1/16)
34               \remove "Separating_line_group_engraver"
35               \override SpacingSpanner.strict-note-spacing = ##t
36               \override RehearsalMark.self-alignment-X = #-1
37               \override RehearsalMark.Y-offset = #10
38               \override RehearsalMark.X-offset = #10
39             }
40             \context {
41               \Voice
42               \consists "Horizontal_bracket_engraver"
43               \override HorizontalBracket.direction = #UP
44             }
45             indent = 0
46             line-width = 4\cm
47           }
48         }
49       }
50     #)})

```

```

51
52
53 \paper {
54   #(set-paper-size "a4" 'portrait)
55   top-margin = 1 \cm
56   bottom-margin = 1 \cm
57   left-margin = 2 \cm
58   ragged-bottom = ##t
59
60   top-system-spacing =
61   #'(basic-distance . 15 )
62   (minimum-distance . 15 )
63   (padding . 0 )
64   (stretchability . 0))
65
66   system-system-spacing =
67   #'(basic-distance . 35 )
68   (minimum-distance . 35 )
69   (padding . 0 )
70   (stretchability . 0))
71
72   last-bottom-spacing =
73   #'(basic-distance . 10 )
74   (minimum-distance . 10 )
75   (padding . 0 )
76   (stretchability . 0))
77
78   %systems-per-page = 3
79   first-page-number = 1
80   print-first-page-number = ##t
81
82   % for lilypond version 2.19.83
83   %print-page-number = ##t
84   %oddHeaderMarkup = \markup { \fill-line { \line { \on-the-fly #not-first-page {\pad-markup #2 { \concat {\italic {"to kill a monarch "}} (seed: xxx)}}}}}}
85   %evenHeaderMarkup = \markup { \fill-line { \line { \on-the-fly #not-first-page {\pad-markup #2 { \concat {\italic {"to kill a monarch "}} (seed: xxx)}}}}}}
86   %oddFooterMarkup = \markup { \fill-line {
87     \concat {
88       " "
89       \fontsize #1.5
90       \on-the-fly #print-page-number-check-first
91       \fromproperty #'page:page-number-string
92       " "}}}}
93   %evenFooterMarkup = \markup { \fill-line {
94     \concat {
95       " "
96       \fontsize #1.5
97       \on-the-fly #print-page-number-check-first
98       \fromproperty #'page:page-number-string
99       " "}}}}
100 }
101
102   % for lilypond version 2.24.1
103   print-page-number = ##t
104   oddHeaderMarkup = \markup { \fill-line { \line { \unless \on-first-page {\pad-markup #2 { \concat {\italic {"to kill a monarch "}} (seed: xxx)}}}}}}
105   evenHeaderMarkup = \markup { \fill-line { \line { \unless \on-first-page {\pad-markup #2 { \concat {\italic {"to kill a monarch "}} (seed: xxx)}}}}}}
106   oddFooterMarkup = \markup { \fill-line {
107     \concat {
108       " "
109       \fontsize #1.5
110       \fromproperty #'page:page-number-string
111       " "}}}}
112   evenFooterMarkup = \markup { \fill-line {
113     \concat {
114       " "
115       \fontsize #1.5
116       \fromproperty #'page:page-number-string
117       " "}}}}
118 }
119
120 \header {
121   title = \markup { \italic {to kill a monarch}}
122   composer = \markup \right-column {"michael winter" "(berlin, germany; 2021)"}
123   poet = "seed: 19800725"
124   tagline = ""
125 }
126
127 \header {
128   title = \markup { \italic {to kill a monarch}}
129   composer = \markup \right-column {"michael winter" "(berlin, germany; 2021)"}
130   poet = "seed: xxx"
131   tagline = ""
132 }
133
134 #(set-global-staff-size 11)
135
136 \layout {
137   indent = 0.0\cm
138   line-width = 17.5\cm
139   ragged-last = ##f
140   ragged-right = ##f
141
142   \context {
143     \Score
144     \override BarNumber.stencil = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
145     \override Stem.stemlet-length = #0.75
146     proportionalNotationDuration = #(ly:make-moment 1/16)
147     \remove "Separating-line-group-engraver"
148     \override RehearsalMark.self-alignment-X = #-1
149     \override RehearsalMark.Y-offset = #10
150     \override RehearsalMark.X-offset = #-8
151     %\override RehearsalMark.outside-staff-priority = #0
152     %added for new lilypond
153     rehearsalMarkFormatter = #format-mark-box-numbers
154   }
155   \context {
156     \Staff
157
158     \override VerticalAxisGroup.staff-staff-spacing =
159     #'(basic-distance . 20 )
160     (minimum-distance . 20 )
161     (padding . 0 )
162     (stretchability . 0))
163
164     \override VerticalAxisGroup.default-staff-staff-spacing =
165     #'(basic-distance . 20 )
166     (minimum-distance . 20 )
167     (padding . 0 )
168     (stretchability . 0))
169     \override TextScript.staff-padding = #2
170     \override TextScript.self-alignment-X = #0
171   }
172   \context {
173     \StaffGroup
174     \name "SemiStaffGroup"
175     \consists "Span.bar-engraver"

```

```

176 \override SpanBar.stencil =
177   #(lambda (grob)
178     (if (string=? (ly:grob-property grob 'glyph-name) "")
179       (set! (ly:grob-property grob 'glyph-name) "")
180       (ly:span-bar::print grob))
181   )
182 \context {
183   \Score
184   \accepts SemiStaffGroup
185 }
186 }
187 \midi { }
188
189
190 \score{
191 \new Score
192 <<
193   \new SemiStaffGroup {
194     <<
195       \include "includes/part.I.ly"
196       \include "includes/part.II.ly"
197       \include "includes/part.III.ly"
198     >>
199   }
200   \include "includes/part.star.ly"
201 >>
202
203 \layout{}
204 \midi{}
205 }
206

```