

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)

Approximately the first third of the piece is a sort of extended prelude. The rest of the piece 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. 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

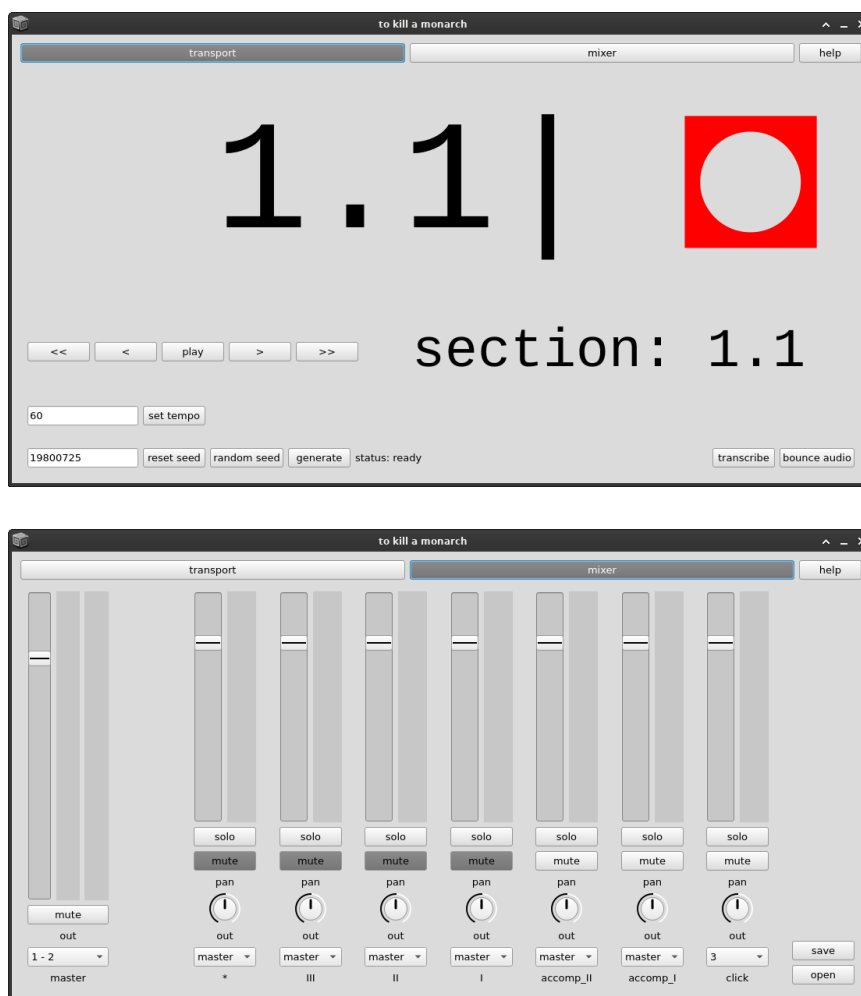
The score is written in a cut time with a tempo where the half note equals 60 beats per minute. The piece maybe be played at a slower tempo, but no less than the half note at 50 beats per minute.

## 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://gitea.unboundedpress.org/mwinter/to\\_kill\\_a\\_monarch](https://gitea.unboundedpress.org/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 written using SuperCollider version 3.11.2 and Lilypond version 2.22.0.



application user interface

*to kill a monarch*

seed: 19800725

michael winter  
(berlin, germany; 2021)

1.1

III 1/1 II 1/1 I

+0 +0 +0

1/1 1/1 1/1

I = 60

+41 +16 +4 +0

13/8 6/5 9/8 1/1

II

III

\* -31 +2 -31 +0

III 7/4 3/2 7/4 1/1

1.2

I +2 -31

3/2 7/4

II +2

3/2

III +4 -49 +2

11/8 9/8 3/2

\* +16 -49 -31 +41 +2 +4

6/5 11/8 7/4 13/8 3/2 9/8

I -49 +0

11/8 1/1

II +0 -49

1/1 11/8

III -31 +16 +4 +2 +0

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

\* +0 -49

III 1/1 11/8

-1-

**1.3**

Measure 13: I (+4, 9/8), II (+0, 1/1), III (+16, 6/5), \* (-31, III 7/4).  
 Measure 14: I (+2, 3/2), II (+0, 7/4), III (-31, 13/8), \* (-31, 11/8).  
 Measure 15: I (+0, 1/1), II (+2, 9/8), III (+41, 1/1), \* (+0, 1/1).  
 Measure 16: I (+2, 3/2), II (+41, 13/8), III (+2, 3/2), \* (+0, 1/1).

**1.4**

Measure 17: I (+41, 13/8), II (+4, 9/8), III (-49, 11/8), \* (+0, 1/1).  
 Measure 18: I (+41, 13/8), II (+4, 9/8), III (-49, 11/8), \* (+0, 1/1).  
 Measure 19: I (+41, 13/8), II (+4, 9/8), III (-49, 11/8), \* (+0, 1/1).  
 Measure 20: I (+41, 13/8), II (+4, 9/8), III (-49, 11/8), \* (+0, 1/1).

**1.5**

Measure 21: I (-49, 11/8), II (+2, 3/2), III (-31, 7/4), \* (+41, 13/8).  
 Measure 22: I (+16, 6/5), II (-49, 11/8), III (+41, 13/8), \* (+2, 3/2).  
 Measure 23: I (-31, 7/4), II (+0, 1/1), III (+4, 9/8), \* (+0, 1/1).  
 Measure 24: I (+2, 3/2), II (-31, 7/4), III (+16, 6/5), \* (-31, 7/4).

(25)

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

(29) 1.6

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

(33) 1.7

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

37

Staff I: Measures 37-40. Time signatures: 3/2, 7/4, 1/1. Accidentals: +2, +0, -31.

Staff II: Measures 37-40. Time signatures: 9/8, 7/4. Accidentals: +4, -31.

Staff III: Measures 37-40. Time signatures: 3/2, 1/1, 11/8. Accidentals: +2, +0, -49.

Staff \*: Measures 37-40. Time signatures: 9/8, 13/8. Accidentals: +4, +41.

41

1.8

Staff I: Measures 41-44. Time signatures: 3/2, 11/8, 3/2. Accidentals: +2, -49, +2.

Staff II: Measures 41-44. Time signatures: 7/4. Accidental: -31.

Staff III: Measures 41-44. Time signatures: 3/2, 7/4, 9/8. Accidentals: +2, -31, +4.

45

Staff I: Measures 45-48. Time signatures: 11/8, 6/5, 13/8, 1/1, 3/2, 7/4, 9/8, 3/2, 11/8. Accidentals: -49, +16, +41, +0, +2, -31, +4, +2, -49.

Staff II: Measures 45-48. Time signatures: 1/1, 13/8. Accidentals: +0, +41.

Staff III: Measures 45-48. Time signatures: 3/2, 13/8, 6/5. Accidentals: +2, +41, +16.

Staff \*: Measures 45-48. Time signatures: 6/5, 11/8, 3/2, 6/5. Accidentals: +16, -49, +2, +16.

**1.9**

49

I

II

III

\*

3/2

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

+2

-31 +4 -49 +2 +0 +16 +41 -49

+0

1/1

-31 +2 +0

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

**1.10**

53

I

II

III

\*

6/5 13/8 1/1

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

11/8 9/8 7/4

9/8

+16 +41 +0

+0 -31 +2 +4

-49 +4 -31

+4

-31 +0 +2

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

+0 +2

1/1 3/2

III 3/2

**1.11**

57

I

II

III

\*

7/4 11/8 9/8

7/4 6/5 13/8 7/4 11/8 1/1 1/1 3/2 7/4

7/4 1/1

-31 -49 +4

-31 +16 +41 -31 -49 +0 +0 +2

-31 +16

7/4 6/5

+4 +0

9/8 1/1



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, +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, 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

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

III: +16, 6/5

\*: +16, 6/5

77 1.15

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

III: +2, 3/2

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

81

II: -31, 7/4

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

\*: -31, III 7/4, 11/8, -49, 11/8, +2, 3/2

85 **1.16**

I

II

III

\*

-49  
11/8

-31  
7/4

+41  
13/8

-49  
11/8

+0  
1/1

+2  
3/2

89

I

II

III

\*

+0  
1/1

+4  
9/8

+41  
13/8

+0  
1/1

+41  
13/8

+0  
1/1

-31  
7/4

+16  
6/5

-49  
11/8

+41  
13/8

-31  
7/4

**1.17**

93

I

II

III

\*

+2  
3/2

+0  
1/1

+41  
13/8

+4  
9/8

-49  
11/8

-31  
7/4

+16  
6/5

+2  
3/2

+16  
6/5

-49  
11/8

-31  
7/4

+16  
6/5

+41  
13/8

+0  
1/1

+4  
3/2

+2  
3/2

97 1.18

I  $\frac{13}{8}$

II  $\frac{1}{1}$   $\frac{3}{2}$   $\frac{9}{8}$

III

\*

$\frac{13}{8}$   $\frac{1}{1}$   $\frac{9}{8}$

13/8 1/1 9/8

101 1.19

I  $\frac{1}{1}$   $\frac{7}{4}$   $\frac{3}{2}$

II

III  $\frac{3}{2}$   $\frac{11}{8}$   $\frac{9}{8}$   $\frac{13}{8}$   $\frac{7}{4}$   $\frac{1}{1}$

\*

$\frac{6}{5}$   $\frac{9}{8}$   $\frac{1}{1}$   $\frac{7}{4}$   $\frac{11}{8}$

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

105

I  $\frac{9}{8}$   $\frac{6}{5}$   $\frac{1}{1}$   $\frac{3}{2}$   $\frac{7}{4}$

II  $\frac{7}{4}$   $\frac{13}{8}$

III  $\frac{6}{5}$   $\frac{11}{8}$   $\frac{9}{8}$

\*

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

11/8 3/2 7/4

113

I

II

III

\*

1.21

+16 +0 +2 -49 +41 +16 +2 +0 +4 +0 -31

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

+41

13/8

+41 +0

13/8 1/1

+4 +41 +16 +2 -31 -49

9/8 13/8 6/5 3/2 7/4 11/8

117 -49 +16

I 11/8 6/5

II +0 1/1

III -31 7/4 +16 6/5

\* +2 +0 +16 +2 +4 11/8 3/2 9/8

1.22

*to kill a monarch* (seed:19800725)

The image displays a musical score for the song "The Rose Tree" in G major. It consists of four staves, labeled I, II, III, and a solo part marked with an asterisk (\*). The key signature is one sharp (F#), indicating G major. The time signature is 3/8. The score includes various musical notations such as notes, rests, and accidentals. Fret numbers are indicated above the notes, and interval markings (e.g., 3/2, 1/1, 6/5, 11/8) are placed below the notes. The solo part is marked with an asterisk and includes a 7/4 time signature change.

125

1.23

I

II

III

\*

7/4

III<sup>3/2</sup>

The image displays a musical score for the song "The Rose Tree". It consists of four staves, labeled I, II, III, and \*. Staff I is in treble clef with a key signature of one flat (B-flat). Staff II is in alto clef with a key signature of one flat. Staff III is in bass clef with a key signature of one flat. Staff \* is in treble clef with a key signature of one flat. The score includes various musical notations such as notes, rests, and accidentals. Time signatures are indicated below the staves: 11/8, 9/8, 1/1, 13/8, 3/2, 7/4, 13/8, 1/1, 7/4, 9/8, 11/8, 9/8, 13/8, 3/2, 11/8, 6/5, 7/4, and 9/8. A box labeled "1.24" is positioned above the first staff. A circled number "129" is located at the top left of the first staff. A circled number "8" is located at the bottom left of the fourth staff. The score is divided into two measures by a double bar line. The first measure contains the main melody and accompaniment, while the second measure contains a repeat of the melody and accompaniment.

133

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

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

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

Staff \*: +2 +0 +41 +0 -49 +16  
III 3/2 1/1 13/8 1/1 11/8 6/5

1.25

137

Staff I: +41 +2  
13/8 3/2

Staff II: +41  
13/8

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

Staff \*: -31  
III 7/4

141

1.26

Staff I: +16 -31 +4 +2  
6/5 7/4 9/8 1/1 3/2

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

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

145

Staff I: Treble clef, key of B-flat major. Measures 145-148.

Staff II: Bass clef, key of B-flat major. Measures 145-148. Interval markings: +4, +16, +2.

Staff III: Bass clef, key of B-flat major. Measures 145-148. Interval markings: +0, +41, +2, +4, +16, +0, -49, +2, +41.

Staff \*: Treble clef, key of B-flat major. Measures 145-148. Interval markings: +0, +4, +2, -31.

Measure 145: 1/1, 3/2, 13/8, 9/8, 6/5, 3/2, 1/1, 11/8, 13/8, 1/1, 9/8, 3/2, 7/4.

1.27

149

Staff I: Treble clef, key of B-flat major. Measures 149-152. Interval markings: -49, +41, +16.

Staff II: Bass clef, key of B-flat major. Measures 149-152. Interval markings: -49, +0, +2.

Staff III: Bass clef, key of B-flat major. Measures 149-152. Interval markings: +0.

Staff \*: Treble clef, key of B-flat major. Measures 149-152. Interval markings: +0, +4, +16.

Measure 149: 11/8, 13/8, 6/5, 11/8, 1/1, 3/2, 1/1, 9/8, 6/5, 11/8, 1/1, 3/2, 1/1.

1.28

153

Staff I: Treble clef, key of B-flat major. Measures 153-156. Interval markings: +16, +2, +41.

Staff II: Bass clef, key of B-flat major. Measures 153-156. Interval markings: -31, +2, +41.

Staff III: Bass clef, key of B-flat major. Measures 153-156. Interval markings: +2, +4, +0, -31.

Staff \*: Treble clef, key of B-flat major. Measures 153-156. Interval markings: -49, +41, +0, -31, +0.

Measure 153: 7/4, 6/5, 3/2, 13/8, 3/2, 9/8, 1/1, 7/4, 11/8, 13/8, 1/1, 7/4, 1/1.



157

1.29

Staff I:  $+0$   $+4$   $-49$   $+16$   $+41$   $-31$   $+2$   $-49$   $+0$   $-49$   $+4$   $+2$

Staff II:  $+4$   $+0$   $+2$   $-31$   $+16$

Staff III:  $+16$   $-49$   $+0$

Staff \*:  $+4$   $+16$   $+4$

Time signatures:  $1/1$   $9/8$   $6/5$   $13/8$   $7/4$   $3/2$   $11/8$   $1/1$   $9/8$   $3/2$

Other time signatures:  $9/8$   $1/1$   $3/2$   $7/4$   $6/5$   $1/1$   $6/5$   $11/8$   $1/1$   $9/8$   $6/5$   $III^{9/8}$

161

Staff I:  $+41$   $+2$   $-49$   $+0$

Staff II:  $+41$   $+4$   $-49$   $+0$

Staff III:  $+41$   $+4$

Staff \*:  $+2$   $+41$   $-49$   $+16$   $+0$

Time signatures:  $13/8$   $9/8$   $3/2$   $11/8$   $1/1$   $13/8$   $9/8$   $3/2$   $11/8$   $13/8$   $6/5$   $1/1$

1.30

165

Staff I:  $+16$

Staff II:  $+16$

Staff III:  $+16$

Staff \*:  $-31$   $+2$   $+4$

Time signatures:  $6/5$   $III^{7/4}$   $3/2$   $9/8$

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 -49 +16 (3/2 1/1 13/8 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 +2 +41 -31 +16 +2 +16 (1/1 11/8 3/2 13/8 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 7/4 1/1 13/8 9/8 11/8 1/1 3/2 1/1)

Staff \*: -31 -49 +4 +41 +2 (III 11/8 7/4 9/8 13/8 3/2)

177 1.32

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

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

**1.33**

181

Interval markings for system 1.33:

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

**1.34**

185

Interval markings for system 1.34:

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

**1.35**

189

Interval markings for system 1.35:

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

193 1.36

I  $\frac{7}{4}$   $\frac{6}{5}$   $\frac{11}{8}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{6}{5}$   $\frac{11}{8}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{3}{2}$   $\frac{1}{1}$

II  $\frac{13}{8}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{1}{1}$   $\frac{6}{5}$

III  $\frac{3}{2}$   $\frac{6}{5}$   $\frac{11}{8}$   $\frac{7}{4}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{1}{1}$   $\frac{6}{5}$   $\frac{9}{8}$   $\frac{6}{5}$   $\frac{13}{8}$   $\frac{3}{2}$

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

197

I  $\frac{7}{4}$   $\frac{9}{8}$   $\frac{11}{8}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{3}{2}$   $\frac{6}{5}$

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

III  $\frac{7}{4}$   $\frac{6}{5}$   $\frac{1}{1}$   $\frac{9}{8}$

\*  $\text{III}^{\frac{3}{2}}$

201 1.37

I  $\frac{7}{4}$   $\frac{6}{5}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{11}{8}$

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

III  $\frac{11}{8}$   $\frac{1}{1}$

\*  $\text{III}^{\frac{1}{1}}$

205 1.38

I:  $3/2$   $9/8$   $1/1$   $7/4$   $1/1$

II:  $7/4$   $6/5$   $13/8$   $3/2$   $7/4$   $9/8$   $11/8$   $3/2$

III:  $7/4$   $3/2$   $11/8$   $1/1$   $3/2$   $6/5$

\*:  $13/8$   $3/2$   $11/8$   $9/8$   $1/1$

209

I:  $6/5$   $13/8$   $11/8$   $3/2$   $1/1$   $9/8$

II:  $1/1$   $11/8$   $6/5$

III:  $13/8$   $1/1$   $9/8$   $11/8$   $9/8$   $6/5$

\*:  $III^{6/5}$   $7/4$

213 1.39

I:  $3/2$

II:  $3/2$

III:  $3/2$

\*:  $3/2$   $1/1$   $III^{3/2}$

217 1.40

Staff I: +2, 3/2

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

Staff III: +4, 9/8, -31, 7/4, +0, 1/1, -49, 11/8

Staff \*: +16, 6/5

221

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

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

Staff III: +41, +0, +2, 13/8, 1/1, 3/2

Staff \*: -49, 11/8

225 1.41

Staff I: +16, 6/5

Staff II: -49, 11/8

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

Staff \*: +2, 3/2, 6/5

229 1.42

Staff I: -31, +0, +4, +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, +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 4/3 I

+0 +4 +0 +0 +2

9/8 1/1 3/2

(241)

I

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

II

13/8

III

7/4 11/8 13/8

\*

+0 +4 +8

II<sup>1/1</sup> I<sup>3/2</sup> III<sup>9/8</sup>

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

**2.2**

III 4/3 II 9/8 I

+4 +0 +2 +2 +0

1/1 3/2 3/2

(245)

I

6/5 13/8 3/2

7/4 6/5 13/8

II

III

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

\*

II<sup>11/8</sup> I<sup>13/8</sup>

+18 +42 +4 -31 +16

+41

**2.3**

III 3/2 II 4/3 I

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

4/3 3/2 9/8

(249)

I

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

3/2

II

1/1

III

3/2

\*

II<sup>9/8</sup> I<sup>1/1</sup> II<sup>3/2</sup> III<sup>9/8</sup> II<sup>11/8</sup>

+2 +4 -49 -31 +0 +6

+8 -49



**2.4**

III  $\frac{3}{2}$  II  $\frac{1}{1}$  I

+2 +0 +4 +2

1/1 1/1 4/3

(253)

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

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

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

III +18 +6 +42

6/5 9/8 13/8

\* +6

III  $\frac{9}{8}$

(257)

I -47 +4

11/8 3/2

II +16 +2

6/5 3/2

III -29 +4 +2

7/4 3/2 1/1

\* +18 -47

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

**2.5**

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

+2 +0 +2 -49

3/2 3/2 12/11

(261)

I -49 -33 -45 -49

1/1 6/5 9/8 1/1

II

III

\* -47 +4 -29 +6

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

265

2.6

III 1/1 II 1/1 I

+0 -49 +2 -49 -49

11/8 12/11 1/1

I

-8 -49 -45

3/2 6/5 9/8

13/8 1/1

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

6/5 11/8

II

-49 -45 -33 +3 -49 -8 +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 -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

\*

-8 -33 -47 +3 -8 -45 +20 -47

6/5 13/8 3/2 11/8 9/8 7/4 3/2

**3.1**

III 13/12 II 4/3 I

-49 -49 -8 -49 -47

1/1 16/13 4/3

(277)

I

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

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

II

+43

11/8

III

-33 -47

6/5 3/2

\*

**3.2**

III 13/12 II 1/1 I

-49 -47 -8 -47

4/3 1/1 1/1

(281)

I

-6

13/8

II

-8

1/1

III

-8 +20

13/8 7/4

\*

-8 +20 +40 +5

III 13/8 7/4 I 5/4 11/8

(285)

I

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

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

II

-6 -8 -39

3/2 1/1 7/4

III

-47 +22

1/1 7/4

\*

-6 +32 +40 +40 +5 -45

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

-24-

**3.3**

III  $\frac{13}{12}$  II  $\frac{13}{12}$  I

-47 -8 -47 -47 -8

1/1 13/12 13/12

(289)

I  $\frac{11}{8}$

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

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

III +5 11/8

\* -6 +22 +32 -4

III  $\frac{13}{8}$  7/4 I  $\frac{13}{8}$  9/8

**3.4**

III  $\frac{13}{12}$  II  $\frac{1}{1}$  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

II

III -6 +40 -43 -45

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

\* +5 -45 -6

III  $\frac{11}{8}$  3/2 III  $\frac{13}{8}$

(297)

I -45 3/2

II -8 +32 -22 -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  $\frac{13}{8}$  III  $\frac{5}{4}$  5/4

**3.5**

III 13/12 1/1 II 13/12 I

-47 -8 -47 -47 -8

13/12 13/12 13/12

301

I

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

II

3/2 5/4 3/2

III

7/4 1/1

\*

15/4 III 11/8 3/2

+32 -22 -6 -4 -8 -39 +43 +32 -22

+40 -45 -45

+22 -47

-22 +5 -45

**3.6**

III 13/12 1/1 II 13/12 I

-47 -8 -47 -8 -47

13/12 1/1 13/12

305

I

II

III

5/4

\*

13/8 I 13/8 III 5/4 I 5/4

+40 -6 +32 +40 -22

309

I

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

II

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

III

13/8 3/2

\*

II 11/8 3/2

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

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

+32 -6

+5 -45

**3.7**

III 1/1 II 1/1 I

-8 -47 -47 -47

13/12 1/1 1/1

313

I

II

III

\*

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

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

5/4

III 9/8

317

I

II

III

\*

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

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

-47 1/1 +40 5/4 +22 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 -47 -6 -45 +22 -6 -47

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

**4.1**

III 5/4 II 1/1 I

-47 -47 +40 -47

1/1 5/4 1/1

321

I

II

III

\*

+40 -47 +40

5/4 1/1 5/4

+22 7/4

+5 11/8

4.2

III 13/10 II 5/4 I

-47 +40 +40 -6 -47

5/4 13/10 1/1

325

I +5 11/8

II +40 -20 1/1 13/8

III -6 +5 13/8 11/8

\* -6 +22 +8 7/4 7/4

III 13/8 II 7/4

329

I +22 -47 -43 7/4 1/1 9/8

II -4 +9 3/2 6/5 +45 11/8

III +42 +26 +40 -20 +43 +42 +8 +8 5/4 3/2 1/1 13/8 9/8 7/4 3/2 7/4

\* +5 -45 +26 -6 +22 +9 11/8 3/2 III 5/4 I 13/8 7/4 II 6/5

4.3

III 5/4 II 13/10 I

+40 -6 -47 -47 -6

1/1 13/8 13/8

333

I -4 3/2

II -43 -47 9/8 1/1

III

\* +8 III 7/4

337

Staff I: 9/8, 13/8, 6/5

Staff II: 3/2, 7/4, 1/1

Staff III: 5/4, 1/1, 7/4

Staff \*: 1 3/2, 13/8

4.4

Staff I: 13/8, 5/4, 13/10

Staff II: 11/8, 7/4, 1/1, 13/8, 3/2, 1/1

Staff III: 13/8, 3/2, 7/4

Staff \*: II 7/4, III 7/4

4.5

Staff I: 13/8, 1/1, 13/10

Staff II: 7/4

Staff III: 1/1

Staff \*: III 11/8, 3/2, 1/1, 3/2, 6/5, 9/8, 13/8, 7/4, 6/5



349

I

II

III

\*

5.1

14/11

8/7

16/11

1/1

8/7

16/11

-6

-37

-6

+45

353

I

II

III

\*

III<sup>6/5</sup>

357

I

II

III

\*

**5.2**

III 8/7 II 14/11 I

-6 -37 -37 -6 +45

8/7 8/7 1/1

(361)

I

11/8 5/4 9/8

-4 +31

II

9/8 7/4 11/8

-2 -37 +45

III

7/4 3/2 1/1 13/8 11/8 6/5 7/4 1/1

-37 +3 -33 +31 -22 -37

\*

11/8 9/8

-4 -2

**5.3**

III 8/7 II 8/7 I

-37 -6 -6 -37 +45 -37

8/7 8/7 14/11

(365)

I

1/1

II

1/1 6/5

-37 -22

III

9/8

-2

\*

3/2 13/8 7/4

-35 +3 +31

(369)

I

6/5 13/8 7/4 11/8 3/2 1/1 7/4 11/8 13/8

-22 +3 +31 -35 -37 +31 +14

II

III

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

-37 -4 -37 -2 -4

\*

9/8 9/8

-33 -2

**5.4**

III 1/1 II 1/1 I

-6 -37 -37 -37

8/7 1/1 1/1

(373)

I

II

III

\*

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

13/8 1/1 9/8 6/5 1/1

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

+14 -35 -33 +31 +3 -37 -22 +14 -33

+3 -37 -33 -22 -37

-35 -37 +31 +31 -33

(377)

I

II

III

\*

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

3/2 13/8

7/4

1/1 13/8 6/5 11/8 9/8 7/4 13/8

-33 +31 -37 -22 +14 -33 +3 -37 -35 -22

-35 +3

-37 +3 -22 +14 -33 +31

**6.1**

III 4/3 II 4/3 I

-37 -35 -37 -37

4/3 1/1 1/1

(381)

I

II

III

\*

1/1 11/8

6/5

11/8

+16

+14

-22

-37

6.2

III 8/5 II 4/3 I 6/5

-35 -22 -37 -35

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 I 9/8 6/5 11/8

6.3

III 6/5 II 4/3 I 8/5

-22 -35 -37 -37 -35

1/1 4/3 4/3

393

I

-35

1/1

II

-22 -37 -35 +14 +3 -22 +31

1/1 3/2 11/8 13/8 6/5 7/4

III

-22 +30 -35 +19

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

\*

+3 -32

13/8 I 9/8

-33-

**6.4**

III 6/5 II 8/5 I  
-22 -37 -35  
1/1 1/1 1/1

(397)

I: +5 -35  
13/8 1/1

II: -33 -37 -35  
1/1 9/8 3/2 7/4

III: -18 -22 -30 -35  
7/4 9/8 1/1 5/4

\*: +47 -33  
III 7/4 II 9/8

**6.5**

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

(401)

I: +33 +16 -35 -32 +5  
7/4 11/8 1/1 9/8 13/8

II: +14 +3 -33  
11/8 13/8 9/8

III: -18  
9/8

\*: +5 +14 +14 +30 -49  
I 13/8 II 11/8 11/8 III 11/8 III 5/4

**6.6**

III 16/11 II 12/11 I  
-35 +14 -35 -37 -37 -35

(405)

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

II: -49 +16 +5 -33  
5/4 11/8 13/8 3/2

III: -33 -32 +5 +16 -35 -33 -35 -49  
3/2 9/8 13/8 11/8 7/4 5/4 1/1

\*: +16 -33  
11/8 3/2

409

6.7

1/1

III 12/11 II 12/11 I

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

12/11 16/11 12/11

I

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

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

II

-22

6/5

III

-17 +16 +14

7/4 3/2 1/1

\*

+33 -17 +14 +0 -49

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

II 11/8

413

I

+16 +0 +18 +14 +16

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

II

+0 -46 +14 +18

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

III

-32 +33 +16 +5 +33 -32 -49 -32 +16

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

\*

-35 -46 -17 +33 +18 +18 +0

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

417

6.8

1/1

III 1/1 II 1/1 I

-35 +14 +14 +14

12/11 1/1 1/1

I

-17

7/4

II

-46 +16 +0 -35

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

3/2 5/4

III

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

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

-35-

421

I  $\frac{11}{8}$   $\frac{5}{4}$   $\frac{13}{8}$   $\frac{7}{4}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{1}{1}$

II  $\frac{1}{1}$   $\frac{11}{8}$   $\frac{5}{4}$   $\frac{3}{2}$   $\frac{9}{8}$   $\frac{1}{1}$   $\frac{9}{8}$

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

\*  $\frac{11}{8}$

Intervals: -35, +0, -46, -17, +18, +16, -17, +14

Intervals: +16, +18, +14, -35, +0, +14, +18

Intervals: +14, +18, -17, +16, +14, -35

Interval: -35

7.1

III  $\frac{3}{2}$  II  $\frac{1}{1}$  I

III  $\frac{1}{1}$   $\frac{3}{2}$   $\frac{1}{1}$

425

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

II  $\frac{11}{8}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{5}{4}$   $\frac{7}{4}$

III  $\frac{13}{8}$   $\frac{3}{2}$   $\frac{5}{4}$   $\frac{9}{8}$   $\frac{1}{1}$   $\frac{5}{4}$

\*  $\frac{13}{8}$   $\frac{7}{4}$

Intervals: -35, +16, -33, +16, -15, +2, -46, +16, +0, +18, +0, -46, -17

7.2

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

III  $\frac{1}{1}$   $\frac{13}{12}$   $\frac{3}{2}$

429

I  $\frac{5}{4}$

II

III  $\frac{7}{4}$

\*  $\frac{11}{8}$   $\frac{11}{8}$

Intervals: +2, -17, +6, -33

433

1/1

-44

3/2

-46

1/1

-35 +0 +16 +14 +18

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

+0 -35 -46

III 5/4 11/8 13/8

7.3

13/8

III 13/12 II 3/2 I

+14 +16 -46 +16 +14

3/2 1/1 3/2

437

13/8

-5

11/8

-33

11/8

+2

III 5/4

441

+18 -46

9/8 13/8

-42 +23 -46 -30 +6

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

+16

1/1

7.4

11/8

III 3/2 II 12/11 I

+16 -46 +14 +14 -35

1/1 13/8 11/8



445

-35

1/1

+16 +14 +18 +14 +16 +14 -35 +18 -35

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

+18 +2

3/2 5/4

-33

III 11/8

7.5

12/11 1/1 12/11

III 1/1 II 12/11 I

+16 -35 +14 -35 -35 +16

12/11 11/8 12/11

449

+34

7/4

-33 +2 +20

5/4 11/8 9/8

+6

13/8

-31 +34

7/4 9/8

-33

11/8

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

453

+18 +16

3/2 1/1

-48 -31 -35 -33 -31

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

-44 -15 -31 -48

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

457

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

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

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

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

8.1 16/13 11/8 22/13 1/1 16/13

461

11/8 16/13

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

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

465

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

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

1/1 1/1

13/8 11/8 3/2 3/2

8.2

22/13

III 16/13 II 11/8 I

+17 -35 -35 +6 +6 +17

11/8 16/13 22/13

469

I

-43

11/8

13/8 7/4 1/1

-31 -33

9/8 3/2 9/8

II

III

\* +34

7/4

473

I

+19

3/2

+6 -43 +6 -43 +8

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

-33

3/2

+6

13/8

-48 -33

5/4 3/2

III

\* +19

1 3/2

-43

II 11/8

8.3

16/13

1/1 16/13

III 1/1 II 11/8 I

-35 +6 +6 +17 -35

16/13 1/1 11/8

477

I

+6 -33 +6

13/8 3/2 13/8

+46

13/8

+6 -25

1/1 7/4

II

III

+6

1/1

+46

13/8

-25

7/4

\* III 13/8

**8.4**

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

+6 +6 -35 +6

1/1 1/1 16/13

(481)

I  $\frac{1}{1}$   $\frac{11}{8}$   $\frac{6}{5}$   $\frac{7}{4}$   $\frac{3}{2}$

II

III +21  $\frac{6}{5}$  +10  $\frac{9}{8}$  +6 +46 -43  $\frac{13}{8}$   $\frac{11}{8}$   $\frac{1}{1}$

\*  $\frac{1}{3/2}$   $\frac{1}{1}$

(485)

I -43  $\frac{11}{8}$  +10  $\frac{9}{8}$

II +21  $\frac{6}{5}$  -43  $\frac{11}{8}$  +10  $\frac{1}{1}$  +6 +8 -25  $\frac{3}{2}$   $\frac{7}{4}$   $\frac{1}{1}$

III +8  $\frac{3}{2}$  +10  $\frac{9}{8}$  -25  $\frac{11}{8}$  +21  $\frac{6}{5}$  +6  $\frac{1}{1}$  +46  $\frac{13}{8}$   $\frac{3}{2}$

\* -43  $\frac{11}{8}$  -25  $\frac{7}{4}$  +8  $\frac{3}{2}$  +21  $\frac{6}{5}$

**9.1**

III  $\frac{3}{2}$  II  $\frac{1}{1}$  I

+6 +6 +8 +6

1/1 3/2 1/1

(489)

I  $\frac{13}{8}$   $\frac{1}{1}$

II +21 -43 -25  $\frac{6}{5}$  +10  $\frac{9}{8}$  +21  $\frac{6}{5}$  +46  $\frac{13}{8}$

III

\* +46  $\frac{13}{8}$  +10  $\frac{9}{8}$

493

I

II

III

\*

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

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

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

III 9/8 6/5

505

9.4

III 1/1 II 1/1 I

+10 +8 +10 +6 +10

1/1 4/3 9/8

I

-25 +8 -43

7/4 3/2 11/8

II

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

6/5 7/4 9/8 11/8 1/1 3/2 9/8

III

+10 +12 -21

1/1 3/2 7/4

\*

-39 +46

11/8 13/8

509

I

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

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

II

-39 +10 +12 +14

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

III

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

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

\*

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

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

513

10.1

III 3/2 II 11/8 I

+10 +10 +12 +10 -39

1/1 3/2 11/8

I

-4

5/4 3/2

II

+10 +23

1/1 8/5

III

+10

1/1

\*

-4 +10 +12 -39 -35

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

-43-

10.2

III 3/2 II 12/11 I 11/8

+10 +12 +12 +10 -39

3/2 3/2 1/1

(517)

I -35 +47 -39 -37 +2 -39

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

II -2 -48 +14

5/4 13/8 3/2

III +14 +23

9/8 8/5

\* +23 -21 +23 -2

III<sup>8/5</sup> 8/5 II<sup>5/4</sup> 7/4

(521)

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

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

II +14 -21 +23 -39

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

III -20 +12 -2

7/4 1/1 5/4

\* +12 +14 +30

I<sup>11/8</sup> II<sup>9/8</sup> I<sup>7/4</sup>

10.3

III 1/1 II 11/8 I 11/8

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

12/11 11/8 11/8

(525)

I +23 +10 +14

8/5 1/1 9/8

II -37 +47

3/2 5/4

III -35 +12 -37

9/8 11/8 3/2

\* -35 +23 +47

III<sup>9/8</sup> I<sup>8/5</sup> III<sup>5/4</sup>

10.4

55/32

III 55/32 II 1/1 I

-39 -39 +23 +10 -39

1/1 55/32 11/8

529

I

II

III

\*

+12 -37

11/8 3/2

533

I

II

III

\*

-37 +47 +30 +2

5/4 7/4 13/8

3/2

-8 +23 +27 -36 +23 +25 -8 +23 +10 -36 +23

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

+47 -39 +47

5/4 1/1 5/4

-36 -8 +47

II<sup>13/8</sup> 7/4 III<sup>5/4</sup>

+47

5/4

10.5

55/32

III 1/1 II 55/32 I

-39 +23 +23 -39

55/32 1/1 1/1

537

I

II

III

\*

+12 +47 -37 +12 -39 +30

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

5/4 3/2 11/8

+25 -25 +10

3/2 11/8 5/4

+27

9/8

+12 -25 +25 +37 -36 -8

I<sup>11/8</sup> III<sup>11/8</sup> 3/2 8/5 13/8 7/4

-45-



10.6

III  $\frac{1}{1}$  II  $\frac{1}{1}$  I  
+23 +23 -39 +23  
1/1 1/1 55/32

(541)

I  $\frac{1}{1}$   $\frac{13}{8}$   $\frac{9}{8}$  +23 -36 +27

II  $\frac{9}{8}$  +27 -8 +25 +23  $\frac{3}{2}$   $\frac{7}{4}$   $\frac{1}{1}$

III  $\frac{3}{2}$  +25 -36  $\frac{13}{8}$  -8 +23 +25  $\frac{7}{4}$   $\frac{1}{1}$   $\frac{3}{2}$

\*  $\frac{1}{5/4}$  +47 +12  $\frac{11}{8}$  -8  $\text{III } \frac{7}{4}$  +23  $\frac{1}{1}$

(545)

I -8 +10 +23 +27 +25  $\frac{7}{4}$   $\frac{5}{4}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{3}{2}$

II +27 -36 -8 +23 -25 +10 +25 -8 -36 -25  $\frac{9}{8}$   $\frac{13}{8}$   $\frac{7}{4}$   $\frac{1}{1}$   $\frac{11}{8}$   $\frac{5}{4}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{13}{8}$   $\frac{11}{8}$

III -25 +10 -8  $\frac{11}{8}$   $\frac{5}{4}$   $\frac{7}{4}$

\* -25 +23 -8 +27 +25 +10 -8 -36 -25 +25 +23  $\frac{11}{8}$   $\frac{1}{1}$   $\frac{7}{4}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{5}{4}$   $\frac{7}{4}$   $\frac{11}{8}$   $\frac{3}{2}$   $\frac{1}{1}$   $\frac{13}{8}$

11.1

III  $\frac{13}{8}$  II  $\frac{13}{8}$  I  
+23 -36 +23 +23  
13/8 1/1 1/1

(549)

I +10 -25 -8 -36  $\frac{5}{4}$   $\frac{11}{8}$   $\frac{7}{4}$   $\frac{13}{8}$

II +27  $\frac{9}{8}$  +25 -25  $\frac{3}{2}$   $\frac{11}{8}$

III -34 -32 +4 -36  $\frac{13}{8}$   $\frac{1}{1}$   $\frac{3}{2}$   $\frac{9}{8}$

\* +10 -36  $\frac{5}{4}$   $\frac{13}{8}$  +10  $\text{II } \frac{5}{4}$  -25  $\frac{11}{8}$

11.2

557

I

+10 +27 -36 -8 +25 +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 +23 -36 +23 +10

13/8 1/1 5/4

I

13/8 -50

II

+33 -50 +4 +15 -34 -36 -32 -50

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

III

+25 +27 +10 -25 -8 +25 +23

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

\*

-50 +27 +10 +4

II 5/4 III 9/8 5/4 II 13/8

-47-

11.4

13/8

5/4

II

13/10

I

+23 +10

-36 +23

+10 -36

5/4

13/8

13/10

565

I

II

III

\*

+33

7/4

+25

3/2

+25

-39

III<sup>3/2</sup>

I<sup>11/8</sup>

-25

11/8

-4

-21

5/4

7/4

11.5

1/1

III

13/10

II

13/10

I

+10 -36

+23 +10

-36 +10

13/10

5/4

13/10

569

I

II

III

\*

-50 -32 +15 -36

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

-50 -36 -34 -32 +33

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

+25 +10

3/2 5/4

+23 +27 +25

1/1 9/8 3/2

-50 -21 +10 -39 +10 -21

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

+10

1/1

-32

9/8

-50

5/4

-21

III<sup>7/4</sup>

573

I

II

III

\*

-39

11/8

+10

1/1

-50

13/8

+15

-32

-34 +33

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

+14

-4

-39 -32

11/8 III<sup>9/8</sup>

-50

5/4

-48-

**11.6**

III 13/10 II 1/1 I

-36 +10 +10 -36 +10

13/10 13/10 1/1

(577)

I

II

III

\*

-50

II 5/4

-36 -50 -32

1/1 5/4 9/8

+10 -50 -4

1/1 13/8 5/4

**11.7**

III 1/1 II 1/1 I

+10 -36 +10 +10

1/1 13/10 1/1

(581)

I

II

III

\*

+14 -4 +10 +12 -4

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

-21

7/4

-21 +14

III 7/4 9/8

(585)

I

II

III

\*

+14 +10 +12 -50 -21 -39 -4 -21 -50 +14 -39 +12 -4

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

9/8 1/1 3/2

-21 +10 -39 -4 +12 +10 -4 -39 +14 -50

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

7/4

+12 +10 -50 -39 +10 -4 +12

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

+12 -39 +10 -4 -21 -39 +14 -50 -4 +10 -50 -21

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

1/1

589

III 13/8

II 1/1

I

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

+10

-50

+12 -39 -4 +14

12.1

III 9/8 16/13 18/13 I

II 9/8 1/1 16/13

I

\* 7/4 5/4 7/4 1/1 3/2

+10 +14 +10 +10 -50

+19

-21 -4 +10 +12

+14

-4

II 5/4

12.2

III 9/8 16/13 18/13 I

II 9/8 1/1 16/13

I

\* 1/1 13/8 3/2

+18

-50 -9 -48

+18 -9 +12

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

601

-50 -46

1/1 9/8

-21 -21 -4

7/4 5/4 7/4

-35 -46

11/8 13/8

-21 +2 -4 +18 -9

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

12.3

18/13 18/13 1/1 I

+14 -50 +10 +14 -50

18/13 9/8 1/1

605

-50 +2 +19 -48

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

-46

13/8

+37

5/4

+19 -46

7/4 III 9/8

12.4

1/1 1/1 1/1 I

-50 +14 -50 -50

1/1 18/13 1/1

609

+37 -9 -46 -50 -48 +37 +19 -9 +19

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

+18 +14 -35

1/1 11/8 9/8

+19 -9 -50 -48

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

-46 +37 +2

9/8 5/4 11/8

613

I

-46 +37 +2 +19 -9

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

II

-50 +19 -50 -48 +37

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

III

-46 +19 -48 +2 -46 +37 -9 -50 +19 -48 -50

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

\*

-50 +37 -48 -46 -50 +37 +19 -46

III<sup>1/1</sup> 5/4 3/2 9/8 1/1 5/4 7/4 9/8

13.1

1/1

III 5/4 II 5/4 I

-50 +37 -50 -50

5/4 1/1 1/1

617

I

+2

11/8

II

-46 +19

9/8 7/4

III

+19 -48 -12 +40

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

\*

-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

9/8

III 5/4 II 10/9 I

+37 -50 -50 -46

1/1 1/1 9/8

621

I

-48 -50 +2 -9 -50

3/2 1/1 11/8 13/8 1/1

II

+5 +37 -23

7/4 1/1 13/8

III

-9 +38 -46 +37

13/8 III<sup>3/2</sup> II<sup>9/8</sup> 5/4

\*

13.3

III 10/9 II 5/4 9/8 I

+37 -46 -50 +37 -46 -50

10/9 5/4 9/8

625

I

-44 -5 -46 -42 -46 +23 +40 -44

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

II

+37 -48 -50 -48 -46 +2

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

III

-48 +37 +38

6/5 1/1 3/2

\*

+2 -48 -44 +38

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

629

I

+37 +2 +19 -48 -50 -46 +37 +19 -48 -9 -48 +37 -50

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

II

III

+40 -46 -44

5/4 1/1 3/2

\*

-23

II<sup>13/8</sup>

13.4

III 10/9 II 1/1 I

-46 +37 -50 -46

1/1 1/1 9/8

633

I

+19

7/4

II

-48 +40 +37 +5

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

III

+23 -44

7/4 3/2

\*

-23 -46 +37

13/8 I<sup>9/8</sup> 5/4 -53-



637

13.5

III 10/9 II 1/1 I

-46 -37 -46

1/1 1/1 1/1

I

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

-42 -44 -42

+40

II

-23 +40 -12

+38

13/8 9/8 3/2 11/8

III

-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

641

13.6

III 1/1 II 1/1 I

-46 +37 -46

1/1 10/9 1/1

I

-5 -46 +5 +23

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

II

+37

1/1

III

+5 -44 -46 +23 -5 +5 -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>

645

I

-44 -46 -46 -5 +23 -44 +40

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

3/2 11/8

II

III

+40 +23 -42 -46 +23 -5 -42 -46 +40 +23 -44

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

11/8

\*

-44 -5 +23 -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 -42

I 9/8

II +23 -40 -46 +23 -44 -5 -44 +40

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

III -46 -5 +5

1/1 13/8 11/8

\* -46 +40 -5

1/1 5/4 13/8

14.1

III 3/2 II 1/1 I

-46 -46 -44 -46

1/1 3/2 1/1

653

I -46

1/1

II +7 -42 +42 -44 +42 -3 +7 +25

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

III -42

9/8

\* -44 +23

3/2 III 7/4

14.2

III 9/8 II 4/3 I

-46 -42 -44 -46 -46 -44

9/8 3/2 3/2

657

I -44 +5 +40

3/2 11/8 5/4

II +42 -3 +42

5/4 13/8 5/4

III -44

3/2

\* +42 -42

II 5/4 III 9/8

**14.3**

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

1/1 3/2 1/1

(661)

I 1/1 7/4 9/8 5/4

II -46 1/1

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

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

(665)

I -44 1/1 +7 +42 -44 -42  
11/8 5/4 1/1 3/2

II -40 9/8 +7 11/8

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

II -42 3/2

III +9 11/8

\* +25 7/4 -3 III 13/8  
-56-

673

-44 +25

7/4 1/1

-44 +25 -42 -44

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

+42 -40

5/4 9/8

+7 -44 +42 -42 -44 +25 +42

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

13/8

677

-42 +7

3/2 11/8

-44

1/1

-44 +25 -42 -3 -42

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

15.1

12/11

III 4/3 II 16/11 I

-44 -44 -42 -44 +7

1/1 4/3 16/11

681

+7 +11 +7 -24 +9 +7

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

-38 -1 -40 +27 -42 -40

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

+25 +7

7/4 11/8

+44 -40 +42 +42 +7

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

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

III  $\frac{16}{11}$  II  $\frac{12}{11}$  I

-44 +7 -42 -44 +7 -42

(685)  $\frac{16}{11}$   $\frac{4}{3}$   $\frac{12}{11}$

I  $\frac{13}{8}$   $\frac{5}{4}$   $\frac{9}{8}$   $\frac{11}{8}$   $\frac{7}{4}$   $\frac{3}{2}$   $\frac{5}{4}$   $\frac{11}{8}$   $\frac{7}{4}$   $\frac{3}{2}$   $\frac{13}{8}$

II  $\frac{11}{8}$   $\frac{7}{4}$   $\frac{5}{4}$   $\frac{1}{1}$   $\frac{7}{4}$   $\frac{11}{8}$

III  $\frac{3}{2}$   $\frac{13}{8}$   $\frac{1}{1}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{6}{5}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{13}{8}$

\*  $\frac{11}{8}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{1}{1}$   $\frac{7}{4}$   $\frac{1}{5/4}$

+23 +25 +44

III  $\frac{6}{5}$  II  $\frac{7}{4}$  I  $\frac{5}{4}$

**15.3**  $\frac{14}{11}$

III  $\frac{1}{1}$  II  $\frac{14}{11}$  I

+7 -44 +7 -42 +25

(689)  $\frac{1}{1}$   $\frac{16}{11}$   $\frac{7}{6}$

I  $\frac{5}{4}$   $\frac{11}{8}$   $\frac{1}{1}$

II  $\frac{1}{1}$   $\frac{13}{8}$   $\frac{3}{2}$   $\frac{6}{5}$

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

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

+44 +25 -24 +25

+7 +9 +48 +23

**15.4**  $\frac{14}{11}$

III  $\frac{14}{11}$  II  $\frac{1}{1}$  I

+7 +7 +25 +25 +7

(693)  $\frac{1}{1}$   $\frac{14}{11}$   $\frac{14}{11}$

I  $\frac{9}{8}$   $\frac{3}{2}$   $\frac{13}{8}$   $\frac{9}{8}$

II  $\frac{7}{4}$

III  $\frac{6}{5}$   $\frac{3}{2}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{13}{8}$   $\frac{1}{1}$

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

+29 -24 +7 +9 +11 +48 +7 +29

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

15.5

14/11 1/1 14/11 I

+7 +25 +25 +7

14/11 1/1 1/1

697

I

II

III

\*

-35 +25 +27 -6 -24 +25 +29

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

-24 +23 +9 +11

7/4 13/8 6/5 9/8

+48 -24

III 13/8 7/4

701

I

II

III

\*

-41 -24 +23 +7

11/8 7/4 13/8 6/5 1/1

-24 -35 +25 +11 +29

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

-24 -35 +25

11/8 13/8 1/1

+27 -24 -35 -6 +23 +29

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

15.6

1/1 14/11 14/11 I

+25 +7 +25 +7 +25

14/11 1/1 14/11

705

I

II

III

\*

-35 -6 -24 -35 +27

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

+9

3/2

+48

I 13/8

**15.7**

III  $\frac{1}{1}$  II  $\frac{14}{11}$  I

+7 +25 +25 +25 +7

14/11 1/1 14/11

709

I  $\frac{5}{4}$   $\frac{3}{2}$  +11 +27

II  $\frac{5}{4}$   $\frac{1}{1}$  +11 +25

III  $\frac{7}{4}$   $\frac{6}{5}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{9}{8}$   $\frac{7}{4}$  -24 +23 +7 +48 +11 +23 -24

\*  $\text{III} \frac{7}{4}$  -24 +27  $\text{II} \frac{3}{2}$

713

I  $\frac{1}{1}$   $\frac{7}{4}$   $\frac{6}{5}$   $\frac{13}{8}$   $\frac{7}{4}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{11}{8}$  -24 +7 +23 +48 -24 +7 +11 -41

II  $\frac{3}{2}$  +27

III  $\frac{11}{8}$  -24

\*  $\text{III} \frac{13}{8}$  -35  $\frac{7}{4}$  -6  $\frac{9}{8}$  +29  $\frac{5}{4}$  +11

**15.8**

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

+25 +25 +7 +25

1/1 1/1 14/11

717

I  $\frac{3}{2}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{9}{8}$   $\frac{11}{8}$   $\frac{3}{2}$   $\frac{9}{8}$  +27 +25 -35 +29 -24 +11 +29

II  $\frac{1}{1}$   $\frac{13}{8}$   $\frac{5}{4}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{5}{4}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{11}{8}$   $\frac{3}{2}$   $\frac{1}{1}$   $\frac{13}{8}$  -35 +25 +27 +11 +29 +27 -6 +11 +25 +29 -24 +27 +25

III  $\frac{1}{1}$   $\frac{3}{2}$   $\frac{5}{4}$   $\frac{9}{8}$   $\frac{3}{2}$   $\frac{7}{4}$   $\frac{5}{4}$   $\frac{1}{1}$   $\frac{9}{8}$   $\frac{11}{8}$   $\frac{3}{2}$   $\frac{1}{1}$   $\frac{13}{8}$  -35 +25 +27 +11 +29 +27 -6 +11 +25 +29 -24 +27 +25

\*  $\text{III} \frac{7}{4}$  -6  $\frac{5}{4}$  +11

721

I  $\frac{5}{4}$   $\frac{1}{1}$

II  $\frac{1}{1}$   $\frac{7}{4}$   $\frac{3}{2}$   $\frac{11}{8}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{5}{4}$   $\frac{1}{1}$   $\frac{11}{8}$   $\frac{3}{2}$

III  $\frac{5}{4}$   $\frac{9}{8}$

\*  $\frac{9}{8}$   $\frac{11}{8}$   $\frac{1}{1}$   $\frac{7}{4}$   $\frac{5}{4}$

+11 +25 +29 +27 +25 -6 -35 +11 +25 -24

16.1

III  $\frac{9}{8}$  II  $\frac{9}{8}$  I  $\frac{1}{1}$

+25 +29 +25 +25 +25

725

I  $\frac{13}{8}$

II  $\frac{5}{4}$

III  $\frac{1}{1}$   $\frac{7}{4}$

\*  $\frac{3}{2}$   $\frac{9}{8}$   $\text{III}^{\frac{6}{5}}$   $\frac{6}{5}$   $\text{II}^{\frac{3}{2}}$

+27 +29 +44 +27 -35

16.2

III  $\frac{9}{8}$  II  $\frac{1}{1}$  I  $\frac{9}{8}$

+29 +25 +25 +29 +25

729

I  $\frac{3}{2}$   $\frac{7}{4}$   $\frac{1}{1}$

II  $\frac{7}{4}$   $\frac{9}{8}$   $\frac{1}{1}$   $\frac{13}{8}$   $\frac{5}{4}$

III  $\frac{13}{8}$   $\frac{1}{1}$   $\frac{9}{8}$

\*  $\frac{13}{8}$   $\frac{7}{4}$

+27 -6 +25 -6 +29 +25 -35 +11 -31 +29 +33 -6 -35



The musical score for '16.3' consists of four staves, labeled I, II, III, and \*. Staff I is in treble clef, while staves II, III, and \* are in bass clef. The score is divided into two measures by a double bar line. Above the staves, there are various musical notations including notes, rests, and dynamic markings. Above staff I, there are markings for '16.3', 'III 9/8', 'II 9/8', and 'I 1/1'. Above staff II, there are markings for '+25', '+29', '+29', '+25', and '+25'. Above staff III, there are markings for '9/8', '9/8', and '1/1'. Above staff \*, there are markings for '9/8', '9/8', and '1/1'. The score includes various musical notations such as notes, rests, and dynamic markings.

The image displays a musical score for the song "The Rose Tree". It consists of four staves, labeled I, II, III, and \*. Staff I is in treble clef and contains the melody with time signatures 9/8, 5/4, and 1/1. Staff II is in alto clef and contains a harmonic line with time signatures 5/4, 13/8, and 1/1. Staff III is in bass clef and contains another harmonic line with time signatures 6/5 and 1/1. Staff \* is in treble clef and contains a fourth line with time signatures 11/8 and 13/8. Various accidentals and fingerings are indicated throughout the score.

**16.4**

16/13

III  $\frac{9}{8}$  II  $\frac{18}{13}$  I

+29 +25 +25 -35

1/1 1/1 16/13

745

I

II

III

\*

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

5/4

7/4 III  $\frac{7}{4}$  II  $\frac{5}{4}$

749

I

II

III

\*

1/1 11/8

11/8 9/8

3/2 1/1

III  $\frac{9}{8}$  I  $\frac{13}{8}$  III  $\frac{6}{5}$  11/8 3/2

**16.5**

16/13

III  $\frac{18}{13}$  II  $\frac{9}{8}$  I

+29 +25 -35 -35 +25

1/1 16/13 16/13

753

I

II

III

\*

5/4 7/4 3/2

1/1

7/4 13/8 9/8

III  $\frac{7}{4}$  II  $\frac{11}{8}$  I  $\frac{5}{4}$  II  $\frac{13}{8}$

16.6

1/1

III

18/13

II

18/13

I

+29 -35 18/13 -35 +29 18/13 +25 +29 9/8

757

I

II

III

\*

1/1 11/8

9/8

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

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

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

16.7

1/1

III

18/13

II

18/13

I

-35 +29 18/13 +29 -35 18/13 +29 -35 18/13

761

I

II

III

\*

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

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

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

765

I

II

III

\*

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

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

**16.8**

III 18/13 II 1/1 I  
+29 -35 -35 +29 -35

18/13 18/13 1/1

769

I

II

III

\*

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

-33 -31 +17 -35

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

+31 -20 +33 +31 +29

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

+6 +17 -33 +34 -48

+34

III<sup>7/4</sup>

**16.9**

III 18/13 II 1/1 I  
-35 +29 -35

1/1 1/1 1/1

773

I

II

III

\*

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

-20 +31 -2 -20

11/8 6/5 13/8

+44 -31

-33 +17 -31

3/2 11/8 9/8

1/1

-20 +31 -48

II<sup>11/8</sup> 3/2 III<sup>5/4</sup>

**16.10**

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

1/1 18/13 1/1

777

I

II

III

\*

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

+34 +6 -33 -31

3/2 11/8

+31 +29 -2 +33 +31 +29 -20

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

11/8

-33 +17

11/8 3/2

to kill a monarch (seed:19800725)

781

I

II

III

\*

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

9/8 3/2 1/1

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

III 9/8 11/8

785

I

II

III

\*

13/8 7/4

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

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

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
22 */
```

### 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 //-----CHOOSE AND COLLECT FUNCTIONS-----
78 hdChoose = {arg mode, exp = 1, weights = [1, 1, 1, 1, 1, 1, 1];
79   var probs;
80   probs = pow((1 / mode.collect({arg fr;
81     harmonicDistance.value(if(fr == [1], [1], {[2], [1]}), {fr}))), exp) * weights;
82   mode.wchoose(probs.normalizeSum)
83 };
84
85 wchooseDict = {arg dict, exp = 1, limit = 0, isFR = true;
86   var keyList, probs;
87   keyList = if(isFR, {
88     dict.keys.asList.sort({arg a, b; harmonicDistance.value(a) < harmonicDistance.value(b)});
89   }, {
90     dict.keys.asList.sort({arg a, b; a.convertDigits(2) < b.convertDigits(2)});
91   });
92   probs = keyList.collect({arg key;
93     var count = dict[key][\count];
94     if(count < limit, {0}, {count})
95   });
96   probs = pow(probs, exp);
97   keyList.wchoose(probs.normalizeSum)
98 };
99
100 collectRoots = {arg dict; dict.keys.collect({arg fr;
101   dict[fr][\root][0]}.asList.sort({arg a, b; harmonicDistance.value(a) < harmonicDistance.value(b)});
102 });
103
104 //-----GENERATE MODE SEQUENCE-----
105
106 initModeState = {
107   var curModeState, frSet;
108   curModeState = Dictionary.new;
109   frSet = [[1, 1], [9, 8], [6, 5], [11, 8], [3, 2], [13, 8], [7, 4]].collect({arg fr;
110     [1] ++ fr[0].factors, [1] ++ fr[1].factors});
111   frSet.do({arg fr;
112     var mode, count;
113     mode = genMode.value;
114     count = if(fr == [1], [1], {10}, {1});
115     curModeState.add(fr->
116       Dictionary.with(*(\count->count, \mode->frSet, \root->[[[1], [1], frSet], \mult->fr, \fr->fr]))
117   });
118   curModeState;
119 };
120
121 advanceMode = {arg lastModeState, lastCadenceState, forceHS = false;
122   var curModeState, curRoots, lastRoots, lastCadenceRoot, changeCount, modSpeed;
123
124   curModeState = lastModeState.deepCopy;
125   curRoots = collectRoots.value(curModeState);
126   lastRoots = collectRoots.value(lastModeState);
127   lastCadenceRoot = collectRoots.value(lastCadenceState).asList[0];
128   changeCount = 0;
129   modSpeed = if(forceHS, {1}, {[1, 2, 3].wchoose([2, 3, 1].normalizeSum)});
130
131   while({
132     ((curRoots == lastRoots) && (changeCount < modSpeed)) ||
133     (changeCount < modSpeed) && (curRoots.size > 1)
134   }, {
135     var roots, rootSel, mults, multProbs, multSel, new;
136
137     //bump for length of time its been around
138     curModeState.keysValuesDo({arg key, val;
139       val[\count] = val[\count] + 1;
140       if(val[\count] > 100, {val[\count] = 1});
141     });
142
143     // max 3 roots that were not the last cadencial root and only in the mode of the last cadence
144     roots = curModeState.reject({arg val;
145       (val[\fr] == lastCadenceRoot) ||
146       lastCadenceState.includesKey(val[\fr]).not || //consider two steps out?
147       ((curRoots.size >= 3) && curRoots.includes(val[\fr]).not)
148     });
149
150     rootSel = wchooseDict.value(roots, 1, 2);
151     mults = curModeState[rootSel][\mode];
152     multProbs = mults.collect({arg fr;
153       if(curModeState.keys.includes(frCollapse.value(frAdd.value(rootSel, fr))), {1}, {2})});
154     multSel = hdChoose.value(mults, 0.5, multProbs);
155     new = frCollapse.value(frAdd.value(rootSel, multSel));
156
157     curModeState[rootSel][\count] = curModeState[rootSel][\count] + 2; //bump if gets chosen as a root
158
159     if(curModeState.includesKey(new), {
160       //bump if it gets chosen again
161       curModeState[new][\count] = curModeState[new][\count] + 1;
162       if((curModeState[new][\count] >= 20), {
163         curModeState[new][\root] = [rootSel, curModeState[rootSel][\mode]]
164       }

```

```

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 //-----GENERATE TEMPORAL FRAMEWORK-----
185 initTemporalState = {
186   Dictionary.with(*[[0, 1], [0, 1], [0, 1]].allTuples.collect({arg tuple;
187     tuple->Dictionary.with(*[\count->1]))))
188 };
189
190 genTemporalData = {arg lastTupleState, modeState, cadenceOverride, noParts = 4;
191   var cadence, curTupleState, timeToNextEvent, tuple, temporalData;
192
193   cadence = if(collectRoots.value(modeState).size == 1, {cadenceOverride.not}, {false});
194   curTupleState = lastTupleState.deepCopy;
195   timeToNextEvent = (64 + 50.rand + if(cadence, {50}, {0})).round(16);
196   tuple = wchooseDict.value(curTupleState, isFR: false);
197   if(cadence, {tuple = [1, 1, 1]});
198   curTupleState = curTupleState.keysValuesDo({arg key, val;
199     curTupleState[key][\count] = val[\count] + 1});
200   curTupleState[tuple][\count] = 0;
201   tuple = if(cadence, {[1, 1, 1, 1]}, {[0] ++ tuple});
202
203   temporalData = noParts.collect({arg p;
204     var flourishDensity, genDensity, flourish, beforeLen, before, after, buffer;
205     flourishDensity = if(tuple[p] == 1, {0.125 + 0.5.rand}, {3});
206     if((p == 0) && cadence.not, {flourishDensity = 3});
207     genDensity = if(p == 0, {5}, {20});
208
209     flourish = (if(cadence, {16}, {8}) + 32.rand).collect({[0, 1].wchoose({flourishDensity, 1}.normalizeSum)});
210     buffer = 16.collect({0});
211     beforeLen = ((timeToNextEvent - flourish.size - buffer.size) / if(cadence, {1.25}, {1})).asInteger.rand;
212     before = beforeLen.collect({arg i; [0, 1].wchoose({genDensity, 0.25}.normalizeSum)});
213     after = (timeToNextEvent - before.size - flourish.size - buffer.size).collect({[0, 1].wchoose({genDensity, 1}.normalizeSum)});
214     flourish = before ++ flourish ++ after;
215     if(flourish.sum == 0, {flourish[flourish.size.rand] = 1});
216     flourish = buffer ++ flourish;
217   });
218
219   [temporalData, curTupleState]
220 };
221
222 //-----GENERATE ENSEMBLE PARTS-----
223 initPartStates = {
224   var allRatios;
225   allRatios = [[1, 1], [9, 8], [5, 4], [6, 5], [4, 3], [11, 8], [3, 2], [8, 5], [13, 8], [15, 8], [7, 4]];
226   Dictionary.with(*
227     4.collect({arg part;
228       part->Dictionary.with(*[\multCounts->
229         Dictionary.with(*allRatios.collect({arg fr;
230           [[1] ++ fr[0].factors, [1] ++ fr[1].factors]->1])),
231           \noteCount->0, \index->part, \lastFreq->0, \lastFreqRatio->[[1], [1]], \lastDur->0
232         })
233     })
234   )
235 };
236
237 //this is how roots are distributed to the parts
238 distributeRoots = {arg modeState, lastRoots;
239   var roots;
240   roots = modeState.keys.asList.collect({arg fr; modeState[fr][\root]});
241   roots = roots.asBag.contents.asPairs.reverse.clump(2);
242   roots = roots.sort({arg a, b; a[0] > b[0]}).collect({arg item; item[1]}).wrapExtend(4);
243   roots = [roots[0]] ++ roots[.2].scramble;
244   roots = 4.collect({arg part;
245     var root, rootMod, rootFreq, mode;
246     root = roots[part];
247     rootMod = frDiff.value(root[0], lastRoots[part]);
248     rootFreq = 36.midicps * pow(2, [1, 0, 1, 2][part]) * frToFloat.value(root[0]);
249     [root[0], root[1], rootMod, rootFreq]
250   });
251   roots = roots.collect({arg root, r;
252     var rootRels;
253     rootRels = 4.collect({arg p; frDiff.value(root[0], roots[p][0])});
254     rootRels.removeAt(r);
255     root.add(rootRels)
256   });
257   roots
258 };
259
260 genEnsemblePart = {arg partState, modeState, temporalData, roots, part, offset;
261   var trans, root, mults, rootMod, amp, firstChange, cadence, lastInsRef, ensData;
262
263   trans = pow(2, [1, 0, 1, 2][partState[\index]]);
264   # root, mults, rootMod = roots[part];
265   amp = [0, 1, 2, 3].wchoose([0, 2, 2, 2].normalizeSum);
266   firstChange = false;
267   cadence = if(collectRoots.value(modeState).size == 1, {true}, {false});
268   lastInsRef = nil;
269
270   ensData = [];
271   temporalData.do({arg val, ts;
272     var timeStamp, comp, change;
273
274     partState[\lastDur] = partState[\lastDur] + 1;
275     timeStamp = ts + offset;
276     change = [val == 1, (val == 1) && firstChange.not].wchoose([1, 2].normalizeSum);
277     if(
278       (partState[\index] == 0) &&
279       (frToFloat.value(partState[\lastFreqRatio]) >= 4.0) &&
280       (partState[\lastDur] < 16) && cadence.not,
281       {change = false}
282     );
283
284     if(change, {
285       var mult, multWeights, freq, rootFreq, insRef;
286
287       //this weights notes that are richer and mixes with the DCA algorithm
288
289

```



```

290 multWeights = mults.collect({arg fr;
291     var comp = frCollapse.value(frAdd.value(root, fr));
292     if(modeState.keys.includes(comp), {3}, {1}) * pow(partState[\multCounts][fr], 1);
293 });
294
295 mult = hdChoose.value(mults, 0.5, multWeights);
296 mults.do({arg fr; partState[\multCounts][fr] = partState[\multCounts][fr] + 1});
297 partState[\multCounts][mult] = 0;
298
299 freq = 36.midicps * trans * frToFloat.value(frAdd.value(root, mult));
300
301 //flute special case
302 if((partState[\index] == 0) && cadence.not, {
303     var mode, continue, freqRatio;
304     mode = modeState.keys.asList.collect({arg fr;
305         [
306             frCollapse.value(frAdd.value(modeState[fr][\root][0], modeState[fr][\mult])),
307             modeState[fr][\root][0], modeState[fr][\mult]
308         ]
309     });
310     mode = mode.sort({arg a, b; frToFloat.value(a[0]) < frToFloat.value(b[0])});
311     mode = mode ++ mode.collect({arg fr; [frAdd.value(fr[0], [[2], [1]]), fr[1], fr[2]]});
312     mode = mode ++ mode.collect({arg fr; [frAdd.value(fr[0], [[4], [1]]), fr[1], fr[2]]});
313     continue = true;
314     while({continue}, {
315         # freqRatio, root, mult = mode[partState[\noteCount] % 15];
316         freq = 36.midicps * trans * frToFloat.value(frAdd.value([1], [1]), freqRatio));
317         continue = (freq <= partState[\lastFreq]) && (partState[\noteCount] % 15) != 0;
318         partState[\noteCount] = partState[\noteCount] + 1;
319     });
320     partState[\lastFreq] = freq;
321     partState[\lastFreqRatio] = freqRatio;
322     insRef = roots.slice(nil, 0).deepCopy.drop(1).indexOfEqual(root) + 1;
323     insRef = if(lastInsRef != insRef, {lastInsRef = insRef; insRef}, {lastInsRef = insRef; nil});
324 });
325 if((partState[\index] == 0) && cadence, {
326     insRef = if(firstChange.not, {1}, {nil});
327 });
328
329 rootFreq = 36.midicps * trans * frToFloat.value(root);
330
331 if((partState[\index] == 0) && ((partState[\noteCount] % 15) == 1) && cadence.not, {ensData = ensData.add([0, timeStamp - 8, 0, 0, 0])});
332 ensData = ensData.add([freq, timeStamp, amp, mult, insRef]);
333 firstChange = true;
334 partState[\lastDur] = 0;
335 if((partState[\index] == 0) && cadence, {partState[\lastDur] = 32});
336 });
337
338 ensData = [[0, ensData[0][1] - 4, 0, 0, 0]] ++ ensData;
339 [ensData, partState]
340 };
341
342 //-----GENERATE ELECTRONIC ACCOMPANIMENT-----
343 genAccompPart = {arg modeState, temporalData, offset, trans, part, register;
344     var firstChange, accompData;
345     firstChange = false;
346     accompData = [];
347     temporalData.do({arg val, tS;
348         var change;
349         change = [val == 1, (val == 1) && firstChange.not].wchoose([1, if(part == 0, {5}, {3})].normalizeSum);
350         if(change, {
351             var sel, freq, amp;
352             sel = wchooseDict.value(modeState, 0.1);
353             freq = 48.midicps * trans * frToFloat.value(sel);
354             amp = [0, 1, 2, 3].wchoose([2, 2, 1, 1].normalizeSum);
355
356             accompData = accompData.add([freq, tS + offset, amp, part]);
357             firstChange = true;
358         });
359     });
360     accompData
361 };
362
363 //-----GENERATE ELECTRONIC BASS-----
364 genBassPart = {arg root, ampCurve, hi;
365     var freq;
366     freq = if(hi,
367         {36.midicps * frToFloat.value(frCollapse.value(frAdd.value(root, [[3], [2]]))},
368         {36.midicps * frToFloat.value(root)});
369     ampCurve.collect({arg sec, iter; [freq, sec[1]]})
370 };
371
372 //-----GENERATE AMP CURVES-----
373 genAmpCurve = {arg temporalData1, temporalData2, offset1, offset2, type;
374     var firsts1, firsts2, delay, attack, decay, release, min, max, env;
375     firsts1 = temporalData1.collect({arg ptd; ptd.indexOf(1)});
376     firsts2 = temporalData2.collect({arg ptd; ptd.indexOf(1)});
377     delay = switch(type)
378     {0} {0}
379     {1} {0}
380     {2} {firsts1.minItem};
381     attack = switch(type)
382     {0} {offset2 - offset1}
383     {1} {offset2 - temporalData2[0].size + firsts2.minItem - offset1}
384     {2} {firsts1.maxItem - firsts1.minItem};
385     decay = switch(type)
386     {0} {0}
387     {1} {firsts2.maxItem - firsts2.minItem}
388     {2} {temporalData1[0].size - firsts1.maxItem};
389     release = switch(type)
390     {0} {0}
391     {1} {temporalData2[0].size - firsts2.maxItem}
392     {2} {(offset2 - temporalData2[0].size) - offset1};
393     min = switch(type)
394     {0} {0.15}
395     {1} {0}
396     {2} {0};
397     max = switch(type)
398     {0} {0.5}
399     {1} {0.65}
400     {2} {1};
401
402     env = Env.dadsr(delay, attack, decay, 0.25, release, curve: \cub).range(min, max);
403     ((delay + attack + decay + release) / 1).asInteger.collect({arg iter; [env.at(iter * 1), offset1 + (iter * 1)]})
404 };
405
406 //-----GENERATE ALL MUSIC DATA-----
407 genMusicData = {arg seed;
408     var minTotalDur, minSection1Dur, dUnit, curLen, cadence,
409     ultimateSubsection, ultimateSection, ultimateCadenceCount,
410     minTotalLen, minSection1Len,

```

```

415 modeState, temporalState, partStates,
416 lastCadenceTemporalData, lastCadenceState, lastSectionPoint,
417 ensData, accompData, bassData, ampData,
418 sectionData, sectionNavDict,
419 sectionCount, subsectionCount,
420 lastRoots, roots, ampDataTmp;
421
422 thisThread.randSeed = seed;
423
424 # minTotalDur, minSection1Dur, dUnit, curLen, cadence = [23 * 60, 8 * 60, 8.reciprocal, 0, false];
425 # ultimateSubsection, ultimateSection, ultimateCadenceCount = [false, false, 0];
426 # minTotalLen, minSection1Len = [(minTotalDur / dUnit).round(16), (minSection1Dur / dUnit).round(16)];
427 # modeState, temporalState, partStates = [initModeState.value, initTemporalState.value, initPartStates.value];
428 # lastCadenceTemporalData, lastCadenceState, lastSectionPoint = [nil, modeState.deepCopy, 0];
429 # ensData, accompData, bassData, ampData = [4.collect({[]}), 4.collect({6.collect({[]})}), 2.collect({[]}), 3.collect({[]})];
430 # sectionData, sectionNavDict = [Dictionary.new, Dictionary.new];
431 # sectionCount, subsectionCount = [1, 1];
432
433 while((curLen < minTotalLen) || ((curLen >= minTotalLen) && cadence.not) || ultimateSection.not), {
434   var temporalData;
435   # temporalData, temporalState = genTemporalData.value(temporalState, modeState, curLen <= minSection1Len);
436
437   collectRoots.value(modeState).collect({arg fr; [fr[0].asFloat.product, fr[1].asFloat.product]}).postln;
438   //modeState.keys.postln;
439   "-----".postln;
440
441   lastRoots = if(curLen == 0, {4.collect({[1], [1]})}, {roots.slice(nil, 0)});
442   roots = distributeRoots.value(modeState, lastRoots);
443
444   sectionData.add((curLen / 4).asInteger->[roots, lastRoots.collect({arg fr, part;
445     [fr, 36.midicps * pow(2, [1, 0, 1, 2][part]) * frToFloat.value(fr)]]}, sectionCount, subsectionCount, cadence, ultimateSubsection));
446   sectionNavDict.add([sectionCount, subsectionCount]->[(curLen / 16 + 1).asInteger]);
447
448   4.do({arg part;
449     var musicData, partState, accompRoutine;
450     # musicData, partState = genEnsemblePart.value(partStates[part], modeState, temporalData[part], roots, part, curLen);
451     ensData[part] = ensData[part] ++ musicData;
452     partStates[part] = partState;
453
454     //use an independent random number generator for the accompaniment
455     accompRoutine = Routine({
456       thisThread.randSeed = Date.seed;
457       6.do({arg register;
458         musicData = genAccompPart.value(modeState, temporalData[part], curLen, pow(2, part + register), part, register);
459         accompData[part][register] = accompData[part][register] ++ musicData;
460       });
461     });
462     accompRoutine.value;
463
464   });
465
466   subsectionCount = subsectionCount + 1;
467
468   if(curLen == 0, {
469     lastCadenceTemporalData = temporalData;
470     lastSectionPoint = curLen;
471   });
472
473   curLen = curLen + temporalData[0].size;
474
475   if(curLen > minSection1Len, {
476     if(collectRoots.value(modeState).size == 1, {
477
478       ampData[0] = ampData[0] ++ genAmpCurve.value(lastCadenceTemporalData, temporalData, lastSectionPoint, curLen, 0);
479       ampDataTmp = genAmpCurve.value(lastCadenceTemporalData, temporalData, lastSectionPoint, curLen, 1);
480       ampData[1] = ampData[1] ++ ampDataTmp;
481       bassData[0] = bassData[0] ++ genBassPart.value(collectRoots.value(modeState).asList[0], ampDataTmp, true);
482
483       if(sectionCount == 1, {
484         ampData[2] = ampData[2] ++ ((curLen - temporalData[0].size) / 1).asInteger.collect({arg iter; [0, iter * 1]});
485         bassData[1] = bassData[1] ++ ((curLen - temporalData[0].size) / 1).asInteger.collect({arg iter; [0, iter * 1]});
486       }, {
487         ampDataTmp = genAmpCurve.value(lastCadenceTemporalData, temporalData, lastSectionPoint, curLen, 2);
488         ampData[2] = ampData[2] ++ ampDataTmp;
489         bassData[1] = bassData[1] ++ genBassPart.value(collectRoots.value(lastCadenceState).asList[0], ampDataTmp, false);
490       });
491
492       sectionData.add((curLen - temporalData[0].size) / 4).asInteger->
493       sectionData[(curLen - temporalData[0].size) / 4].asInteger.put(5, true));
494       (subsectionCount - 1).do({arg subsectionIndex;
495         sectionNavDict.add([sectionCount, subsectionIndex + 1]->[sectionNavDict[[sectionCount, subsectionIndex + 1][0], subsectionCount - 1]);
496       });
497       # sectionCount, subsectionCount = [sectionCount + 1, 1];
498       # lastCadenceTemporalData, lastCadenceState, lastSectionPoint = [temporalData, modeState, curLen];
499       cadence = true;
500
501       //this should ensure that the final cadence is a HS
502       if(curLen >= minTotalLen, {ultimateCadenceCount = ultimateCadenceCount + 1});
503       ultimateSection = ultimateCadenceCount > 1;
504     }, {
505       cadence = false
506     });
507     modeState = advanceMode.value(modeState, lastCadenceState, curLen >= minTotalLen);
508   });
509 });
510
511 ampDataTmp = genAmpCurve.value(lastCadenceTemporalData, [[1], [1], [1], [1]], lastSectionPoint, lastSectionPoint + 360, 2);
512 ampData[2] = ampData[2] ++ ampDataTmp;
513 bassData[1] = bassData[1] ++ genBassPart.value(collectRoots.value(lastCadenceState).asList[0], ampDataTmp, false);
514
515 [ensData, accompData, bassData, ampData, sectionData, sectionNavDict]
516 };
517 )

```

## tkam\_sonifier.scd

```

1 (
2   var formatPatternData;
3
4   //-----ALLOCATE BUSES-----
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, preampBuses, accompBuses, postampBuses;
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.kr(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.kr(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 // group data by measures for navigation
75 formatPatternData = {arg musData, measureLen, rel, print = false;
76   var dataLen;
77   dataLen = musData[0][0].size + 1;
78   musData.collect({arg partData;
79     var res;
80     res = partData;
81     res = res.collect({arg mData, index; mData.add(if(index != (res.size - 1), {rel}, {5.rand + 5})));
82     res = res.flop;
83     res = res.add(res[1]);
84     res[1] = (res[1].differentiate.drop(1) ++ [10]);
85     res = res.flop ++ measureLen.collect({arg measure; dataLen.collect({0}) ++ [measure * 16]});
86     res = res.sort({arg a, b; a.last < b.last}).flop;
87     res = res.insert(1, (res.last.differentiate.drop(1) ++ [10])).flop;
88     res = res.separate({arg a, b; (a.last / 16).trunc != (b.last / 16).trunc});
89     res.collect({arg measureData; measureData.flop})
90   }).flop
91 };
92
93 //-----GENERATE PATTERNS-----
94 //this generates patterns grouped by measures except for bass data and amp curve data which are much higher resolution
95 //these are used to make playable patterns
96 `genPatterns = [arg ensData, accompData, bassData, ampData, sectionData, preampBusses, accompBusses, postampBusses;
97   var measureLen, ensDataFormatted, accompDataFormatted, bassDataFormatted, ampDataFormatted,
98   dUnit, section, subsection, patterns;
99
100   measureLen = ((
101     ensData.collect({arg partData; partData.last[1]}) ++
102     accompData.flatten.collect({arg partData; partData.last[1]})
103   ).maxItem.ceil(16) / 16).asInteger + 1;
104
105   ensDataFormatted = formatPatternData.value(ensData, measureLen, 0.1, true);
106   accompDataFormatted = formatPatternData.value(accompData.flatten, measureLen, 0.01);
107   dUnit = 8.reciprocal;
108
109   patterns = measureLen.collect({arg measure;
110     if(sectionData[measure * 4] != nil, {
111       section = sectionData[measure * 4][2];
112       subsection = sectionData[measure * 4][3];
113     });
114     Ppar(
115       //check how amplitude is being handled
116       ensDataFormatted[measure].collect({arg musData, p;
117         Pbind(
118           \instrument, \ens. ++ `hash,
119           \freq, Pseq(musData[0].replace(0, Rest(0))),
120           \dur, Pseq(musData[1] * dUnit),
121           \sustain, Pseq(musData[2] * dUnit),
122           \amp, [1, 0.7, 0.5, 0.3][p],
123           \ampBus, preampBusses[0].index,
124           \bus, postampBusses[p].index,
125           \rel, Pseq(musData[6])
126         )
127       }) ++
128       //check how amplitude and attack are being handled
129       accompDataFormatted[measure].collect({arg musData;
130         Pbind(
131           \instrument, \accomp. ++ `hash,
132           \freq, Pseq(musData[0].replace(0, Rest(0))),
133           \dur, Pseq(musData[1] * dUnit),
134           \sustain, Pseq(musData[2] * dUnit),
135           \amp, Pseq(musData[3].collect({arg item; [0, 2, 4, 8][item]} * 0.0125 * 1),
136           \ampBus, preampBusses[0].index,
137           \bout, Pseq(musData[4].collect({arg index; accompBusses[index].index})),
138           \rel, Pseq(musData[5])
139         )
140       }) ++
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 };
160 [patterns, bassData, ampData]
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
169         patterns[2].collect({arg pattern, p;
170             Pmono(\amp_curve. ++ `hash,
171                 \amp, Pseq(Pseq(pattern.slice(nil, 0) [(startMeasure * 16)..], 1), 1 * `dUnit, \cub),
172                 \dur, 1 * `dUnit,
173                 \bus, preampBusses[p].index
174             )
175         }) ++
176         patterns[1].collect({arg pattern, p;
177             Pmono(\bass_mono. ++ `hash,
178                 \freq, Pseq(pattern.slice(nil, 0) [(startMeasure * 16)..], 1),
179                 \dur, 1 * `dUnit,
180                 \ampBus, preampBusses[p + 1].index,
181                 \bus, postampBusses[4].index
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.start", "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 ++ "\" \\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.start_delimiter.engraver\"}\n<<\n\" ++
66   lyStaffDef.value(name, nameShort, nameMidi) ++
67   <<<\n{ \" +
68   "\n\\set Score.markFormatter = #format-mark-box-numbers \" +
69   "\ntempo 2 = 60\n\" +
70   "\numericTimeSignature \\time 2/2\n\" +
71   "\clef \" ++ clef ++ \"\n\" ++ lyStr + \"\\fermata\" +
72   \">>> \\bar |.\" \n\" \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     "\\\weak Accidental.color #(rgb-color 0.6 0.6 0.6) \n\" ++
81     "\\\weak 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[2][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[1][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
133 // barline and ossia definition
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 if((beat % (16 * 3)) == 0, {break = "\\pageBreak"});
146 if(beat != 0, {"\\n>>>\\n" + barline + break}, {""}) + "\\n<<\\n" + ossia + "{"
147 };
148
149 // add tie
150 lyTie = {"-"};
151
152 lyNoteName = {arg freq, spellingPref = \sharps;
153   if(freq != -1, {
154     lyNoteNameStr[spellingPref][((freq.cpsmidi).round(1) % 12)] ++
155     lyOctStr[(((freq).cpsmidi).round(1) / 12).asInteger - 2];
156   }, {"x"});
157 };
158
159 lyCentDev = {arg freq, padding = true;
160   var centDev;
161   centDev = ((freq.cpsmidi - (freq.cpsmidi).round(1)) * 100).round(1).asInteger;
162   "\\markup { " ++ if(padding, {"\\pad-markup #0.2 \\\""}, {"\\\""}) ++
163   if(centDev >= 0, {"+"}, {"-"}) ++ centDev.asString ++ "\\\"";
164 };
165
166 lyFreqRatio = {arg freqRatioMult, ref, padding = true, lower = 3, attachedToNote = true;
167   var res, ratio;
168   res = "\\markup { " + if(attachedToNote, {""}, {"\\normal-size"}) +
169   "\\lower # " ++ lower + if(padding, {"\\pad-markup #0.2 \\\""}, {"\\\""});
170   ratio = "\\\" + freqRatioMult[0].product.asInteger ++ "/" ++ freqRatioMult[1].product.asInteger ++ "\\\"";
171   res = if(ref != nil,
172     {
173       res ++ "\\concat{ \\\" + [nil, "III", "II", "I"][ref] ++ "\\normal-size-super " ++ ratio ++ "\\\"";
174     }, {
175       res ++ ratio
176     }
177   );
178   if(attachedToNote, {"-" ++ res}, {res})
179 };
180
181
182 lyNote = {arg freq, noteLength, freqRatioMult, ref, spellingPref = \sharps, addMarkup = true, frHide = false, padding = true;
183   lyNoteName.value(freq, spellingPref) ++
184   lyDur.value(noteLength) ++
185   if(addMarkup, {
186     "<MARKUP" ++
187     lyCentDev.value(freq, padding) ++
188     if(frHide, {""}, {lyFreqRatio.value(freqRatioMult, ref, padding)}) ++
189     "MARKUP>"
190   }, {""})
191 };
192
193 lyDur = {arg noteLength;
194   switch(noteLength, 1, {"16"}, 2, {"8"}, 3, {"8."}, 4, {"4"});
195 };
196
197 lyBeamOpen = {"["};
198 lyBeamClosed = {"]"};
199
200 consolidateNotes = {arg lyStr, part;
201   var noteRegex, markupRegex, fullNoteRegex, restRegex, fullRestRegex, res;
202   noteRegex = "(?<>[a-g])(?es|is)?(?[.'])+24)";
203   markupRegex = if(part != 0, {"<MARKUP.{75,85}MARKUP>?"}, {"<MARKUP.{75,115}MARKUP>?"});
204   fullNoteRegex = noteRegex + markupRegex ++ "(?\\h+\\|\\h+\\|k<n>?";
205   restRegex = "(?<>4)";
206   fullRestRegex = "(?<>4)(?\\h+\\|\\k<r>?";
207   res = lyStr;
208   [6, 4, 3, 2].do({arg len;
209     [fullNoteRegex, fullRestRegex].do({arg regex;
210       res.findRegexp(regex ++ "{ " ++ (len-1) ++ "}") .clump(3).do({arg match;
211         var word, note, markup, lyDur;
212         word = match[0][1];
213         note = match[1][1];
214         markup = match[2][1];
215         lyDur = switch(len, 6, {"1."}, 4, {"1"}, 3, {"2."}, 2, {"2"});
216         res = res.replace(word, note.replace("4", lyDur) ++ markup));
217       });
218     });
219   res.replace("<MARKUP", "").replace("MARKUP>", "");
220 };
221
222
223 transcribe = {arg rawMusicData, sectionData, seed;
224   var basePath, scoreFile, musicData, insData, insNames, insNamesShort, insMidi, insClef;
225
226   basePath = "dir +/" + ".." +/" lilypond" +/" seed." + seed;
227   basePath.mkdir;
228   (basePath +/" includes").mkdir;
229
230   scoreFile = File(basePath +/" tkam.score.ly".standardizePath,"w");
231   scoreFile.write(File.readAllString(basePath +/" .." +/" template" +/" tkam.score.template.ly").replace("seed: xxx", "seed: " + seed));
232   scoreFile.close;
233
234   musicData = formatMusicData.value(rawMusicData);
235
236   insData = [
237     {"*", "-", "clarinet", "\\treble_8\\\""},
238     {"III", "III", "clarinet", "bass"},
239     {"II", "II", "clarinet", "alto"},
240     {"I", "I", "clarinet", "treble"}
241   ];
242
243   insNames = insData.slice(nil, 0);
244   insNamesShort = insData.slice(nil, 1);
245   insMidi = insData.slice(nil, 2);
246   insClef = insData.slice(nil, 3);
247
248   musicData.do({arg part, p;
249     var lyFile, lyStr, lastMusAtom, measureCount, spellingPref,
250     tmpSectionData, pcRoot, partLookup, quality;
251
252     //create file
253     lyFile = File(basePath +/" includes" +/" part-" + [star, "III", "II", "I"][p] ++ ".ly".standardizePath,"w");
254
255     //start lypond directives
256     lyStr = "";
257     lastMusAtom = nil;
258     measureCount = 0;
259     spellingPref = \sharps;
260     tmpSectionData = nil;
261     part.clump(4).do({arg beat, i;
262       var gSum;
263       gSum = 0;

```

```

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

```

## 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
50  buildMetronome = {arg win;
51    var clock, metronome, layout;
52
53    clock = StaticText(win).string(" 1.1").font(Font("Liberation Mono", 200));
54    metronome = StaticText(win).string([-30, -105, -104].collect({arg int; int.asAscii})
55      .as(String)).font(Font("Liberation Mono", 300)).stringColor(Color.red);
56

```

```

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(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(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, countOff, 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", "acomp.II", "acomp.I", "click"];
258     var trackNames = ["*", "III", "II", "I", "acomp.II", "acomp.I", "click"];
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];
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     }).defer, '/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 ++ "};";
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("~/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("tkam: 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                         mixerButton = Button().states-([["mixer", Color.white, Color.grey], ["mixer", Color.black]]).action-({
354                             tabButtonReset.value; mixerButton.value = 0; tabs.index = 1 }.inEnvir).value(1), stretch: 1
355                     ]
356                 ),
357                 helpButton = Button().states-([["help", Color.white, Color.grey], ["help", Color.black]]).action-({
358                     tabButtonReset.value; helpButton.value = 0; tabs.index = 2 }.inEnvir).value(1)
359             ),
360             tabs = StackLayout(masterView, faderView, helpView);
361         );
362     );

```

## tkam\_score\_template.ly

```

1 \version "2.19.83"
2
3 #(define (override-color-for-all-grobs color)
4   (lambda (context)
5     (let loop ((x all-grob-descriptions))
6       (if (not (null? x))
7         (let ((grob-name (caar x)))
8           (ly:context-pushpop-property context grob-name 'color color)
9           (loop (cdr x))))))
10
11 #(define-markup-command (relMark layout props mus) (ly:music?)
12   #:properties ((size -2))
13   (interpret-markup layout props
14     #f
15     \markup {
16       \score {
17         \new Staff { $mus }
18         \layout {
19           \context {
20             \Staff
21             \remove Time-signature-engraver
22             fontSize = #-2
23             \hide Stem
24             \override TextScript.outside-staff-priority = ##f
25             \override StaffSymbol.staff-space = #(magstep -2)
26             \override StaffSymbol.thickness = #(magstep -2)
27             \override TextScript.self-alignment-X = #-0.4
28             \override TextScript.staff-padding = #1
29           }
30           \context {
31             \Score
32             proportionalNotationDuration = #(ly:make-moment 1/16)
33             \remove "Separating-line-group-engraver"
34             \override SpacingSpanner.strict-note-spacing = ##t
35             \override RehearsalMark.self-alignment-X = #-1
36             \override RehearsalMark.Y-offset = #10
37             \override RehearsalMark.X-offset = #10
38           }
39           \context {
40             \Voice
41             \consists "Horizontal-bracket-engraver"
42             \override HorizontalBracket.direction = #UP
43           }
44           indent = 0
45           line-width = 4\cm
46         }
47       }
48     }
49   #f))
50
51
52 \paper {
53   #(set-paper-size "a4" 'portrait)
54   top-margin = 1\cm
55   bottom-margin = 1\cm
56   left-margin = 2\cm
57   ragged-bottom = ##t
58
59   top-system-spacing =
60   #'((basic-distance . 15 )
61     (minimum-distance . 15 )
62     (padding . 0 )
63     (stretchability . 0))

```

```

64
65 system-system-spacing =
66 #'((basic-distance . 35 )
67 (minimum-distance . 35 )
68 (padding . 0 )
69 (stretchability . 0))
70
71 last-bottom-spacing =
72 #'((basic-distance . 10 )
73 (minimum-distance . 10 )
74 (padding . 0 )
75 (stretchability . 0))
76
77 %systems-per-page = 3
78 first-page-number = 1
79 print-first-page-number = ##t
80
81 print-page-number = ##t
82 oddHeaderMarkup = \markup { \fill-line { \line { \on-the-fly #not-first-page {\pad-markup #2 { \concat {\italic {"to kill a monarch "}} (seed: xxx)}}}}}
83 evenHeaderMarkup = \markup { \fill-line { \line { \on-the-fly #not-first-page {\pad-markup #2 { \concat {\italic {"to kill a monarch "}} (seed: xxx)}}}}}
84 oddFooterMarkup = \markup { \fill-line {
85   \concat {
86     " "
87     \fontsize #1.5
88     \on-the-fly #print-page-number-check-first
89     \fromproperty #'page:page-number-string
90     " "}}}
91 evenFooterMarkup = \markup { \fill-line {
92   \concat {
93     " "
94     \fontsize #1.5
95     \on-the-fly #print-page-number-check-first
96     \fromproperty #'page:page-number-string
97     " "}}}
98 }
99
100 \header {
101   title = \markup { \italic {to kill a monarch}}
102   composer = \markup \right-column {"michael winter" "(berlin, germany; 2021)"}
103   poet = "seed: xxx"
104   tagline = ""
105 }
106
107 #(set-global-staff-size 11)
108
109 \layout {
110   indent = 0.0\cm
111   line-width = 17.5\cm
112   ragged-last = ##f
113   ragged-right = ##f
114
115   \context {
116     \Score
117     \override BarNumber.stencil = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
118     \override Stem.stemlet-length = #0.75
119     proportionalNotationDuration = #(ly:make-moment 1/16)
120     \remove "Separating-line.group-engraver"
121     \override RehearsalMark.self-alignment-X = #-1
122     \override RehearsalMark.Y-offset = #10
123     \override RehearsalMark.X-offset = #-8
124     %\override RehearsalMark.outside-staff-priority = #0
125   }
126   \context {
127     \Staff
128
129     \override VerticalAxisGroup.staff-staff-spacing =
130       #'((basic-distance . 20 )
131         (minimum-distance . 20 )
132         (padding . 0 )
133         (stretchability . 0))
134
135     \override VerticalAxisGroup.default-staff-staff-spacing =
136       #'((basic-distance . 20 )
137         (minimum-distance . 20 )
138         (padding . 0 )
139         (stretchability . 0))
140     \override TextScript.staff-padding = #2
141     \override TextScript.self-alignment-X = #0
142   }
143   \context {
144     \StaffGroup
145     \name "SemiStaffGroup"
146     \consists "SpanBar-engraver"
147     \override SpanBar.stencil =
148       #(lambda (grob)
149         (if (string=? (ly:grob-property grob 'glyph-name) "|")
150             (set! (ly:grob-property grob 'glyph-name) ""))
151         (ly:span-bar::print grob)))
152   }
153   \context {
154     \Score
155     \accepts SemiStaffGroup
156   }
157 }
158
159 \midi { }
160
161
162 \score{
163 \new Score
164 <<
165 \new SemiStaffGroup {
166 <<
167 \include "includes/part.I.ly"
168 \include "includes/part.II.ly"
169 \include "includes/part.III.ly"
170 >>
171 }
172 \include "includes/part.star.ly"
173 >>
174
175 \layout{}
176 \midi{}
177 }

```