

UNIVERSITY OF MIAMI

*AN INTRODUCTION TO MODELING COMPOSITION  
THROUGH ABJAD'S MODEL OF MUSIC NOTATION*

BY  
GREGORY ROWLAND EVANS

A THESIS

SUBMITTED TO THE FACULTY  
OF THE UNIVERSITY OF MIAMI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR  
THE DEGREE OF MASTER OF MUSIC

CORAL GABLES, FLORIDA  
MAY 2019

© 2019 - GREGORY ROWLAND EVANS  
ALL RIGHTS RESERVED.

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of  
the requirements for the degree of  
Master of Music

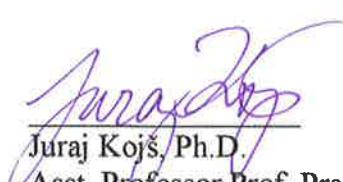
COMPUTATIONAL NOTATIONS

Gregory Rowland Evans

Approved:



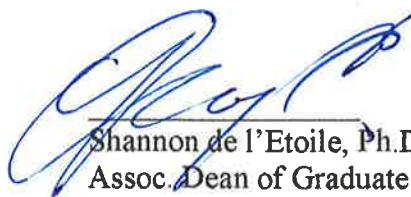
Charles Mason, D.M.A.  
Professor  
Chair of Theory and Composition



Juraj Kojs, Ph.D.  
Asst. Professor Prof. Practice  
Dir. of Electronic Music Studios



William Pirkle, M.S.  
Assoc. Professor  
Dir. of Music Engineering Technology



Shannon de l'Etoile, Ph.D.  
Assoc. Dean of Graduate Studies

# *An Introduction to Modeling Composition through Abjad's Model of Music Notation*

## ABSTRACT

In my recent music, I have begun to make extensive use of the Abjad API for Formalized Score Control in the Python programming language to produce music notation that is illustrated via the Lilypond engraving engine. In the research that led to this paper, I sought to find avenues for computationally modeling the act of composition with Abjad's model of music notation. In order to be best equipped to compose scores with these tools, the user should have a basic understanding of Python and Lilypond. The first chapter of this paper outlines various fundamentals in these environments. The second chapter discusses the underlying methodology behind the use of Abjad as a tool for both music composition and music engraving. After grasping the basic functionality within Abjad's notational model, it became clear to me that further software could be written to increase the efficiency of the use of Abjad, as well as to model my own idiosyncratic compositional workflow. The third and final chapters consists of appendices of my own tools, written in Python, along with source code and scores of music I have composed with the concurrent use of Python, Abjad, and Lilypond as a demonstration of my own compositional process and the power that these programming paradigms afford the composer. The tools I have written are a work in progress and my future research will consist of improvements to their functionality and to the order of operations of my compositional process in order to compose with the least redundant code possible.

**THIS THESIS IS DEDICATED TO ALL COMPOSERS LOOKING TO FORMALIZE THEIR MUSIC WITHOUT  
THE PAIN OF ARTHRITIS.**

# Acknowledgments

SPECIAL THANKS, to my Mother, Father, and Brother for listening to my speeches. And to Ivan, Jeff, Josiah, and Trevor, without whom none of this would have been possible. Special thanks is also extended to Charles Mason, the advisor of this thesis, for his patient encouragement and enthusiasm throughout the writing process.

# Contents

BLANK PAGE

COVER IMAGE xii

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>SOME PREREQUISITE KNOWLEDGE OF LILY POND AND PYTHON</b>         | <b>1</b>  |
| 1.1      | Lilypond . . . . .   | 2         |
| 1.1.1    | Comparision with Other Software . . . . .                          | 2         |
| 1.1.2    | Lilypond's Model of Musical Rhythm . . . . .                       | 3         |
| 1.1.3    | Context Concatenation . . . . .                                    | 4         |
| 1.2      | Python . . . . .   | 5         |
| 1.2.1    | Lists . . . . .  | 5         |
| 1.2.1.1  | Slicing . . . . .  | 5         |
| 1.2.1.2  | List Comprehensions . . . . .                                      | 6         |
| 1.2.2    | Dictionaries . . . . .   | 8         |
| 1.2.2.1  | Dictionary Comprehensions . . . . .                                | 9         |
| 1.2.3    | Modelling Objects . . . . .  | 9         |
| <b>2</b> | <b>MY COMPOSITIONAL PRACTICE WITH ABJAD, LILY POND, AND PYTHON</b> | <b>13</b> |
| 2.1      | Methodology . . . . .  | 13        |
| 2.1.1    | The Usefulness of Abjad for me as a Composer . . . . .             | 16        |
| 2.1.2    | Automating Potentially Tedious Tasks . . . . .                     | 17        |
| 2.1.2.1  | Creating Notes . . . . .   | 17        |
| 2.1.2.2  | Dynamics, Articulations, and Hairpins . . . . .                    | 22        |
| 2.1.2.3  | Using <i>abjad.BowContactPoint()</i> . . . . .                     | 25        |
| 2.1.2.4  | Stylesheets . . . . .  | 40        |
| 2.1.2.5  | Composing with Models and Algorithms . . . . .                     | 45        |

|         |  |     |
|---------|--|-----|
| 2.1.3   | The Need to Build Tools for a More Personalized Approach to Music-Making . . . | 52  |
| 2.1.3.1 | Building Tools . . . . .   | 52  |
| 2.1.3.2 | abjad-ext . . . . .  | 53  |
| 2.1.3.3 | rmakers . . . . .  | 53  |
| 2.1.4   | Other Packages . . . . .   | 54  |
| 2.1.5   | MusicMaker . . . . .   | 54  |
| 2.1.6   | AttachmentHandlers . . . . .   | 61  |
| 2.2     | Editing Source Code . . . . .  | 77  |
| 2.2.1   | Clef.py . . . . .  | 78  |
| 2.2.2   | Articulation.py . . . . .  | 78  |
| 2.2.3   | Microtonal Expansion in Abjad 2.2.1 . . . . .                                  | 78  |
| 2.2.3.1 | Process . . . . .  | 79  |
| 2.2.3.2 | File Systems and Alterations . . . . .   | 80  |
| 2.3     | Conclusion . . . . .   | 80  |
|         | BLANK PAGE   | 82  |
| A       | APPENDIX OF SOURCE CODE  | 83  |
| A.1     | AttachmentHandlers . . . . .   | 83  |
| A.1.1   | ArticulationHandler . . . . .  | 83  |
| A.1.2   | ClefHandler . . . . .  | 84  |
| A.1.3   | DynamicHandler . . . . .   | 86  |
| A.1.4   | GlissandoHandler . . . . .   | 87  |
| A.1.5   | NoteheadHandler . . . . .  | 88  |
| A.1.6   | PitchHandler . . . . .   | 89  |
| A.1.7   | SlurHandler . . . . .  | 90  |
| A.1.8   | TrillHandler . . . . .   | 91  |
| A.2     | Cchar (for two cellos) Source Code . . . . .                                   | 92  |
| A.2.1   | Segment . . . . .  | 92  |
| A.2.1.1 | Segment_I . . . . .  | 92  |
| A.2.2   | Stylesheet . . . . .   | 108 |
| A.3     | Tianshu (for 12 players) Source Code . . . . .                                 | 115 |
| A.3.1   | Segments . . . . .   | 115 |
| A.3.1.1 | Segment_I . . . . .  | 115 |
| A.3.1.2 | Segment_II . . . . .   | 149 |
| A.3.1.3 | Segment_III . . . . .  | 184 |
| A.3.1.4 | Segment_IV . . . . .   | 218 |

|  |            |
|--|------------|
| A.3.2 Stylesheet . . . . .   | 253        |
| A.4 Four Ages of Sand (for Flute, Alto Saxophone, and Violoncello) Source Code . . . . . | 259        |
| A.4.1 Segments . . . . .   | 259        |
| A.4.1.1 Segment_I . . . . .  | 259        |
| A.4.1.2 Segment_II . . . . .   | 270        |
| A.4.1.3 Segment_III . . . . .  | 282        |
| A.4.1.4 Segment_IV . . . . .   | 294        |
| A.4.2 Stylesheet . . . . .   | 305        |
| <b>B APPENDIX OF SCORES</b>  | <b>311</b> |
| B.1 Scores . . . . .   | 311        |
| B.1.1 Cthar (for two cellos) Score . . . . .   | 311        |
| B.1.2 Tianshu (for 12 players) Score . . . . .   | 327        |
| B.1.3 Four Ages of Sand (for flute, alto saxophone, and violoncello) Score . . . . .     | 405        |
| <b>REFERENCES</b>  | <b>415</b> |

# List of figures

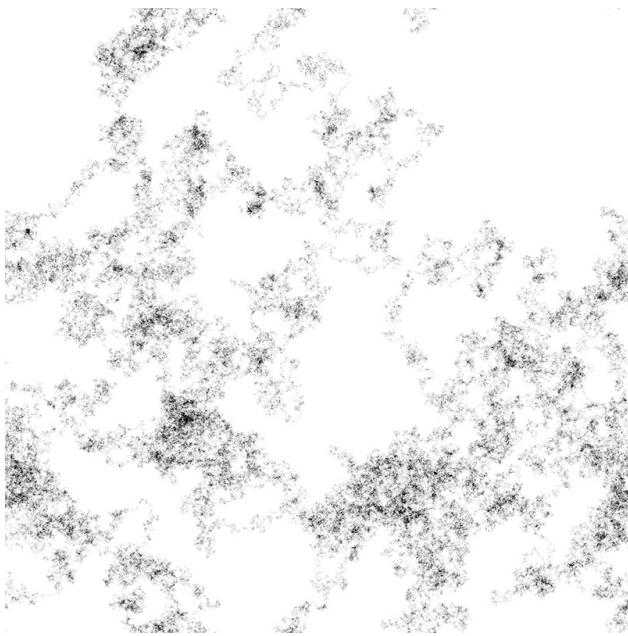
|  |     |
|--|-----|
| o.0.1 A 2-dimensional random walk of 100 iterations by Paul Bourke. . . . .              | xii |
| 2.1.1 A default note. . . . .  | 18  |
| 2.1.2 A note with the user-input duration value of (1, 8) and pitch value of 11. . . . . | 19  |
| 2.1.3 A staff with notes of varying pitch and duration. . . . .                          | 19  |
| 2.1.4 A staff with many notes. . . . .   | 21  |
| 2.1.5 Notes with dynamics. . . . .   | 23  |
| 2.1.6 More notes with dynamics. . . . .  | 24  |
| 2.1.7 Notes with algorithmic dynamics. . . . .   | 25  |
| 2.1.8 Bow tablature. . . . .   | 27  |
| 2.1.9 Extended bow tablature. . . . .  | 31  |
| 2.1.10 Very long bow tablature. . . . .  | 34  |
| 2.1.11 Extremely long bow tablature. . . . .   | 40  |
| 2.1.12 A mapping of a random walk. . . . .   | 47  |
| 2.1.13 A mensuration canon. . . . .  | 52  |
| 2.1.14 Notation from MusicMaker with two rmakers and PitchHandlers. . . . .              | 61  |
| 2.1.15 Trills. . . . .   | 63  |
| 2.1.16 Four voice demonstration of AttachmentHandlers. . . . .                           | 77  |
| 2.2.1 ekmelicStyle evans scale. . . . .  | 79  |
| 2.2.2 An eighth tone random walk. . . . .  | 80  |
| B.1.1 A 2-dimensional random walk of 1000 iterations by Paul Bourke. . . . .             | 416 |

# List of Code Examples

|      |  |    |
|------|--|----|
| 1.1  | Lilypond syntax . . . . .  | 4  |
| 1.2  | Printing an item of a list through indexing . . . . .            | 5  |
| 1.3  | Printing an item of a list through indexing: RESULT . . . . .    | 6  |
| 1.4  | Printing items of a list through slicing . . . . .               | 6  |
| 1.5  | Printing items of a list through slicing: RESULT . . . . .       | 6  |
| 1.6  | Inserting elements into a list through slicing . . . . .         | 6  |
| 1.7  | Inserting elements into a list through slicing: RESULT . . . . . | 6  |
| 1.8  | Creating a list with a list comprehension . . . . .              | 7  |
| 1.9  | Creating a list with a list comprehension: RESULT . . . . .      | 7  |
| 1.10 | Acting upon elements in a list comprehension . . . . .           | 7  |
| 1.11 | Acting upon elements in a list comprehension: RESULT . . . . .   | 7  |
| 1.12 | Rewriting a list comprehension as a “for loop” . . . . .         | 7  |
| 1.13 | Appending elements to a list . . . . .                           | 7  |
| 1.14 | Appending elements to a list: RESULT . . . . .                   | 7  |
| 1.15 | Extending a list with elements . . . . .                         | 8  |
| 1.16 | Extending a list with elements: RESULT . . . . .                 | 8  |
| 1.17 | Printing elements from a dictionary . . . . .                    | 8  |
| 1.18 | Printing elements from a dictionary: RESULT . . . . .            | 8  |
| 1.19 | Printing elements from a dictionary: ERROR . . . . .             | 8  |
| 1.20 | Printing elements from a dictionary: CORRECTION . . . . .        | 8  |
| 1.21 | Making a dictionary comprehension . . . . .                      | 9  |
| 1.22 | Making a dictionary comprehension: RESULT . . . . .              | 9  |
| 1.23 | Creating an empty class in python . . . . .                      | 9  |
| 1.24 | Adding attributes to classes . . . . .                           | 9  |
| 1.25 | Defining attributes in classes . . . . .                         | 10 |
| 1.26 | Creating a subclass . . . . .                                    | 10 |
| 1.27 | Adding methods to a subclass . . . . .                           | 10 |

|      |   |    |
|------|---|----|
| 1.28 | Creating more subclasses . . . . .  | 10 |
| 1.29 | Instantiating objects with attribute values . . . . .                               | 10 |
| 1.30 | Interacting with objects . . . . .  | 11 |
| 1.31 | Interacting with objects: RESULT . . . . .  | 11 |
| 2.1  | Import statement format . . . . .   | 17 |
| 2.2  | Format for object instantiation . . . . .   | 17 |
| 2.3  | Showing an <i>abjad.Note()</i> object . . . . .                                     | 17 |
| 2.4  | Showing an instance of an <i>abjad.Note()</i> object: RESULT . . . . .              | 17 |
| 2.5  | Altering default values in an instance <i>abjad.Note()</i> object . . . . .         | 18 |
| 2.6  | Altering default values in an instance <i>abjad.Note()</i> object: RESULT . . . . . | 18 |
| 2.7  | Populating a staff with notes . . . . .   | 19 |
| 2.8  | Populating a staff with notes: RESULT . . . . .                                     | 19 |
| 2.9  | Faster note creation . . . . .  | 20 |
| 2.10 | Tuple zip: RESULT . . . . .   | 20 |
| 2.11 | Duration list comprehension . . . . .   | 20 |
| 2.12 | Duration list comprehension: RESULT . . . . .                                       | 20 |
| 2.13 | Pitch and duration zip: RESULT . . . . .  | 20 |
| 2.14 | Note object list comprehension: RESULT . . . . .                                    | 21 |
| 2.15 | Showing the staff . . . . .   | 21 |
| 2.16 | Showing the staff: RESULT . . . . .   | 21 |
| 2.17 | Attaching dynamics . . . . .  | 22 |
| 2.18 | Attaching dynamics: RESULT . . . . .  | 22 |
| 2.19 | Attaching more dynamics . . . . .   | 22 |
| 2.20 | Attaching more dynamics: RESULT . . . . .   | 23 |
| 2.21 | Attaching dynamics through an algorithm . . . . .                                   | 24 |
| 2.22 | Attaching dynamics through an algorithm: RESULT . . . . .                           | 24 |
| 2.23 | Bow tablature . . . . .   | 26 |
| 2.24 | Bow tablature: RESULT . . . . .   | 26 |
| 2.25 | Extended bow tablature . . . . .  | 28 |
| 2.26 | Extended bow tablature: RESULT . . . . .  | 28 |
| 2.27 | Very long bow tablature . . . . .   | 31 |
| 2.28 | Very long bow tablature: RESULT . . . . .   | 31 |
| 2.29 | Extremely long bow tablature . . . . .  | 34 |
| 2.30 | Extremely long bow tablature: RESULT . . . . .                                      | 34 |
| 2.31 | Cthar stylesheet . . . . .  | 40 |
| 2.32 | Random walk . . . . .   | 45 |

|      |   |     |
|------|---|-----|
| 2.33 | Random walk: RESULT                         | 45  |
| 2.34 | Mensuration canon                           | 47  |
| 2.35 | Mensuration canon: RESULT                   | 49  |
| 2.36 | MusicMaker source                           | 55  |
| 2.37 | Using MusicMaker with PitchHandler          | 57  |
| 2.38 | Using MusicMaker with PitchHandler: RESULT  | 60  |
| 2.39 | Using TrillHandler                          | 61  |
| 2.40 | Using TrillHandler: RESULT                  | 62  |
| 2.41 | Demonstration of AttachmentHandlers         | 63  |
| 2.42 | Demonstration of AttachmentHandlers: RESULT | 70  |
| A.1  | ArticulationHandler                         | 83  |
| A.2  | ClefHandler                                 | 84  |
| A.3  | DynamicHandler                              | 86  |
| A.4  | GlissandoHandler                            | 87  |
| A.5  | NoteheadHandler                             | 88  |
| A.6  | PitchHandler                                | 89  |
| A.7  | SlurHandler                                 | 90  |
| A.8  | TrillHandler                                | 91  |
| A.9  | Cchar Segment_I                             | 92  |
| A.10 | Cchar Stylesheet                            | 108 |
| A.11 | Tianshu Segment_I                           | 115 |
| A.12 | Tianshu Segment_II                          | 149 |
| A.13 | Tianshu Segment_III                         | 184 |
| A.14 | Tianshu Segment_IV                          | 218 |
| A.15 | Tianshu Stylesheet                          | 253 |
| A.16 | Four Ages of Sand Segment_I                 | 259 |
| A.17 | Four Ages of Sand Segment_II                | 270 |
| A.18 | Four Ages of Sand Segment_III               | 282 |
| A.19 | Four Ages of Sand Segment_IV                | 294 |
| A.20 | Four Ages of Sand Stylesheet                | 305 |



**Figure 0.0.1:** A 2-dimensional random walk of 100 iterations by Paul Bourke.

*The artist-conceptor will have to be knowledgeable and inventive in such varied domains as mathematics, logic, physics, chemistry, biology, genetics, paleontology (for the evolution of forms), the human sciences and history; in short, a sort of universality, but one based upon, guided by and oriented toward forms and architectures.*

(1985, *Arts/Sciences: Alloys* p.3)

Iannis Xenakis

# 1

## Some Prerequisite Knowledge of Lilypond and Python

IN MY EXPERIENCE composing scores with the help of computational systems, I have found that the Abjad Application Programming Interface<sup>1</sup> for formalized score control provides the greatest power and flexibility. Abjad is significant because of the freedom with which it provides composers with the ability to manipulate their musical material and the ability to control not only the musical elements of a score, but also other graphic features as well. Every score that is created with Abjad is engraved by the Lilypond music notation engine.<sup>2</sup> Because of this interdependence, the composer should become familiar with Lilypond's model of music notation as well as elements of Lilypond's syntax. Since Abjad is an API in the Python programming language,<sup>3</sup> it is essential that the composer be familiar with writing Python code. In this chapter, some basic concerns about Lilypond and Python will be discussed, while information directly related to the Abjad API follows in chapter two.

---

<sup>1</sup><http://abjad.mbrsi.org/>

<sup>2</sup><http://lilypond.org/>

<sup>3</sup><https://www.python.org/>

## 1.1 LILYPOND

### 1.1.1 COMPARISION WITH OTHER SOFTWARE

MOST MODERN COMPOSERS will be familiar with the plethora of options available for digital music engraving. The purpose of this paper is not to delve into the history of modern engraving practices, but it is important to note that, by far, the most popular engraving software available for the consumer market today are Finale<sup>4</sup> and Sibelius,<sup>5</sup> with a few new robust programs like Dorico<sup>6</sup> beginning to appear. These systems, packed with many features, are suitable for the majority of composers' needs. They allow composers to be able to engrave pitches and rhythms in traditional Western notation and provide a number of formatting options that expand upon these traditions, allowing the user to create professional-quality documents, but it is not insignificant to note that many composers find the musical model of these programs to be overly restrictive upon musical creativity. As an example, with most of the common engraving software, users often must click through several menus to engrave a tuplet other than a triplet, especially if that tuplet does not contain successive rhythms of the same duration. Programs such as Dorico and the most recent versions of Finale have supplemented some of these issues through keyboard shortcuts<sup>7</sup> and opened a clearer accessibility to the engraving of inected tuplets, but otherwise it is clear: these programs are tailored to a specific set of engraving requirements. This software is made for people engraving fairly traditional music like transcriptions, orchestrations, film scores, and so on, which do not typically make extensive use of this kind of notation.

While the programs are flexible and can be used for other means, I decided that what I require from a musical score is significantly restricted by the software. It becomes tedious to write music with many tuplets or other graphical oddities. While some composers have written their own engraving programs, like NoteAbility Pro,<sup>8</sup> which can handle a number of contemporary techniques with ease, other

---

<sup>4</sup><https://www.finalemusic.com/>

<sup>5</sup><https://www.avid.com/sibelius>

<sup>6</sup><https://www.dorico.com/>

<sup>7</sup>Most notation software also allows the user to define their own keyboard shortcuts.

<sup>8</sup><http://debussy.music.ubc.ca/NoteAbility/>

composers have resorted to simply composing in graphic editors and drawing-oriented software, which brings the act of engraving much closer to the act of handwriting a piece; but even with these paradigm shifts, few notation engines show any friendliness to structural formalization. Finale and Sibelius have features that allow the user to program certain procedures,<sup>9</sup> but these are limited. Programs like Patch Work (and its kin Patch Work Graphic Language<sup>10</sup>) and OpenMusic<sup>11</sup> were created in order to supplement this limitation. These programs allow the composer to manipulate data to represent musical elements which are then engraved within the program in the case of PWGL. In the case of OpenMusic, the elements are exported as MusicXML,<sup>12</sup> to be engraved by another software; but the complex MusicXML files produced by these programs are not always stable and often produce fallacious results or completely fail to convert.<sup>13</sup> The combination of Abjad and Lilypond surmounts many of these concerns. Abjad simply writes text files of Lilypond code, which removes the concern of file transfer errors, and Lilypond represents each element of a score, music, text, or graphic, in a syntax that is simple and consistent across a number of engraving complexities allowing the composer to engrave almost anything as part of the score.<sup>14</sup>

### 1.1.2 LILYPOND'S MODEL OF MUSICAL RHYTHM

Another important aspect of Lilypond is its modelling of rhythmic content. Lilypond makes a distinction, unlike other notation engines, between written and prolated durations. In programs like OpenMusic, a set of triplet eighth notes would be written as durations of  $\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$ ,<sup>15</sup> but in Lilypond they would be written as  $\frac{1}{8}, \frac{1}{8}, \frac{1}{8}$  prolated by a duration of  $\frac{3}{2}$ . This means that traditional rhythmic notation like whole notes through

---

<sup>9</sup>I have mostly seen plugins related to layout spacing and harmonic analysis, but it is my understanding there are more capabilities available.

<sup>10</sup><http://www2.siba.fi/PWGL/>

<sup>11</sup><http://repmus.ircam.fr/openmusic/home>

<sup>12</sup><https://www.musicxml.com/>

<sup>13</sup>Incidentally, my first serious attempt to learn Abjad stems exactly from the fact that neither Finale 2012 nor Dorico 1 could convert MusicXML files I had created in OpenMusic 6.12. OMLily was the salvation of this music, but engraving features of the score other than pitch and rhythm proved tedious.

<sup>14</sup>This is possible because Lilypond inherently has no GUI and writing text files is the intended user interaction with the program. It is also possible because Lilypond is open source. Abjad could potentially be reworked to engrave music with a different engine, but Lilypond was and still is the most feasible option.

<sup>15</sup>Often, composers use OpenMusic's RTM syntax which is comprised of LISP-like nested lists like the following, which does not model rhythm in the way described in this paragraph: (1 (3 (2 (1 2 -1 1)) 3))

sixty-fourth notes and beyond are written as usual, but are prolated by a surrounding tuplet bracket with a given duration. Abjad follows, as much as possible, the same conventions as Lilypond's notational model.

### 1.1.3 CONTEXT CONCATENATION

Lilypond also has a feature referred to as "Context Concatenation." A context in Lilypond can be thought of as a staff with various features and formats associated with it. When given a name, a context is able to be appended to another context with the same name, as long as the files share the same score structure.<sup>16</sup>

This allows the composer to write various sections of a piece in isolation and to stitch them together into a final document as a secondary process. Users should note a similarity between Lilypond syntax and L<sup>A</sup>T<sub>E</sub>X<sup>17</sup> syntax, both of which share conceptual similarities with HTML<sup>18</sup> code. The following is a simple example of Lilypond syntax containing an unusual tuplet:<sup>19</sup>

```

1 \version "2.19.82"
2 \language "english"
3 \score {
4     \new Staff
5     { \times 2/9{
6         c'8
7         cs'8
8         d'4.
9         ef'8
10        \times 3/2 {
11            e'8
12            f'8      }
13        }
14        fs'8
15        g'8
16        af'4
17        a'4    }
18 }
```

**Code Example 1.1:** Lilypond syntax

---

<sup>16</sup>i.e. instrumentation as well as other invisible contexts

<sup>17</sup><https://www.latex-project.org/>

<sup>18</sup><https://www.w3.org/html/>

<sup>19</sup>Note that it should be clear from this example that it is no more difficult to engrave this unusual tuplet than a regular triplet of straight eighth notes.

## 1.2 PYTHON

IN PYTHON, there are a number of data types. Some of the important data types to address when discussing Abjad are integers, floating-point decimals, booleans, variables, strings, lists, tuples, and dictionaries. Each of these data types have specific features and behave in characteristic ways. Both integers and floating-point decimals, often called floats, are numbers. Integers can be used to signify numeric value in whole numbers while floats offer a more refined gradation of values. Variables are names that are assigned to other values or processes. With variables, users are able to refer to elements throughout a file without rewriting the information many times by hand.

### 1.2.1 LISTS

#### 1.2.1.1 SLICING

An important process to comprehend when composing with Abjad is that of list manipulation.<sup>20</sup> There are many processes that can be performed on and with lists. The concept of slicing will be discussed first. Readers vaguely familiar with Python may recognize the format  $[x : y]$  when referring to slicing a list. In Python, the programmer can refer to items within a list via their indices. The index is the location within a list where an item exists. These indices begin at zero. An example set of indices is  $[0, 1, 2, 3, 4, 5]$ ,<sup>21</sup> but the Python slices  $[x : y]$  do not refer to items, even though indices do refer to items. The indicators within a slice actually refer to the spaces between items. It is possible to test this principle as follows:<sup>22</sup>

```
1 >>> letters = ['a', 'b', 'c', 'd', 'e', 'f']
2 >>> print(letters[2])
```

**Code Example 1.2:** Printing an item of a list through indexing

Which results in:

---

<sup>20</sup>In fact, most of the work composers do when using Abjad involves storing and manipulating data in lists and dictionaries. Most elements of the score end up in a list at some point.

<sup>21</sup>It is also possible to use a negative index. The first element of a list is still index 0, but the final element of the list is -1.

<sup>22</sup>This explanation comes from an email sent by Trevor Bača to the Abjad mailing list.

```
1 c
```

**Code Example 1.3:** Printing an item of a list through indexing: RESULT

but:

```
1 >>> letters = ['a', 'b', 'c', 'd', 'e', 'f']
2 >>> print(letters[0:2])
```

**Code Example 1.4:** Printing items of a list through slicing

results in:

```
1 a
2 b
```

**Code Example 1.5:** Printing items of a list through slicing: RESULT

The following example presents a logical pitfall:

```
1 >>> letters = ['a', 'b', 'c', 'd', 'e', 'f']
2 >>> letters[-1:1] = 'xyz'
3 >>> print(letters)
```

**Code Example 1.6:** Inserting elements into a list through slicing

Which will result in:

```
1 ['a', 'b', 'c', 'd', 'e', 'x', 'y', 'z', 'f']
```

**Code Example 1.7:** Inserting elements into a list through slicing: RESULT

It is demonstrated here that, in fact, this slicing refers to the continuous space between -1 and 1. The direction proceeds from right to left because the slice was begun with a negative number.

#### 1.2.1.2 LIST COMPREHENSIONS

Another of the many actions that are able to be performed on lists is that of list comprehension. List comprehensions allow the programmer to quickly create lists whose contents follow simple parameters. Consider the built-in Python function *range()*, which allows the user to increment integers up until the user-input point. If Python were asked to print each item within *range(5)*, then 0, 1, 2, 3, and 4 would be written to the terminal. A list comprehension could be written as follows:

```

1 >>> foo = [x for x in range(5)]
2 >>> print(foo)

```

**Code Example 1.8:** Creating a list with a list comprehension

Which will result in:

```

1 [0, 1, 2, 3, 4]

```

**Code Example 1.9:** Creating a list with a list comprehension: RESULT

It is also possible to act upon the elements within this list:

```

1 >>> bar = [x*3 for x in range(5)]
2 >>> print(bar)

```

**Code Example 1.10:** Acting upon elements in a list comprehension

Which will result in:

```

1 [0, 3, 6, 9, 12]

```

**Code Example 1.11:** Acting upon elements in a list comprehension: RESULT

This process can be substituted by a “for loop,” which is useful for more complicated functions, but can be overly verbose for processes better handled by list comprehensions:

```

1 >>> increments = range(5)
2 >>> spam = []
3 >>> for x in increments:
4     ...     spam.append(x*3)
5 >>> print(spam)

```

**Code Example 1.12:** Rewriting a list comprehension as a “for loop”

In this example, the built-in function *append()* is used. It is important to make a distinction between *append()* and *extend()*. This can be illustrated as follows:

```

1 >>> list_1 = [0, 1, 2, 3]
2 >>> list_2 = [4, 5, 6, 7]
3 >>> list_1.append(list_2)
4 >>> print(list_1)

```

**Code Example 1.13:** Appending elements to a list

Which results in:

```

1 [0, 1, 2, 3, [4, 5, 6, 7]]

```

**Code Example 1.14:** Appending elements to a list: RESULT

but:

```

1 >>> list_1 = [0, 1, 2, 3]
2 >>> list_2 = [4, 5, 6, 7]
3 >>> list_1.extend(list_2)
4 >>> print(list_1)
```

**Code Example 1.15:** Extending a list with elements

results in:

```

1 [0, 1, 2, 3, 4, 5, 6, 7]
```

**Code Example 1.16:** Extending a list with elements: RESULT

### 1.2.2 DICTIONARIES

A dictionary is much like a list, but in the case of a dictionary, elements of the list are referred to by keys:

```

1 musician = {'name': 'Greg', 'instrument': 'cello', 'age': 24}
2 print(musician['instrument'])
```

**Code Example 1.17:** Printing elements from a dictionary

resulting in:

```

1 cello
```

**Code Example 1.18:** Printing elements from a dictionary: RESULT

Is not possible to refer to elements in the dictionary from the right side of the key. The following example produces an error:

```

1 musician = {'name': 'Greg', 'instrument': 'cello', 'age': 24}
2 print(musician['cello'])
```

**Code Example 1.19:** Printing elements from a dictionary: ERROR

To make this kind of cross-definition work, the user must add the keys in reverse as follows:

```

1 musician = {'name': 'Greg', 'instrument': 'cello', 'age': 24, 'Greg': 'name',
2   'cello': 'instrument', 24: 'age'}
3 print(musician['cello'])
```

**Code Example 1.20:** Printing elements from a dictionary: CORRECTION

### 1.2.2.1 DICTIONARY COMPREHENSIONS

Dictionary comprehensions are also possible and follow the same structure as list comprehensions:

```

1 keys = ['Name', 'Instrument', 'Age']
2 definitions = ['Greg', 'Cello', 24]
3 musician = {key:definition for key, definition in zip(keys, definitions)}
4 print(musician)

```

**Code Example 1.21:** Making a dictionary comprehension

resulting in:

```

1 {'Name': 'Greg', 'Instrument': 'Cello', 'Age': 24}

```

**Code Example 1.22:** Making a dictionary comprehension: RESULT

### 1.2.3 MODELLING OBJECTS

One of the most attractive features of Abjad is that the system allows for the formalization of structures to control the placement and distribution of dynamics, articulations, and in fact, every visual element of the score. This is because Abjad attempts to model music notation rather than musical phenomenology. It treats all elements in a musical score as an object. An object in programming has various attributes and potential modes of behavior. An example of object modelling can be seen in the creation of animals. A first step is to create a general template on which the animals are based.

```

1 >>> class Animal:
2 ...     def __init__(self):

```

**Code Example 1.23:** Creating an empty class in python

Attributes can be added to the basic animal in the `__init__` section.

```

1 >>> class Animal:
2 ...     def __init__(self, name, color, pattern):

```

**Code Example 1.24:** Adding attributes to classes

In order to retrieve the information that is placed in these attributes, the user must add the following below the `__init__` section:

```

1 >>> class Animal:
2 ...     def __init__(self, name, color, pattern):
3 ...         self.name = name
4 ...         self.color = color
5 ...         self.pattern = pattern

```

**Code Example 1.25:** Defining attributes in classes

Now that an Animal object has been created, the programmer can begin to create individual animal types. One could create many animal objects to represent the menagerie, but a possible intermediate step would be to create a sub-class of the Animal. For instance, one could create a cat based on the general animal by doing the following:

```

1 >>> class Cat(Animal):

```

**Code Example 1.26:** Creating a subclass

This cat has all of the same attributes that the general animal has. It is also possible to write functions to be included only in a specific sub-class:

```

1 >>> class Cat(Animal):
2 ...     def speak(self):
3 ...         print('Purr...')

```

**Code Example 1.27:** Adding methods to a subclass

Likewise, other animals can be created in the same fashion:

```

1 >>> class Dog(Animal):
2 ...     def speak(self):
3 ...         print('Woof...')
4 >>> class Giraffe(Animal):
5 ...     def speak(self):
6 ...         print('...giraffe sounds?')

```

**Code Example 1.28:** Creating more subclasses

Once the programmer has created objects to model different types of animals, specific animals with names, colors, and coat patterns can be defined by creating an instance of the animal objects.

```

1 >>> huckle = Cat('huckle', 'orange', 'tabby')
2 >>> ginger = Dog('ginger', 'tan', 'fluffy')
3 >>> spooks = Cat('spooks', 'grey', 'tabby')
4 >>> geoffrey = Giraffe('geoffrey', 'brown and yellow', 'spotted')

```

**Code Example 1.29:** Instantiating objects with attribute values

These object instances can be queried for certain information:

```

1 >>> print('Huckle is ' + huckle.color)
2 >>> print('Spooks is a ' + spooks.pattern)
3 >>> print('The dog's name is ' + ginger.name)
4 >>> print('Geoffrey is ' + geoffrey.color)
5 >>> huckle.speak()
6 >>> spooks.speak()
7 >>> ginger.speak()
8 >>> geoffrey.speak()
```

**Code Example 1.30:** Interacting with objects

Which results in the following output at the terminal:

```

1 Huckle is orange
2 Spooks is a tabby
3 The dog's name is ginger
4 Geoffrey is brown and yellow
5 Purr...
6 Purr...
7 Woof!
8 ...giraffe sounds?
```

**Code Example 1.31:** Interacting with objects: RESULT

Working with Python quickly becomes very complex, depending on the needs of the programmer, but much can be accomplished with an understanding of Python's data types, lists, dictionaries, and object modelling. In the following chapter, concepts in Lilypond and Python pertaining specifically to Abjad will be introduced in the context of my own use of the software for my compositional process.

This page is intentionally left blank.

*One of the most significant areas of investigation in live instrumental music - that pertaining to the degree of approximation attained in any given realization of a particular type of musical text - offers itself most directly for transference into the computer studio, by reason of the obviously significant extent to which the concept of efficiency (in both human and computer terms) is (at least in real time) self-contradictory.*  
*(1998, Composer-Computer-Active Form par.3)*

Brian Ferneyhough

# 2

## My Compositional Practice With Abjad, Lilypond, and Python

### 2.1 METHODOLOGY

IN THE PRECEDING CHAPTER, I presented some of the strengths and potential weaknesses of Abjad and Lilypond when compared with similar programming paradigms, as well as some potential logical pitfalls when working with these programs. In my recent compositional practice, I have begun to amalgamate a workflow out of the ecosystem of Python, Abjad, and Lilypond, by learning from and embracing the idiosyncrasies of each. The use of these tools in tandem is advantageous for my work due to the flexibility of Lilypond's notational algorithm and Abjad's clarification of Lilypond's model of music notation through Python's object-oriented nature, as well as Python's vast logical and mathematical abilities. Not only are Abjad and Lilypond both full of diverse features, but due to their open source nature, the source code for each is accessible to the user for further modification. Occasionally, I have found the need to tweak Abjad's source code in order for it to perform functions that I desire, but more often than this, the composer will find the need to build tools to simplify the process of engraving.

In my work, I often desire a structural rigor, where rhythms, pitches, and orchestration, among other

parameters are balanced together by a plan or logic that gives meaning to potential musical realities. A rigorous structure tends to fall apart when constructed by hand because humans are prone to err, while computers, conversely, do not make mistakes unless they are taught a false procedure. The computer does not have the ability to produce a logical fallacy unless the error is programmed into its underlying functionality. Because of this, working with the Python programming language allows for a consistency in formal rigor that might be otherwise unattainable by intuition or hand-written calculations and graphs. It also allows for the potential modeling of complex systems and algorithmic music, where human intuition is placed in a more subordinate role to formal design.

Lilypond's ability to draw lines and shapes, and its less restrictive model of notation than other software, allow the composer to have greater graphic freedom than other notation software. Another notable feature of Lilypond is its lack of a GUI,<sup>23</sup> allowing the program more memory power when calculating spacing to avoid collisions, which results in greater visual clarity upon the first engraving of a piece. Also, since it allows the user to include functions in the Scheme programming language,<sup>24</sup> the user is able to affect other features like proportional spacing across an entire score instead of manually clicking and dragging note heads as one would do while using Finale or Sibelius. Lilypond has the ability to manage all visual aspects of a score and can also be used to export image files in the *pdf* and *png* formats, along with *midi* files. Finally, a great feature of Lilypond is its context concatenation ability. As mentioned in the previous chapter, this allows multiple, separate Lilypond files to be combined with one another to stitch together separate segments of a full composition into one document.

An advantage to the Abjad composition paradigm is its ability to manage polyphony. Other programming paradigms like PWGL or OM are a little more restrictive in this regard. Often, in PWGL and OM, using a procedure in one instrument as well as the next requires the user to instantiate a function multiple times and to alter the settings of the function in order for the music to seem continuous.<sup>25</sup> This duplication of processes that were carried out in other voices clutters up the workspace with redundant

---

<sup>23</sup>Graphic User Interface

<sup>24</sup><https://www.scheme.com/>

<sup>25</sup>This issue would be solved with generators, but functions in OpenMusic don't naturally behave in this way. Admittedly, I do not know LISP very well and never wrote my own LISP functions in OM.

information. In Abjad, the two concepts of copying and continuing are very distinct,<sup>26</sup> allowing the composer to specifically use either technique as needed. Since Abjad is an API,<sup>27</sup> in Python, it becomes very easy to cross-reference the same material-generating functions across different voices and at different points in time within the score. This comes from the fact that the music composed with Abjad is written as a text file, allowing composers to create and manipulate any object or function they choose; whereas programs like PWGL and OM are slightly restricted by their GUI. Though there are ways for composers to write their own functions in these programs, they are more difficult to manipulate and it is not entirely obvious to a beginner that it is even possible to do so. Because Abjad has no GUI, it inherently invites the composer to write the source code as part of the act of composition.

Though one could theoretically compose an entire score and only compile the Python file once the score is finalized, Abjad allows for an iterative workflow of composing, compiling, critiquing, and correcting in a cycle that lasts until the composer is satisfied with the composition. The speed of modern computation as opposed to hand written calculation and engraving makes this workflow reasonable.

In Abjad, elements of a score are modeled as Python objects. Some objects, like a note or a rest for instance, have a duration attribute, but a note has an attribute that a rest does not: pitch. All elements of the score are objects with properties and attributes, therefore the entire score is manipulable via Abjad and, by extension, various formal means. This is a feature that is not present in OM and is difficult to achieve in PWGL, as OM does not display articulations or dynamics within the score viewing windows and PWGL's interface is difficult to read.<sup>28</sup> This is, in part, because these programs have different foci and goals. OM is typically used like a calculator for composers to generate options for materials with which to compose and PWGL, while able to export data to other notation engines, is equipped with its own ENP,<sup>29</sup> with which music is rendered. Both OM and PWGL are based on CLOS,<sup>30</sup> but I believe that the legibility of Python scripts as well as the large number of Python programmers makes it a much better candidate

---

<sup>26</sup>This distinction, as mentioned earlier, comes from whether or not the programmer retrieves data from a generator or another reservoir.

<sup>27</sup>Application Programming Interface

<sup>28</sup>Although many composers have had success with PWGL, its user interface has always seemed too cluttered to me and I have not explored it as thoroughly as I have OpenMusic.

<sup>29</sup>Expressive Notation Package

<sup>30</sup>Common LISP Object System

for the user-end of the system allowing for easy transference of knowledge from one user to another. The objects of notational elements are capable of being manipulated, therefore they can be created, connected, and appended to one another throughout the composition process to create a score through composer-written procedures and functions as well as through built-in tools. In the end, the greatest strength of this ecosystem is its flexibility.

In this chapter, I will discuss the compositional advantages of working with these programs such as how to automate potentially tedious tasks, the benefits of an iterative compositional workflow, and the possibilities for composing with algorithms or models. I will also explain some of my own solutions to composing with Abjad, like my *MusicMaker* and *AttachmentHandlers* classes as well as times when I have edited the Abjad source code.

### 2.1.1 THE USEFULNESS OF ABJAD FOR ME AS A COMPOSER

In my recent music, it is typical for me to focus significantly on formal uniformity and continuous, alternating procedures. These procedures might be in relation to the rhythmic, harmonic, textural, or dynamic material. I have also become very interested in a pseudo-tablature style of notation that features these iterative, procedural factors. It became apparent to me that I could leverage the programming concepts of loops and functions to write music very quickly. With this methodology I have written various programs that organize and produce musical material based on my predetermined structures, allowing me to compose material and generate the product of these procedures quickly. In the course of working in this manner, I have begun to appreciate the necessity of externalizing various tools in order to clean up my composition files. These tools, as well as my general compositional templates, could also easily be used by other composers, but they are tailored explicitly to my own compositional needs. Not only do my tools written in Python help me stay consistent with my formal designs, they also allow me to compose music that is specifically organized to my own tendencies and logic, rather than copying another composer's tools and workflow. Although I have benefited greatly from the programs I have written, they are a work in progress and may not necessarily have universal functionality.<sup>31</sup>

---

<sup>31</sup>All code examples in this paper are written in Python 3, Abjad 3.1, and Lilypond 2.19.82.

## 2.1.2 AUTOMATING POTENTIALLY TEDIOUS TASKS

### 2.1.2.1 CREATING NOTES

There are two options for creating and viewing notes with Abjad. One could open up the terminal, or command line, and activate a Python session in order to write the code or alternatively, it is possible to write code in a text file saved with the `.py` suffix and call Python to compile it once the file is completed. The former method is better for quick testing of loops and materials, while the second method is much more sustainable for the process of composing a score, because it allows the programmer to save progress as well as multiple versions of the code along the way. Regardless of which method is chosen, the code is written in the same way. The first step is always to import the Abjad API into the python session or file so that all of Abjad's tools and properties are available. There are several ways of doing this, but the key to clarity is to be consistent. Throughout this chapter I will use this format:

```
1 >>> import abjad
```

**Code Example 2.1:** Import statement format

This tells Python to instantiate tools through the Abjad namespace. Doing this requires that all Abjad objects be prefixed with `abjad.` followed by whatever object or tool is being used. Thus, a note object will look like this:

```
1 >>> abjad.Note()
```

**Code Example 2.2:** Format for object instantiation

This note can be given a variable name with which the user is able to refer to the note throughout the file and `abjad.show()` can be used to quickly produce a `pdf` file of this note:

```
1 >>> import abjad
2 >>> note = abjad.Note()
3 >>> abjad.show(note)
```

**Code Example 2.3:** Showing an `abjad.Note()` object

This Abjad code will produce a Lilypond file containing the following text:

```
1 \version "2.19.82"  %! LilyPondFile
2 \language "english" %! LilyPondFile
3
```

```

4 \header { %! LilyPondFile
5   tagline = ##f
6 } %! LilyPondFile
7
8 \layout {}
9
10 \paper {}
11
12 \score { %! LilyPondFile
13   {
14     c'4
15   }
16 } %! LilyPondFile

```

**Code Example 2.4:** Showing an instance of an *abjad.Note()* object: RESULT

and will produce the image in a pdf file seen in figure 2.1.1:



**Figure 2.1.1:** A default note.

Notice that the note object has various default values associated with it. The note is rendered with a pitch value of middle c and a duration value of one quarter note. Easily enough, these values are manipulable! Instead, the following could have been written:<sup>32</sup>

```

1 >>> import abjad
2 >>> note = abjad.Note(11, abjad.Duration(1, 8))
3 >>> abjad.show(note)

```

**Code Example 2.5:** Altering default values in an instance *abjad.Note()* object

from which the image in figure 2.1.2 and the following Lilypond code would be received:

```

1 \score { %! LilyPondFile
2   {
3     b'8
4   }
5 } %! LilyPondFile

```

**Code Example 2.6:** Altering default values in an instance *abjad.Note()* object: RESULT



**Figure 2.1.2:** A note with the user-input duration value of (1, 8) and pitch value of 11.

The following are a few strategies for making many notes in a row in order to create a piece. First, a staff and notes should be created. Then, the staff will be filled with notes and finally, the staff will be shown.

Here is one way this can be done:

```

1 >>> import abjad
2 >>> note_1 = abjad.Note(0, abjad.Duration(1, 4))
3 >>> note_2 = abjad.Note(1, abjad.Duration(1, 4))
4 >>> note_3 = abjad.Note(2, abjad.Duration(1, 2))
5 >>> notes = [note_1, note_2, note_3]
6 >>> staff = abjad.Staff(notes)
7 >>> abjad.show(staff)

```

**Code Example 2.7:** Populating a staff with notes

from which the user would receive figure 2.1.3 and the following Lilypond code:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     c'4
5     cs'4
6     d'2
7   }
8 } %! LilyPondFile

```

**Code Example 2.8:** Populating a staff with notes: RESULT



**Figure 2.1.3:** A staff with notes of varying pitch and duration.

As one might begin to suspect, this process of note creation can get quite tedious. Here is one possible

---

<sup>32</sup>[http://abjad.mbrsi.org/appendices/pitch\\_conventions.html](http://abjad.mbrsi.org/appendices/pitch_conventions.html)

alternative approach to writing code with Abjad which is more economical for a longer piece:

```
1 >>> import abjad
2 >>> numerators = [1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1]
3 >>> denominators = [4, 4, 2, 8, 8, 4, 16, 16, 16, 16, 16, 16, 16]
4 >>> durations = [abjad.Duration(y, z) for y, z in zip(numerators, denominators)]
5 >>> pitches = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
6 >>> notes = [abjad.Note(x, y) for x, y in zip(pitches, durations)]
7 >>> note_staff = abjad.Staff(notes)
8 >>> abjad.show(note_staff)
```

### Code Example 2.9: Faster note creation

Here, the use of `zip()` can be seen as well as a list comprehension. With `zip()` the programmer creates a list of numerators and denominators organized as tuples to represent fractions:

```
[1] [(1, 4), (1, 4), (1, 2), (1, 8), (1, 8), (1, 4), (1, 16), (3, 16), (1, 16),  
     (1, 16), (1, 16), (1, 16), ]
```

### Code Example 2.10: Tuple zip: RESULT

and with this list comprehension a list of duration objects based on those fractions is returned:

```
1 >>> [abjad.Duration(y, z) for y, z in zip(numerators, denominators)]
```

resulting in:

```
[abjad.Duration((1, 4)), abjad.Duration((1, 4)), abjad.Duration((1, 2)), abjad.  
    Duration((1, 8)), abjad.Duration((1, 8)), abjad.Duration((1, 4)), abjad.  
    Duration((1, 16)), abjad.Duration((3, 16)), abjad.Duration((1, 16)), abjad.  
    Duration((1, 16)), abjad.Duration((1, 16)), abjad.Duration((1, 16))]
```

**Code Example 2.12:** Duration list comprehension: RESULT

Again, two lists are zipped together, these being the list of pitches and the list of durations:

```
[0, abjad.Duration((1, 4))), (1, abjad.Duration((1, 4))), (2, abjad.Duration((1, 2))), (3, abjad.Duration((1, 8))), (4, abjad.Duration((1, 8))), (5, abjad.Duration((1, 4))), (6, abjad.Duration((1, 16))), (7, abjad.Duration((3, 16))), (8, abjad.Duration((1, 16))), (9, abjad.Duration((1, 16))), (10, abjad.Duration((1, 16))), (11, abjad.Duration((1, 16))),]
```

**Code Example 2.13:** Pitch and duration zip: RESULT

and a note object is created for every pitch and duration in this list with a list comprehension:

```

1 [abjad.Note(0, abjad.Duration((1, 4))), abjad.Note(1, abjad.Duration((1, 4))),
  abjad.Note(2, abjad.Duration((1, 2))), abjad.Note(3, abjad.Duration((1,
  8))), abjad.Note(4, abjad.Duration((1, 8))), abjad.Note(5, abjad.Duration
  ((1, 4))), abjad.Note(6, abjad.Duration((1, 16))), abjad.Note(7, abjad.
  Duration((3, 16))), abjad.Note(8, abjad.Duration((1, 16))), abjad.Note(9,
  abjad.Duration((1, 16))), abjad.Note(10, abjad.Duration((1, 16))), abjad.
  Note(11, abjad.Duration((1, 16))),]

```

**Code Example 2.14:** Note object list comprehension: RESULT

this list of notes is placed inside of a staff and the staff is shown.

```

1 >>> note_staff = abjad.Staff(notes)
2 >>> abjad.show(note_staff)

```

**Code Example 2.15:** Showing the staff

From this process, figure 2.1.4 along with the following Lilypond output are produced:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     c'4
5     cs'4
6     d'2
7     ef'8
8     e'8
9     f'4
10    fs'16
11    g'8.
12    af'16
13    a'16
14    bf'16
15    b'16
16  }
17 } %! LilyPondFile

```

**Code Example 2.16:** Showing the staff: RESULT



**Figure 2.1.4:** A staff with many notes.

If this kind of process is extrapolated, one can begin to create loops<sup>33</sup> to handle tasks of every shape and size. Because this process can be arduous at times, Abjad is equipped with a number of tools out of the

---

<sup>33</sup>a “loop,” or “for loop” is the name of a kind of function structure.

box to assist in processes like note creation such as `abjad.LeafMaker()`, `abjad.NoteMaker()`, `abjad.MeasureMaker()`, and `abjad.SegmentMaker()`. While these features are useful and are at the heart of many other tools like the Abjad-ext package `rmakers`, it is important to realize that it is not necessary to rely on these built-in functions to be able to write music with Abjad.

### 2.1.2.2 DYNAMICS, ARTICULATIONS, AND HAIRPINS

Just like the creation of note objects, one can also simplify and formalize the attachment of dynamics:

```

1 >>> import abjad
2 >>> dynamic_staff = abjad.Staff()
3 >>> dynamic_staff.extend(r"c'4 cs'4 d'2")
4 >>> piano = abjad.Dynamic('p')
5 >>> mezzo_forte = abjad.Dynamic('mf')
6 >>> forte = abjad.Dynamic('f')
7 >>> abjad.attach(piano, dynamic_staff[0])
8 >>> abjad.attach(mezzo_forte, dynamic_staff[1])
9 >>> abjad.attach(forte, dynamic_staff[2])
10 >>> abjad.show(dynamic_staff)

```

**Code Example 2.17:** Attaching dynamics

resulting in figure 2.1.6 and the following Lilypond code:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     c'4
5     \p
6     cs'4
7     \mf
8     d'2
9     \f
10   }
11 } %! LilyPondFile

```

**Code Example 2.18:** Attaching dynamics: RESULT

Simplifying this further by making use of a loop to attach the dynamics to each leaf<sup>34</sup> in the staff, dynamic objects can be created and attached at once:

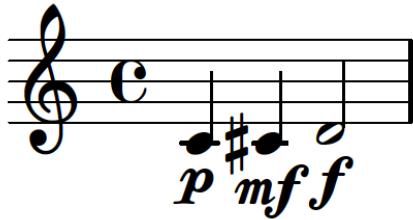
```

1 >>> import abjad
2 >>> new_staff = abjad.Staff()

```

---

<sup>34</sup>[http://abjad.mbrsri.org/core\\_concepts/lcsi.html](http://abjad.mbrsri.org/core_concepts/lcsi.html)



**Figure 2.1.5:** Notes with dynamics.

```

3  >>> new_staff.extend(r"c'4 cs'4 d'2 ef'8 e'8 f'4 fs'16 g'8. af'16 a'16 bf'16 b
4    '16")
5  >>> dynamics = ['niente', 'pppp', 'ppp', 'pp', 'p', 'mp', 'mf', 'f', 'ff', 'fff',
6    'ffff', 'sfz', ]
7  >>> leaves = abjad.select(new_staff).leaves()
8  >>> for leaf, dynamic in zip(leaves, dynamics):
...     abjad.attach(abjad.Dynamic(dynamic), leaf)
>>> abjad.show(new_staff)

```

**Code Example 2.19:** Attaching more dynamics

resulting in the Lilypond code and figure: 2.1.6:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     c'4
5     _ #(make-dynamic-script (markup #:whiteout #:normal-text #:italic "
6     niente"))
7     cs'4
8     \pppp
9     d'2
10    \ppp
11    ef'8
12    \pp
13    e'8
14    \p
15    f'4
16    \mp
17    fs'16
18    \mf
19    g'8.
20    \f
21    af'16
22    \ff
23    a'16
24    \fff
25    bf'16
26    \ffff
27    b'16
28    \sfz
}

```

```
29 } %! LilyPondFile
```

**Code Example 2.20:** Attaching more dynamics: RESULT



**Figure 2.1.6:** More notes with dynamics.

It can be seen that dynamics behave in the same way as other attachable objects. This is also true of articulations and hairpins. In the following example, articulations and hairpins are attached to leaves as well, featuring a possible way to imbue some behavioral qualities into the attachment of these elements.

```
1 >>> import abjad
2 >>> music_staff = abjad.Staff()
3 >>> music_staff.extend(r"c'4 cs'4 d'2 r4 ds'2. e'8 f'8 fs'8 g'8 gs'8 r4. a'1")
4 >>> for run in abjad.select(music_staff).runs():
5 ...     if len(run) > 3:
6 ...         leaves = abjad.select(run).leaves()
7 ...         abjad.attach(abjad.Dynamic('mf'), run[0])
8 ...         for leaf in leaves:
9 ...             abjad.attach(abjad.Articulation('tenuto'), leaf)
10 ...     elif len(run) == 3:
11 ...         abjad.attach(abjad.Dynamic('f'), run[0])
12 ...         abjad.attach(abjad.StartHairpin('>'), run[0])
13 ...         abjad.attach(abjad.Dynamic('mp'), run[-1])
14 ...     elif len(run) == 1:
15 ...         abjad.attach(abjad.Dynamic('ppp'), run[0])
16 >>> abjad.show(music_staff)
```

**Code Example 2.21:** Attaching dynamics through an algorithm

resulting in the Lilypond code and figure 2.1.7:

```
1 \score { %! LilyPondFile
2     \new Staff
3     {
4         c'4
5         \f
6         \>
7         cs'4
8         d'2
9         \mp
```

```

10      r4
11      e '2
12      \mf
13      - \tenuto
14      f '8
15      - \tenuto
16      g '8
17      - \tenuto
18      a ''8
19      - \tenuto
20      b ''8
21      - \tenuto
22      c ''8
23      - \tenuto
24      r4
25      c ''2.
26      \ppp
27  }
28 } %! LilyPondFile

```

**Code Example 2.22:** Attaching dynamics through an algorithm: RESULT



**Figure 2.1.7:** Notes with algorithmic dynamics.

This loop analyzes the length of each run in the staff and chooses what dynamics and articulations to attach based on the result. This is an extremely powerful method for attaching indicators throughout a score. Next, I will introduce a procedure to handle the *abjad.BowContactPoint()* object, which produces a more complex Lilypond result and graphic.<sup>35</sup>

#### 2.1.2.3 USING ABJAD.BOWCONTACTPOINT()

The *abjad.BowContactPoint()* object and an accompanying factory function, *abjad.bow\_contact\_spanner()*, are tools that are able to annotate a staff of notes with fractions intended to

---

<sup>35</sup>For clean and legible notation, users will want to edit the Lilypond context in which this notation occurs to remove clefs and staff lines. An example of this can be seen in the following section on *Stylesheets*. In order to avoid confusion, examples featuring the *abjad.BowContactPoint* tool are engraved with the default staff context.

represent points along the length of a bow.<sup>36</sup> Native in these tools is the ability to calculate whether one fraction is greater or lesser than its surrounding fractions and attach an “up-bow” or “down-bow” marking as needed. Because of this feature, I created a file in Abjad 2.2.1 which I called *abjad.StringContactSpanner* which eliminated the bow markings in order for it to be used universally for any potential parameter. This file was adapted by Trevor Bača into Abjad 3.1’s *abjad.BowContactPoint()* which features an optional keyword to include or exclude these bowings. Here is a possible way to use these tools:

```

1 >>> import abjad
2 >>> bow_staff = abjad.Staff()
3 >>> bow_staff.extend(r"c'4 c'4 c'4 c'4")
4 >>> indicator_1 = abjad.BowContactPoint((3, 3))
5 >>> indicator_2 = abjad.BowContactPoint((2, 3))
6 >>> indicator_3 = abjad.BowContactPoint((1, 3))
7 >>> indicator_4 = abjad.BowContactPoint((0, 3))
8 >>> abjad.attach(indicator_1, bow_staff[0])
9 >>> abjad.attach(indicator_2, bow_staff[1])
10 >>> abjad.attach(indicator_3, bow_staff[2])
11 >>> abjad.attach(indicator_4, bow_staff[3])
12 >>> abjad.bow_contact_spacer(bow_staff, omit_bow_changes=True)
13 >>> abjad.show(bow_staff)

```

### Code Example 2.23: Bow tablature

resulting in the Lilypond code:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     \tweak Y-offset #2.0
5     \tweak stencil #ly:text-interface::print
6     \tweak text \markup {
7       \center-align
8       \vcenter
9       \fraction
10      1
11      1
12    }
13    c'4
14    \glissando
15    \tweak Y-offset #0.6666666666666666
16    \tweak stencil #ly:text-interface::print
17    \tweak text \markup {
18      \center-align
19      \vcenter
20      \fraction
21      2
22

```

---

<sup>36</sup>It makes no difference what pitches are in the staff because the note heads are removed by the tool.

```

2.2          3
2.3      }
2.4  c'4
2.5  \glissando
2.6  \tweak Y-offset #-0.6666666666666666
2.7  \tweak stencil #ly:text-interface::print
2.8  \tweak text \markup {
2.9      \center-align
3.0      \vcenter
3.1      \fraction
3.2          1
3.3          3
3.4  }
3.5  c'4
3.6  \glissando
3.7  \tweak Y-offset #-2.0
3.8  \tweak stencil #ly:text-interface::print
3.9  \tweak text \markup {
4.0      \center-align
4.1      \vcenter
4.2      \fraction
4.3          0
4.4          1
4.5  }
4.6  c'4
4.7  }
4.8 } %! LilyPondFile

```

**Code Example 2.24:** Bow tablature: RESULT

and figure 2.1.8:



**Figure 2.1.8:** Bow tablature.

In the resultant Lilypond code are several, lengthy `\tweak` commands. Composing a score in Lilypond where an instrument has two staves, one of which is a bowing tablature that uses notation similar to what is produced by the `abjad.BowContactPoint()` tool would be even more tedious to write than the note creation process above, making this tool quite useful for speeding up the engraving process. The following examples are a few alternative methods that achieve this kind of notation in a similar manner of reduction as in the note creation examples:

```

1 >>> import abjad
2 >>> new_bow_staff = abjad.Staff()
3 >>> new_bow_staff.extend(r"c'4 c'4 c'2 c'8 c'8 c'4 c'16 c'8. c'16 c'16 c'16 c'16")
4 >>> indicator_1 = abjad.BowContactPoint((3, 3))
5 >>> indicator_2 = abjad.BowContactPoint((2, 3))
6 >>> indicator_3 = abjad.BowContactPoint((1, 3))
7 >>> indicator_4 = abjad.BowContactPoint((0, 3))
8 >>> indicator_5 = abjad.BowContactPoint((2, 3))
9 >>> indicator_6 = abjad.BowContactPoint((1, 3))
10 >>> indicator_7 = abjad.BowContactPoint((3, 3))
11 >>> indicator_8 = abjad.BowContactPoint((0, 3))
12 >>> indicator_9 = abjad.BowContactPoint((1, 3))
13 >>> indicator_10 = abjad.BowContactPoint((2, 3))
14 >>> indicator_11 = abjad.BowContactPoint((3, 3))
15 >>> indicator_12 = abjad.BowContactPoint((0, 3))
16 >>> indicators = [indicator_1, indicator_2, indicator_3, indicator_4,
17     indicator_5,
18     indicator_6, indicator_7, indicator_8, indicator_9, indicator_10,
19     indicator_11, indicator_12, ]
20 >>> leaves = abjad.select(new_bow_staff).leaves()
21 >>> for leaf, indicator in zip(leaves, indicators):
22     abjad.attach(indicator, leaf)
22 >>> abjad.bow_contact_spacer(new_bow_staff, omit_bow_changes=True)

```

**Code Example 2.25:** Extended bow tablature

resulting in the Lilypond code:

```

1 \score { %! LilyPondFile
2     \new Staff
3     {
4         \tweak Y-offset #2.0
5         \tweak stencil #ly:text-interface::print
6         \tweak text \markup {
7             \center-align
8                 \vcenter
9                     \fraction
10                         1
11                         1
12         }
13         c'4
14         \glissando
15         \tweak Y-offset #0.6666666666666666
16         \tweak stencil #ly:text-interface::print
17         \tweak text \markup {
18             \center-align
19                 \vcenter
20                     \fraction
21                         2
22                         3
23         }
24         c'4
25         \glissando

```

```

26      \tweak Y-offset #-0.6666666666666666
27      \tweak stencil #ly:text-interface::print
28      \tweak text \markup {
29          \center-align
30          \vcenter
31          \fraction
32              1
33              3
34      }
35      c'2
36      \glissando
37      \tweak Y-offset #-2.0
38      \tweak stencil #ly:text-interface::print
39      \tweak text \markup {
40          \center-align
41          \vcenter
42          \fraction
43              0
44              1
45      }
46      c'8
47      \glissando
48      \tweak Y-offset #0.6666666666666666
49      \tweak stencil #ly:text-interface::print
50      \tweak text \markup {
51          \center-align
52          \vcenter
53          \fraction
54              2
55              3
56      }
57      c'8
58      \glissando
59      \tweak Y-offset #-0.6666666666666666
60      \tweak stencil #ly:text-interface::print
61      \tweak text \markup {
62          \center-align
63          \vcenter
64          \fraction
65              1
66              3
67      }
68      c'4
69      \glissando
70      \tweak Y-offset #2.0
71      \tweak stencil #ly:text-interface::print
72      \tweak text \markup {
73          \center-align
74          \vcenter
75          \fraction
76              1
77              1
78      }
79      c'16

```

```

80      \glissando
81      \tweak Y-offset #-2.0
82      \tweak stencil #ly:text-interface::print
83      \tweak text \markup {
84          \center-align
85          \vcenter
86          \fraction
87              0
88              1
89      }
90      c'8.
91      \glissando
92      \tweak Y-offset #‐0.6666666666666666
93      \tweak stencil #ly:text-interface::print
94      \tweak text \markup {
95          \center-align
96          \vcenter
97          \fraction
98              1
99              3
100     }
101     c'16
102     \glissando
103     \tweak Y-offset #0.6666666666666666
104     \tweak stencil #ly:text-interface::print
105     \tweak text \markup {
106         \center-align
107         \vcenter
108         \fraction
109             2
110             3
111     }
112     c'16
113     \glissando
114     \tweak Y-offset #2.0
115     \tweak stencil #ly:text-interface::print
116     \tweak text \markup {
117         \center-align
118         \vcenter
119         \fraction
120             1
121             1
122     }
123     c'16
124     \glissando
125     \tweak Y-offset #-2.0
126     \tweak stencil #ly:text-interface::print
127     \tweak text \markup {
128         \center-align
129         \vcenter
130         \fraction
131             0
132             1
133     }

```

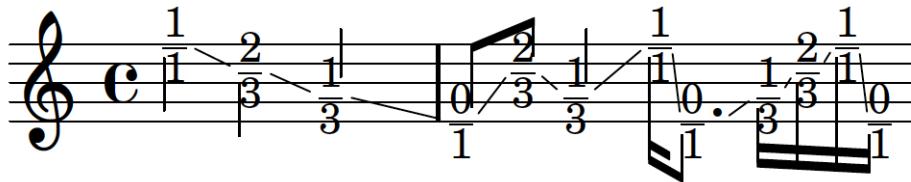
```

134     c'16
135 }
136 } %! LilyPondFile

```

**Code Example 2.26:** Extended bow tablature: RESULT

and figure 2.1.9:



**Figure 2.1.9:** Extended bow tablature.

This example is very similar to the previous example, except for the fact that the process of attaching indicators to leaves has been streamlined. Here is another possibility:

```

1 >>> import abjad
2 >>> new_bow_staff = abjad.Staff()
3 >>> new_bow_staff.extend(r"c'4 c'4 c'4")
4 >>> numerators = [3, 2, 1, 0, 1, 2, 3, 2, 1, 3, 0, 1, ]
5 >>> indicators = [(abjad.BowContactPoint((numerator, 3))) for numerator in
6 numerators]
7 >>> leaves = abjad.select(new_bow_staff).leaves()
8 >>> for leaf, indicator in zip(leaves, indicators):
9 ...     abjad.attach(indicator, leaf)
10 >>> abjad.bow_contact_spacer(new_bow_staff, omit_bow_changes=True)
11 >>> abjad.show(new_bow_staff)

```

**Code Example 2.27:** Very long bow tablature

resulting in the Lilypond code and figure 2.1.10:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     \tweak Y-offset #2.0
5     \tweak stencil #ly:text-interface::print
6     \tweak text \markup {
7       \center-align
8         \vcenter
9           \fraction
10             1
11             1
12       }
13     c'4

```

```

14      \glissando
15      \tweak Y-offset #0.6666666666666666
16      \tweak stencil #ly:text-interface::print
17      \tweak text \markup {
18          \center-align
19          \vcenter
20              \fraction
21                  2
22                  3
23      }
24      c'4
25      \glissando
26      \tweak Y-offset #−0.6666666666666666
27      \tweak stencil #ly:text-interface::print
28      \tweak text \markup {
29          \center-align
30          \vcenter
31              \fraction
32                  1
33                  3
34      }
35      c'4
36      \glissando
37      \tweak Y-offset #−2.0
38      \tweak stencil #ly:text-interface::print
39      \tweak text \markup {
40          \center-align
41          \vcenter
42              \fraction
43                  0
44                  1
45      }
46      c'4
47      \glissando
48      \tweak Y-offset #−0.6666666666666666
49      \tweak stencil #ly:text-interface::print
50      \tweak text \markup {
51          \center-align
52          \vcenter
53              \fraction
54                  1
55                  3
56      }
57      c'4
58      \glissando
59      \tweak Y-offset #0.6666666666666666
60      \tweak stencil #ly:text-interface::print
61      \tweak text \markup {
62          \center-align
63          \vcenter
64              \fraction
65                  2
66                  3
67      }

```

```

68      c '4
69      \glissando
70      \tweak Y-offset #2.0
71      \tweak stencil #ly:text-interface::print
72      \tweak text \markup {
73          \center-align
74          \vcenter
75          \fraction
76              1
77              1
78      }
79      c '4
80      \glissando
81      \tweak Y-offset #0.6666666666666666
82      \tweak stencil #ly:text-interface::print
83      \tweak text \markup {
84          \center-align
85          \vcenter
86          \fraction
87              2
88              3
89      }
90      c '4
91      \glissando
92      \tweak Y-offset #-0.6666666666666666
93      \tweak stencil #ly:text-interface::print
94      \tweak text \markup {
95          \center-align
96          \vcenter
97          \fraction
98              1
99              3
100     }
101     c '4
102     \glissando
103     \tweak Y-offset #2.0
104     \tweak stencil #ly:text-interface::print
105     \tweak text \markup {
106         \center-align
107         \vcenter
108         \fraction
109             1
110             1
111     }
112     c '4
113     \glissando
114     \tweak Y-offset #-2.0
115     \tweak stencil #ly:text-interface::print
116     \tweak text \markup {
117         \center-align
118         \vcenter
119         \fraction
120             0
121             1

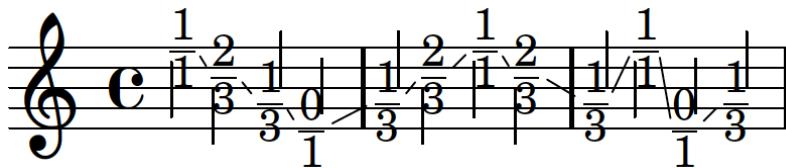
```

```

122     }
123     c'4
124     \glissando
125     \tweak Y-offset #-0.6666666666666666
126     \tweak stencil #ly:text-interface::print
127     \tweak text \markup {
128       \center-align
129       \vcenter
130       \fraction
131         1
132         3
133     }
134   c'4
135 }
136 } %! LilyPondFile

```

**Code Example 2.28:** Very long bow tablature: RESULT



**Figure 2.1.10:** Very long bow tablature.

Here is a further simplification. In this code, the fractions in the indicators are summarized in a list comprehension. If this process is simplified even further it is possible to write code like this:

```

1 >>> import abjad
2 >>> newer_bow_staff = abjad.Staff()
3 >>> newer_bow_staff.extend(r"c'8 c'8 ...")
4 >>> ...
5 >>> leaves = abjad.select(newer_bow_staff).leaves()
6 >>> indicator_numerators = [3, 2, 1, 2, 1, 0, 3, 2, 0, 3, 1, 0, 0, 3, 2, 3, 2,
7 >>>     1, 1, 0, 3, ...]
8 >>> for leaf, numerator in zip(leaves, indicator_numerators):
9 >>>     abjad.attach(abjad.BowContactPoint((numerator, 3)), leaf)
10 >>> abjad.bow_contact_spacer(newer_bow_staff, omit_bow_changes=True)
11 >>> abjad.show(newer_bow_staff)

```

**Code Example 2.29:** Extremely long bow tablature

resulting in the Lilypond code and figure 2.1.11:

```

1 \score { %! LilyPondFile

```

```
2 \new Staff
3 {
4     \tweak Y-offset #2.0
5     \tweak stencil #ly:text-interface::print
6     \tweak text \markup {
7         \center-align
8         \vcenter
9             \fraction
10            1
11            1
12     }
13     c'8
14     \glissando
15     \tweak Y-offset #0.6666666666666666
16     \tweak stencil #ly:text-interface::print
17     \tweak text \markup {
18         \center-align
19         \vcenter
20             \fraction
21                2
22                3
23     }
24     c'8
25     \glissando
26     \tweak Y-offset #-0.6666666666666666
27     \tweak stencil #ly:text-interface::print
28     \tweak text \markup {
29         \center-align
30         \vcenter
31             \fraction
32               1
33               3
34     }
35     c'8
36     \glissando
37     \tweak Y-offset #0.6666666666666666
38     \tweak stencil #ly:text-interface::print
39     \tweak text \markup {
40         \center-align
41         \vcenter
42             \fraction
43               2
44               3
45     }
46     c'8
47     \glissando
48     \tweak Y-offset #-0.6666666666666666
49     \tweak stencil #ly:text-interface::print
50     \tweak text \markup {
51         \center-align
52         \vcenter
53             \fraction
54               1
55               3
```

```

56      }
57 c'8
58 \glissando
59 \tweak Y-offset #-2.0
60 \tweak stencil #ly:text-interface::print
61 \tweak text \markup {
62   \center-align
63   \vcenter
64     \fraction
65       0
66       1
67   }
68 c'8
69 \glissando
70 \tweak Y-offset #2.0
71 \tweak stencil #ly:text-interface::print
72 \tweak text \markup {
73   \center-align
74   \vcenter
75     \fraction
76       1
77       1
78   }
79 c'8
80 \glissando
81 \tweak Y-offset #0.6666666666666666
82 \tweak stencil #ly:text-interface::print
83 \tweak text \markup {
84   \center-align
85   \vcenter
86     \fraction
87       2
88       3
89   }
90 c'8
91 \glissando
92 \tweak Y-offset #-2.0
93 \tweak stencil #ly:text-interface::print
94 \tweak text \markup {
95   \center-align
96   \vcenter
97     \fraction
98       0
99       1
100  }
101 c'8
102 \glissando
103 \tweak Y-offset #2.0
104 \tweak stencil #ly:text-interface::print
105 \tweak text \markup {
106   \center-align
107   \vcenter
108     \fraction
109       1

```

```

110          1
111      }
112 c'8
113 \glissando
114 \tweak Y-offset #-0.6666666666666666
115 \tweak stencil #ly:text-interface::print
116 \tweak text \markup {
117     \center-align
118     \vcenter
119         \fraction
120             1
121             3
122     }
123 c'8
124 \glissando
125 \tweak Y-offset #-2.0
126 \tweak stencil #ly:text-interface::print
127 \tweak text \markup {
128     \center-align
129     \vcenter
130         \fraction
131             0
132             1
133     }
134 c'8
135 \glissando
136 \tweak Y-offset #-2.0
137 \tweak stencil #ly:text-interface::print
138 \tweak text \markup {
139     \center-align
140     \vcenter
141         \fraction
142             0
143             1
144     }
145 c'8
146 \glissando
147 \tweak Y-offset #2.0
148 \tweak stencil #ly:text-interface::print
149 \tweak text \markup {
150     \center-align
151     \vcenter
152         \fraction
153             1
154             1
155     }
156 c'8
157 \glissando
158 \tweak Y-offset #0.6666666666666666
159 \tweak stencil #ly:text-interface::print
160 \tweak text \markup {
161     \center-align
162     \vcenter
163         \fraction

```

```

164          2
165          3
166      }
167 c'8
168 \glissando
169 \tweak Y-offset #2.0
170 \tweak stencil #ly:text-interface::print
171 \tweak text \markup {
172     \center-align
173         \vcenter
174             \fraction
175                 1
176                 1
177     }
178 c'8
179 \glissando
180 \tweak Y-offset #0.6666666666666666
181 \tweak stencil #ly:text-interface::print
182 \tweak text \markup {
183     \center-align
184         \vcenter
185             \fraction
186                 2
187                 3
188     }
189 c'8
190 \glissando
191 \tweak Y-offset #-0.6666666666666666
192 \tweak stencil #ly:text-interface::print
193 \tweak text \markup {
194     \center-align
195         \vcenter
196             \fraction
197                 1
198                 3
199     }
200 c'8
201 \glissando
202 \tweak Y-offset #-0.6666666666666666
203 \tweak stencil #ly:text-interface::print
204 \tweak text \markup {
205     \center-align
206         \vcenter
207             \fraction
208                 1
209                 3
210     }
211 c'8
212 \glissando
213 \tweak Y-offset #-2.0
214 \tweak stencil #ly:text-interface::print
215 \tweak text \markup {
216     \center-align
217         \vcenter

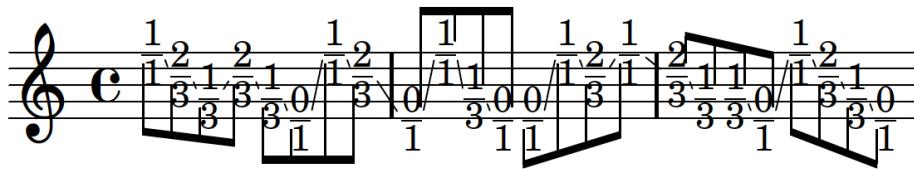
```

```

218          \fraction
219              0
220              1
221      }
222      c'8
223      \glissando
224      \tweak Y-offset #2.0
225      \tweak stencil #ly:text-interface::print
226      \tweak text \markup {
227          \center-align
228          \vcenter
229              \fraction
230                  1
231                  1
232      }
233      c'8
234      \glissando
235      \tweak Y-offset #0.6666666666666666
236      \tweak stencil #ly:text-interface::print
237      \tweak text \markup {
238          \center-align
239          \vcenter
240              \fraction
241                  2
242                  3
243      }
244      c'8
245      \glissando
246      \tweak Y-offset #-0.6666666666666666
247      \tweak stencil #ly:text-interface::print
248      \tweak text \markup {
249          \center-align
250          \vcenter
251              \fraction
252                  1
253                  3
254      }
255      c'8
256      \glissando
257      \tweak Y-offset #-2.0
258      \tweak stencil #ly:text-interface::print
259      \tweak text \markup {
260          \center-align
261          \vcenter
262              \fraction
263                  0
264                  1
265      }
266      c'8
267  }
268 } %! LilyPondFile

```

**Code Example 2.30:** Extremely long bow tablature: RESULT



**Figure 2.1.11:** Extremely long bow tablature.

This version of the code is comprised of the least number of lines. Both the fractions and the attachment processes are summarized in a loop that takes the fractions, puts them in an indicator and attaches those indicators each to a leaf of the staff. Notice that each version of the code results in the same kind of output and each option simplifies the process. Extensive use of *abjad.BowContactPoint()* can be found in the compositions *Armilla*<sup>37</sup> by Josiah Wolf Oberholtzer and *Cthar* by Gregory Rowland Evans.

#### 2.1.2.4 STYLESHEETS

An important concept when working with Lilypond is the idea of a stylesheet. Typically, the beginning of each Lilypond file will be full of information telling Lilypond how to format and render the music within the file. To make use of Lilypond's context concatenation ability, it is best to externalize this information into a file called a stylesheet. An *\include* statement is used to let Lilypond know where to find this information. The stylesheet is written in Lilypond syntax and occasionally Scheme code and may feature information about horizontal spacing proportional to the duration of notes, vertical spacing in staff groups, the removal of time signatures within staves, and the creation of a new context for displaying those time signatures above the staff group. This is also where information about font, font size, paper size and orientation, and header information is stored. The following is the stylesheet that I wrote for my cello duo Cthar:

```

1 % 2018-07-17 19:54
2
3 \version "2.19.82"
4 \language "english"
5 #(set-default-paper-size "letterlandscape")
6 #(set-global-staff-size 10)
7 \include "ekmel.lily"
8 \ekmelicStyle evans

```

<sup>37</sup><https://github.com/josiah-wolf-oberholtzer/armilla>

```

9
10 \header {
11   tagline = ##f
12   breakbefore = ##t
13   title = \markup \override #'(
14     font-name . "Didot"
15     ) \fontsize #15 \bold \center-column {
16       "Cthar"
17     }
18   subtitle = \markup \override #'(
19     font-name . "Didot"
20     ) \fontsize #4 \center-column {
21       "for two cellos"
22     }
23   arranger = \markup \override #'(
24     font-name . "Didot"
25     ) \fontsize #2.5 {
26       "Gregory Rowland Evans"
27     }
28 }

29
30 bowtab = {
31   \override Staff.Clef.stencil = #ly:text-interface::print
32   \override Staff.Clef.text = \markup { \general-align #Y #0.03
33   \epsfile #Y #10 #"bow_position_tablature.eps"
34 }
35 }

36
37 \layout {
38   \accidentalStyle forget
39   indent = #5
40   ragged-right = ##t
41   \context {
42     \name TimeSignatureContext
43     \type Engraver_group
44     \numericTimeSignature
45     \consists Axis_group_engraver
46     \consists Bar_number_engraver
47     \consists Time_signature_engraver
48     \consists Mark_engraver
49     \consists Metronome_mark_engraver
50     \override BarNumber.Y-extent = #'(0 . 0)
51     \override BarNumber.Y-offset = 0
52     \override BarNumber.extra-offset = #'(-4 . 0)
53     %\override BarNumber.font-name = "Didot"
54     \override BarNumber.stencil = #(
55       make-stencil-boxer 0.1 0.7 ly:text-interface::print
56     )
57     \override BarNumber.font-size = 1
58     \override BarNumber.padding = 4
59     \override MetronomeMark.X-extent = #'(0 . 0)
60     \override MetronomeMark.Y-extent = #'(0 . 0)
61     \override MetronomeMark.break-align-symbols = #'(left-edge)
62     \override MetronomeMark.extra-offset = #'(0 . 4)

```

```

63 \override MetronomeMark.font-size = 10
64 \override RehearsalMark.stencil = #(
65     make-stencil-circler 0.1 0.7 ly:text-interface::print
66 )
67 \override RehearsalMark.X-extent = #'(0 . 0)
68 \override RehearsalMark.X-offset = 6
69 \override RehearsalMark.Y-offset = -2.25
70 \override RehearsalMark.break-align-symbols = #'(time-signature)
71 \override RehearsalMark.break-visibility = #end-of-line-invisible
72 \override RehearsalMark.font-name = "Didot"
73 \override RehearsalMark.font-size = 8
74 \override RehearsalMark.outside-staff-priority = 500
75 \override RehearsalMark.self-alignment-X = #center
76     \override TimeSignature.X-extent = #'(0 . 0)
77     \override TimeSignature.X-offset = #ly:self-alignment-interface::x-
aligned-on-self
78     \override TimeSignature.Y-extent = #'(0 . 0)
79 \override TimeSignature.Y-offset = 3
80     \override TimeSignature.break-align-symbol = ##f
81     \override TimeSignature.break-visibility = #end-of-line-invisible
82     \override TimeSignature.font-size = #7
83     \override TimeSignature.self-alignment-X = #center
84     \override VerticalAxisGroup.default-staff-staff-spacing = #'(
85         (basic-distance . 0)
86         (minimum-distance . 10)
87         (padding . 6)
88         (stretchability . 0)
89     )
90 }
91 \context {
92     \Score
93     \remove Bar_number_engraver
94     \remove Mark_engraver
95     \accepts TimeSignatureContext
96     \accepts LipStaff
97     \override BarLine.bar-extent = #'(-2 . 2)
98     \override Beam.breakable = ##t
99     \override Beam.concaveness = #10000
100    \override Glissando.breakable = ##t
101    \override MetronomeMark.font-size = 5
102    \override SpacingSpanner.strict-grace-spacing = ##t
103    \override SpacingSpanner.strict-note-spacing = ##t
104    \override SpacingSpanner.uniform-stretching = ##t
105    \override StaffGrouper.staff-staff-spacing = #'(
106        (basic-distance . 0)
107        (minimum-distance . 6)
108        (padding . 2)
109    )
110    \override TupletBracket.bracket-visibility = ##t
111    \override TupletBracket.minimum-length = #3
112    \override TupletBracket.padding = #2
113    \override TupletBracket.springs-and-rods = #ly:spanner::set-spacing-
rods
114    \override TupletNumber.text = #tuple-number::calc-fraction-text

```

```

115 \override TextSpanner.Y-offset = 1
116 proportionalNotationDuration = #(ly:make-moment 1 50)
117     autoBeaming = ##f
118     tupletFullLength = ##t
119 }
120 \context {
121     \Voice
122     \remove Forbid_line_break_engraver
123 }
124 \context {
125     \Staff
126     \remove Time_signature_engraver
127 }
128 \context {
129     \Staff
130     \name BowStaff
131     \type Engraver_group
132     \alias Staff
133     \bowtab
134     \override Beam.stencil = ##f
135     \override Dots.stencil = ##f
136     \override Flag.stencil = ##f
137     \override Glissando.bound-details.left.padding = #0.5
138     \override Glissando.bound-details.right.padding = #0.5
139     \override Glissando.thickness = #2
140     \override NoteHead.Y-offset = #-5
141     \override NoteHead.extra-offset = #'(0.05 . 0)
142     \override NoteHead.stencil = ##f
143     \override Rest.transparent = ##t
144         \override Script.staff-padding = #2
145         \override StaffSymbol.transparent = ##t
146         \override Stem.direction = #down
147         \override Stem.stencil = ##f
148         \override TimeSignature.stencil = ##f
149     \override Tie.stencil = ##f
150         \override TupletBracket.stencil = ##f
151         \override TupletNumber.stencil = ##f
152 \%\\RemoveEmptyStaves
153 }

154 \context {
155     \Staff
156     \name BeamStaff
157     \type Engraver_group
158     \alias Staff
159     \override Beam.direction = #down
160     \override Beam.positions = #'(5 . 5)
161     \override Clef.stencil = ##f
162     \override Dots.staff-position = #-2
163     \override Flag.Y-offset = #2.93
164     \override NoteHead.no-ledgers = ##t
165     \override NoteHead.stencil = ##f
166     \override Rest.transparent = ##t
167     \override Script.staff-padding = #3

```

```

169     \override StaffSymbol.transparent = ##t
170     \override Stem.direction = #down
171     \override Stem.length = #0.5
172     \override Stem.stem-begin-position = #15.975
173     \override TimeSignature.stencil = ##f
174     \override Tie.stencil = ##f
175     \override TupletBracket.positions = #'(3 . 3)
176 }
177
178 \context {
179     \RhythmicStaff
180     \remove Time_signature_engraver
181 }
182 \context {
183     \StaffGroup
184     \accepts BowStaff
185     \accepts BeamStaff
186 }
187 }
188
189 \paper {
190
191     top-margin = 1.5\cm
192     bottom-margin = 1.5\cm
193
194     %top-margin = .90\in
195     oddHeaderMarkup = \markup ""
196     evenHeaderMarkup = \markup ""
197     oddFooterMarkup = \markup \fill-line {
198         ""
199         \concat {
200             "Cchar ~"
201             \fontsize #2
202             \fromproperty #'page:page-number-string "~ Evans"
203         }
204         ""
205     }
206     evenFooterMarkup = \markup \fill-line {
207         ""
208         \concat { "Cchar ~" \fontsize #2
209             \fromproperty #'page:page-number-string "~ Evans"
210         } ""
211     }
212 }
```

### Code Example 2.31: Cchar stylesheet

In this score, I defined a few new contexts in order to manage the specific visual properties I desired for a staff indicating bow motion with the *abjad.BowContactPoint()* tool. Aside from these properties, the composer is also able to edit graphic elements such as the width and spacing of beams, the thickness of stems, or the shape of flags. In this stylesheet, the clef symbol in the bowing staves is replaced with an *eps*

image of a bow to help indicate what the tablature represents.

These are just a few examples of ways in which Abjad and Lilypond allow for the simplification of processes that, by hand, could be extremely tedious over the course of a lengthy composition. The principles involved in these examples extend to every facet of both composing and engraving. Now that the power that Python can give composers has been described, next I will show how creating these loops and functions can have further ramifications in the process of composing.

#### 2.1.2.5 COMPOSING WITH MODELS AND ALGORITHMS

Composing with Abjad and Python allows the composer to work with models and algorithms. The following is an example where pitches are generated by a random walk which can be seen as a one-dimensional model of Brownian motion.<sup>38</sup> Much of my recent music features a similar procedure as the following:

```

1 >>> import abjad
2 >>> from random import seed
3 >>> from random import random
4 >>> seed(3)
5 >>> random_walk = []
6 >>> random_walk.append(-1 if random() < 0.5 else 1)
7 >>> for i in range(1, 64):
8 ...     movement = -1 if random() < 0.5 else 1
9 ...     value = random_walk[i-1] + movement
10 ...    random_walk.append(value)
11 >>> notes = [abjad.Note(x / 2.0, (1, 8)) for x in random_walk]
12 >>> staff = abjad.Staff(notes)
13 >>> abjad.show(staff)
```

**Code Example 2.32:** Random walk

In this code, the user must first create an empty list. Based on a string of randomly generated numbers, a new list of pitches is created, notated by numbers moving in a step of plus or minus 0.5 that are turned into note objects. They are next placed in a staff. It results in this Lilypond code and figure 2.1.12:

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     bqs8
5     c'8
```

---

<sup>38</sup> Brownian motion is a model used to describe the rapid and random motion of particles in a fluid.

6           **bqs8**  
7           **c'8**  
8           **cqs'8**  
9           **c'8**  
10          **bqs8**  
11          **c'8**  
12          **bqs8**  
13          **b8**  
14          **bqs8**  
15          **b8**  
16          **bqs8**  
17          **b8**  
18          **bqs8**  
19          **b8**  
20          **bqs8**  
21          **c'8**  
22          **cqs'8**  
23          **cs'8**  
24          **dqf'8**  
25          **cs'8**  
26          **dqf'8**  
27          **d'8**  
28          **dqf'8**  
29          **cs'8**  
30          **dqf'8**  
31          **cs'8**  
32          **dqf'8**  
33          **d'8**  
34          **dqs'8**  
35          **ef'8**  
36          **dqs'8**  
37          **ef'8**  
38          **dqs'8**  
39          **ef'8**  
40          **eqf'8**  
41          **ef'8**  
42          **dqs'8**  
43          **d'8**  
44          **dqs'8**  
45          **d'8**  
46          **dqs'8**  
47          **d'8**  
48          **dqs'8**  
49          **d'8**  
50          **dqf'8**  
51          **d'8**  
52          **dqs'8**  
53          **ef'8**  
54          **eqf'8**  
55          **e'8**  
56          **eqs'8**  
57          **f'8**  
58          **fqs'8**  
59          **f'8**

```

60      fqs '8
61      fs '8
62      gqf '8
63      g '8
64      gqs '8
65      g '8
66      gqs '8
67      af '8
68  }
69 } %! LilyPondFile

```

Code Example 2.33: Random walk: RESULT



Figure 2.1.12: A mapping of a random walk.

It is also possible to model more traditional compositional algorithms.<sup>39</sup> This code is more complex than what we have seen before.<sup>40</sup> This code creates a three-voice canon based on the melody input by the user. The melody is transposed and the rhythms are scaled to a different tempo. Voices with phrases that end before the slowest voice completes its phrase are repeated until the bottom voice has finished. Because of how the rhythms are scaled, it is important to use `abjad.mutate().rewrite_meter()` to ensure that all rhythms remain in the appropriate measure:

```

1 >>> import abjad
2 >>> def generate_scaled_staff(scale_factor, staff):
3 ...     staff_pitches = []
4 ...     for logical_tie in abjad.iterate(staff).logical_ties():
5 ...         first_leaf = logical_tie[0]
6 ...         staff_pitches.append(first_leaf.written_pitch)
7 ...     staff_durations = [

```

<sup>39</sup>Admittedly, mensuration canons do not present an extremely complex algorithm, but it meets my definition of algorithm.

<sup>40</sup>This code is adapted from code written by Jeffrey Treviño and presented as a part of the 2018 Abjad summer workshop at CCRMA<sup>41</sup> at Stanford University.

```

8 ... chain.written_duration*scale_factor for chain in abjad.iterate(staff).
9     logical_ties()
10    ]
11 ...
12    scaled_staff = abjad.Staff()
13    maker = abjad.NoteMaker()
14    selections = maker(staff_pitches, staff_durations)
15    scaled_staff.extend(selections)
16    return scaled_staff
17
18 >>> def partition_value(value):
19     if x >= 16:
20         divisions, remainder = divmod(value, 8)
21         parts = [8] * divisions
22         if remainder:
23             parts.append(remainder)
24     return parts
25
26 >>> def process_maxima(durations):
27     output_durations = []
28     for duration in durations:
29         if duration[0] >= 16:
30             numerators = partition_value(duration[0])
31             duration = [(numerator, 1) for numerator in numerators]
32     output_durations.append(duration)
33
34 >>> def scale_and_chop_staff(voice_number, staff, time_signature):
35     scale_factor = 2 ** voice_number
36     scaled_staff = generate_scaled_staff(scale_factor, staff)
37     abjad.mutate(scaled_staff).transpose(voice_number * -7)
38     abjad.mutate(scaled_staff[:]).split([time_signature], cyclic=True)
39     return scaled_staff
40
41 >>> def duplicate_music(num_copies, staff):
42     out_staff = abjad.Staff()
43     for x in range(num_copies):
44         out_staff.extend(abjad.mutate(staff).copy())
45     return out_staff
46
47 >>> def make_scaled_staves(melody_staff, time_signature):
48     scaled_staves = []
49     for voice_number in range(3):
50         scaled_staff = scale_and_chop_staff(voice_number, melody_staff,
51         time_signature)
52         scaled_staves.append(scaled_staff)
53     return scaled_staves
54
55 >>> def duplicate_score(scaled_staves):
56     score = abjad.Score()
57     for scaled_staff, duplicate_index in zip(scaled_staves, reversed(range
58     (3))):
59         scale_factor = 2**duplicate_index
60         staff = duplicate_music(scale_factor, scaled_staff)
61         score.append(staff)
62     return score

```

```

59
60 >>> def format_score(score, key_signature, time_signature):
61 ...     for staff in score:
62 ...         key_sig = abjad.KeySignature(key_signature.tonic, key_signature.
63 ...             mode)
64 ...         abjad.attach(key_sig, staff[0])
65 ...         time_sig = abjad.TimeSignature(time_signature)
66 ...         abjad.attach(time_sig, staff[0])
67 ...         abjad.attach(abjad.Clef('varC'), score[1][0])
68 ...         abjad.attach(abjad.Clef('bass'), score[2][0])
69
70
71
72
73
74
75 >>> def make_canon(melody_staff, key_signature, time_signature):
76 ...     scaled_staves = make_scaled_staves(melody_staff, time_signature)
77 ...     score = duplicate_score(scaled_staves)
78 ...     format_score(score, key_signature, time_signature)
79 ...     return score
80
81
82
83
84
85
86

```

**Code Example 2.34:** Mensuration canon

resulting in the Lilypond code and figure 2.1.13:

```

1 \score { %! LilyPondFile
2   \new Score
3   <<
4     \new Staff
5     {
6       \key c \major
7       \time 4/4
8       c'4
9       cs'8
10      d'8
11      ds'8
12      e'8
13      f'4
14      fs'4
15      g'4
16      gs'8
17      a'8
18      b'8
19      c''8

```

```

20      c'4
21      cs'8
22      d'8
23      ds'8
24      e'8
25      f'4
26      fs'4
27      g'4
28      gs'8
29      a'8
30      b'8
31      c'1'8
32      c'4
33      cs'8
34      d'8
35      ds'8
36      e'8
37      f'4
38      fs'4
39      g'4
40      gs'8
41      a'8
42      b'8
43      c'1'8
44      c'4
45      cs'8
46      d'8
47      ds'8
48      e'8
49      f'4
50      fs'4
51      g'4
52      gs'8
53      a'8
54      b'8
55      c'1'8
56  }
57 \new Staff
58 {
59     \key c \major
60     \time 4/4
61     \clef "varC"
62     f2
63     fs4
64     g4
65     gs4
66     a4
67     bf2
68     b2
69     c'2
70     cs'4
71     d'4
72     e'4
73     f'4

```

```

74      f2
75      fs4
76      g4
77      gs4
78      a4
79      bf2
80      b2
81      c'2
82      cs'4
83      d'4
84      e'4
85      f'4
86  }
87 \new Staff
88 {
89     \key c \major
90     \time 4/4
91     \clef "bass"
92     bf,1
93     b,2
94     c2
95     cs2
96     d2
97     ef1
98     e1
99     f1
100    fs2
101    g2
102    a2
103    bf2
104  }
105 >>
106 } %! LilyPondFile

```

**Code Example 2.35:** Mensuration canon: RESULT

Using Abjad and Python, composers are able to write music full of intricate relationships with precise formal consistency, but a comfortable formalism in score control is not necessarily algorithmic utopia. Though these logical procedures are available and entirely possible, they are optional. The process of composing with Abjad should not be misperceived as a purely algorithmic system for music composition. Certainly, formalizing elements in a score allows for a great amount of consistency and control, but composers have every ability to make decisions and sculpt the music at will if they so desire.<sup>42</sup> Abjad and

---

<sup>42</sup>Composing with the workflow of Python, Abjad, and Lilypond does present some difficulty in composing idiomatically for instruments. Piano music, in particular, presents a great challenge, a challenge that I have yet to surmount. If one is not careful, it is possible to compose music completely unplayable by a human performer.



**Figure 2.1.13:** A mensuration canon.

Lilypond do not dictate what kind of music is able to be composed. It is still the duty of the composer to constrain their musical practices to those they consciously wish to deploy.

### 2.1.3 THE NEED TO BUILD TOOLS FOR A MORE PERSONALIZED APPROACH TO MUSIC-MAKING

#### 2.1.3.1 BUILDING TOOLS

Why do I feel that it is important to write my own compositional tools? This is because each composer has a unique imagination and ideal. I feel greater satisfaction when I do not compose with the same methodology as another musician. I prefer a musical culture where each composer has a unique voice, because without it the beautiful diversity of new music would vanish. Abjad provides a framework for formalized score control but is not restrictive about the practices used to compose.<sup>43</sup> The fact that Abjad provides separate packages of tools for composition, as well as other functions, reveals that it is intended to be used by a variety of users with a variety of backgrounds. There are a handful of official extensions to Abjad under the title of abjad-ext.

---

<sup>43</sup>As I have previously written, even composing in notation programs such as Finale or Sibelius has restrictions and makes certain procedures difficult or impossible.

### 2.1.3.2 ABJAD-EXT

Abjad-ext consists of a number of packages that are not necessary for full functionality of the API. The packages include abjad-ext-tonality,<sup>44</sup> a tonal analysis extension, abjad-ext-book,<sup>45</sup> an extension for rendering Abjad code in L<sup>A</sup>T<sub>E</sub>X, abjad-ext-ipython,<sup>46</sup> an extension for rendering Abjad code in IPython and Jupyter notebooks, abjad-ext-nauert,<sup>47</sup> an extension of quantization tools based on Paul Nauert's Q-Grids, abjad-ext-cli,<sup>48</sup> a Command Line Interface extension, and abjad-ext-rmakers,<sup>49</sup> a rhythm maker tool extension. Each of these packages extend the functionality of Abjad, but I have only seriously used Trevor Bača's rmakers package. These packages exist outside of the main Abjad source in order to emphasize their optionality.<sup>50</sup>

### 2.1.3.3 RMAKERS

<sup>51</sup>

The rmakers consist of a set of tools for generating rhythmic material in certain characteristic ways. Contained in the rmakers package are a basic *RhythmMaker* class, *AccelerandoRhythmMaker*, *EvenDivisionRhythmMaker*, *IncisedRhythmMaker*, *NoteRhythmMaker*, *TaleaRhythmMaker*, and *TupletRhythmMaker*. An extended description of these tools and their functionality can be found in Josiah Oberholtzer's 2015 dissertation *A Computational Model of Music Composition*. (Oberholtzer, 2015, pp. 118–128) I am quite fond of these tools and, even though I intend to write my own rhythm-generating functions in the near future, they are the primary source of rhythmic composition in my recent music.

---

<sup>44</sup><https://github.com/Abjad/abjad-ext-tonality>

<sup>45</sup><https://github.com/Abjad/abjad-ext-book>

<sup>46</sup><https://github.com/Abjad/abjad-ext-ipython>

<sup>47</sup><https://github.com/Abjad/abjad-ext-nauert>

<sup>48</sup><https://github.com/Abjad/abjad-ext-cli>

<sup>49</sup><https://github.com/Abjad/abjad-ext-rmakers>

<sup>50</sup>The rmakers, previously called RhythmMakerTools, were once a part of the main Abjad source, but were externalized because their author felt they were more of a reflection of his own compositional practices than being a universal tool set.

<sup>51</sup>make a subsubsubsection

#### 2.1.4 OTHER PACKAGES

Along with the official abjad-ext packages are other packages by composers who make use of Abjad, including Consort,<sup>52</sup> a package written by Josiah Wolf Oberholtzer and described in detail in his dissertation, mtools<sup>53</sup> by Ivan Alexander Moscotta, and calliope<sup>54</sup> by Randall West. All of these packages present unique and innovative tools for music composition and have encouraged me to find my own way of composing with Abjad. As of the writing of this paper, I have only written one external tool set for composing, but they are used extensively in my scores, these are MusicMaker and the accompanying AttachmentHandler classes.

#### 2.1.5 MUSICMAKER

MusicMaker is a python class of mine which is the result of my attempt to combine material consistency of many kinds. MusicMaker takes the input of an rmaker and an optional set of AttachmentHandler objects. This tool was written because, although the rmakers are capable of generating rhythmic material, they do not handle pitch in any way. One could compose the entire rhythmic framework of a piece and add pitches after the fact. I found myself using many different rmakers throughout the course of a piece and I found that I was working very hard to unify rhythmic gestures by giving them unique harmonic fields and dynamic trajectories. This process became quite difficult and required a trial-and-error process or a tedious amount of pre-compositional calculation by hand outside of the computer program. My solution was to create a tool that could handle many different rmakers, pitch fields, and attachment characteristics at once, leaving the composer to define distinct and alternating characters and processes with which to compose. When MusicMaker is given an rmaker and a list of pitches inside of the accompanying PitchHandler object, it automatically adds those pitches to the rhythms cyclically. When the list of pitches runs out, it repeats endlessly to ensure that there is always pitch material when the MusicMaker is called. Because MusicMaker generates music based on the rhythms and pitches that were input by the user each time that it is called within the python file, it allows the composer to instantiate

---

<sup>52</sup><https://github.com/josiah-wolf-oberholtzer/consort>

<sup>53</sup><https://github.com/ivanalexandermoscotta/mtools>

<sup>54</sup><https://github.com/mirrorecho/calliope/tree/new-base>

multiple MusicMakers with unique rhythmic, harmonic, and dynamic qualities with other attachments. Much of my recent music is composed from alternating fragments of processes begun with MusicMaker. As I write more music, I find that there are more features that I wish to add to MusicMaker and the AttachmentHandler objects. As such, their code is still under revision. The following is the current source code for MusicMaker:

```

1 import abjad
2 from evans.AttachmentHandlers.GlissandoHandler import GlissandoHandler
3 from evans.AttachmentHandlers.NoteheadHandler import NoteheadHandler
4 from evans.AttachmentHandlers.PitchHandler import PitchHandler
5 from evans.AttachmentHandlers.ArticulationHandler import ArticulationHandler
6 from evans.AttachmentHandlers.DynamicHandler import DynamicHandler
7 from evans.AttachmentHandlers.TextSpanHandler import TextSpanHandler
8 from evans.AttachmentHandlers.ClefHandler import ClefHandler
9 from evans.AttachmentHandlers.SlurHandler import SlurHandler
10 from evans.AttachmentHandlers.TrillHandler import TrillHandler
11
12 class MusicMaker:
13     def __init__(
14         self,
15         rmaker,
16         glissando_handler=None,
17         notehead_handler=None,
18         pitch_handler=None,
19         articulation_handler=None,
20         dynamic_handler=None,
21         text_span_handler=None,
22         clef_handler=None,
23         slur_handler=None,
24         trill_handler=None,
25         continuous=False,
26         state=None,
27     ):
28         self.glissando_handler = glissando_handler
29         self.notehead_handler = notehead_handler
30         self.pitch_handler = pitch_handler
31         self.articulation_handler = articulation_handler
32         self.dynamic_handler = dynamic_handler
33         self.text_span_handler = text_span_handler
34         self.clef_handler = clef_handler
35         self.slur_handler = slur_handler
36         self.trill_handler = trill_handler
37         self.continuous = continuous
38         self.rmaker = rmaker
39         self.state = self.rmaker.state
40         self._count = 0
41
42     def __call__(self, durations):
43         return self._make_music(durations)
44

```

```

45     def _make_basic_rhythm(self, durations):
46         if self.continuous == True:
47             state = self.state
48             selections = self.rmaker(durations, previous_state=self.rmaker.
49                                         state)
50             self.state = self.rmaker.state
51         else:
52             selections = self.rmaker(durations, )
53         return selections
54
54     def _make_music(self, durations):
55         selections = self._make_basic_rhythm(durations)
56         if self.pitch_handler == None:
57             start_command = abjad.LilyPondLiteral(
58                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1
59 \startStaff',
60                 format_slot='before',
61             )
62             stop_command = abjad.LilyPondLiteral(
63                 r'\stopStaff \startStaff',
64                 format_slot='after',
65             )
66             literal = abjad.LilyPondLiteral(r'\once \override Staff.Clef.
transparent = ##t', 'before')
66             c_clef = abjad.LilyPondLiteral(r'\clef alto', 'before')
67             abjad.attach(literal, selections[0][0])
68             abjad.attach(c_clef, selections[0][0])
69             abjad.attach(start_command, selections[0][0])
70             abjad.attach(stop_command, selections[0][-1])
71         if self.pitch_handler != None:
72             selections = self.pitch_handler(selections)
73             if self.clef_handler != None:
74                 selections = self.clef_handler(selections)
75             if self.glissando_handler != None:
76                 selections = self.glissando_handler(selections)
77             if self.notehead_handler != None:
78                 selections = self.notehead_handler(selections)
79             if self.articulation_handler != None:
80                 selections = self.articulation_handler(selections)
81             if self.dynamic_handler != None:
82                 selections = self.dynamic_handler(selections)
83             if self.text_span_handler != None:
84                 selections = self.text_span_handler(selections)
85             if self.slur_handler != None:
86                 selections = self.slur_handler(selections)
87             if self.trill_handler != None:
88                 selections = self.trill_handler(selections)
89         return selections

```

**Code Example 2.36:** MusicMaker source

MusicMaker is made to be called with timespans<sup>55</sup> and can be used as follows:

---

<sup>55</sup>Timespans are also described in great detail in Josiah Oberholtzer's dissertation.(Oberholtzer, 2015, pp. 78–118)

```

 1 >>> import abjad
 2 >>> import itertools
 3 >>> import abjadext.rmakers
 4 >>> from evans.AttachmentHandlers.MusicMaker import MusicMaker
 5 >>> from evans.AttachmentHandlers.PitchHandler import PitchHandler
 6
 7 >>> time_signatures = [
 8 ...     abjad.TimeSignature(pair) for pair in [
 9 ...         (4, 4), (5, 4),
10 ...     ]
11 ... ]
12
13 >>> bounds = abjad.mathtools.cumulative_sums([_.duration for _ in
14 time_signatures])
15
15 >>> rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
16 ...     talea=abjadext.rmakers.Talea(
17 ...         counts=[1, 2, 3, 4],
18 ...         denominator=16,
19 ...     ),
20 ...     beamSpecifier=abjadext.rmakers.BeamSpecifier(
21 ...         beamDivisionsTogether=True,
22 ...         beamRests=False,
23 ...     ),
24 ...     extraCountsPerDivision=[0, 1, ],
25 ...     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
26 ...         trivialize=True,
27 ...         extractTrivial=True,
28 ...         rewriteRestFilled=True,
29 ...         rewriteSustained=True,
30 ...     ),
31 ... )
32
32 >>> rmaker_two = abjadext.rmakers.EvenDivisionRhythmMaker(
33 ...     denominators=[8, 16, ],
34 ...     extraCountsPerDivision=[0, ],
35 ...     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
36 ...         trivialize=True,
37 ...         extractTrivial=True,
38 ...         rewriteRestFilled=True,
39 ...     ),
40 ... )
41 ...
42
42 >>> musicmaker_one = MusicMaker(
43 ...     rmaker=rmaker_one,
44 ...     pitches=[0, 1, 2, 3, 4],
45 ...     continuous=True,
46 ... )
47 ...
48 >>> musicmaker_two = MusicMaker(
49 ...     rmaker=rmaker_two,
50 ...     pitchHandler=PitchHandler(pitchList=[4, 3, 2, 1, 0]),
51 ...     continuous=True,
52 ... )
53 >>> silence_maker = abjadext.rmakers.NoteRhythmMaker(

```

```

54     ...
55     division_masks=[
56         abjadext.rmakers.SilenceMask(
57             pattern=abjad.index([0], 1),
58             ),
59     )
60
61 >>> class MusicSpecifier:
62     ...
63     def __init__(self, music_maker, voice_name):
64         self.music_maker = music_maker
65         self.voice_name = voice_name
66
67 >>> voice_1_timespan_list = abjad.TimespanList([
68     abjad.AnnotatedTimespan(
69         start_offset=start_offset,
70         stop_offset=stop_offset,
71         annotation=MusicSpecifier(
72             music_maker=music_maker,
73             voice_name='Voice 1',
74             )),
75     for start_offset, stop_offset, music_maker in [
76         [(0, 4), (2, 4), musicmaker_one],
77         [(2, 4), (3, 4), musicmaker_one],
78         [(3, 4), (4, 4), musicmaker_one],
79         [(6, 4), (8, 4), musicmaker_two],
80         [(8, 4), (9, 4), silence_maker],
81         ],
82     ])
83
84 >>> all_timespan_lists = {
85     'Voice 1': voice_1_timespan_list,
86 }
87
88 >>> global_timespan = abjad.Timespan(
89     start_offset=0,
90     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
91 )
92
93 >>> for voice_name, timespan_list in all_timespan_lists.items():
94     silences = abjad.TimespanList([global_timespan])
95     silences.extend(timespan_list)
96     silences.sort()
97     silences.compute_logical_xor()
98     for silence_timespan in silences:
99         timespan_list.append(
100             abjad.AnnotatedTimespan(
101                 start_offset=silence_timespan.start_offset,
102                 stop_offset=silence_timespan.stop_offset,
103                 annotation=MusicSpecifier(
104                     music_maker=None,
105                     voice_name=voice_name,
106                     )),
107             )

```

```

108     ...
109     ...     timespan_list.sort()
110
111 >>> for voice_name, timespan_list in all_timespan_lists.items():
112     ...     shards = timespan_list.split_at_offsets(bounds)
113     ...     split_timespan_list = abjad.TimespanList()
114     ...     for shard in shards:
115     ...         split_timespan_list.extend(shard)
116     ...     split_timespan_list.sort()
117     ...     all_timespan_lists[voice_name] = timespan_list
118
119
120 >>> score = abjad.Score([
121     ...     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context',
122     ...     ),
123     ...     abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
124     ...     lilypond_type='Staff',),
125     ...     ],
126     ... )
127
128
129 >>> for time_signature in time_signatures:
130     ...     skip = abjad.Skip(1, multiplier=(time_signature))
131     ...     abjad.attach(time_signature, skip)
132     ...     score['Global Context'].append(skip)
133
134
135 >>> def make_container(music_maker, durations):
136     ...     selections = music_maker(durations)
137     ...     container = abjad.Container([])
138     ...     container.extend(selections)
139     ...     return container
140
141
142 >>> def key_function(timespan):
143     ...     return timespan.annotation.music_maker or silence_maker
144
145
146 >>> for voice_name, timespan_list in all_timespan_lists.items():
147     ...     for music_maker, grouper in itertools.groupby(
148     ...         timespan_list,
149     ...         key=key_function,
150     ...     ):
151     ...         durations = [timespan.duration for timespan in grouper]
152     ...         container = make_container(music_maker, durations)
153     ...         voice = score[voice_name]
154     ...         voice.append(container)
155
156
157 >>> for voice in abjad.iterate(score['Voice 1']).components(abjad.Voice):
158     ...     for i, shard in enumerate(abjad.mutate(voice[:]).split(
159     ...         time_signatures)):
160     ...         time_signature = time_signatures[i]
161     ...         abjad.mutate(shard).rewrite_meter(time_signature)
162
163
164 >>> for voice in abjad.select(score).components(abjad.Voice):
165     ...     for run in abjad.select(voice).runs():
166     ...         specifier = abjadext.rmakers.BeamSpecifier(
167     ...             beam_each_division=False,
168     ...         )

```

```

159 ...             specifier(run)
160 ...     abjad.beam(voice[:], beam_lone_notes=False, beam_rests=False, )
161
162 >>> abjad.show(score)

```

**Code Example 2.37:** Using MusicMaker with PitchHandler

With the assistance of a stylesheet for formatting, results in the following Lilypond code and figure

**2.1.14:**

```

1 \score { %! LilyPondFile
2     \new Score
3     <<
4         \context TimeSignatureContext = "Global Context"
5         {
6             \time 4/4
7             s1 * 1
8             \time 5/4
9             s1 * 5/4
10        }
11        \context Staff = "Staff 1"
12        {
13            \context Voice = "Voice 1"
14            {
15                {
16                    c'16
17                    [
18                    cs'16
19                    ~
20                    cs'16
21                    d'16
22                    ~
23                    d'8
24                    ef'8
25                    ~
26                    \times 4/5 {
27                        ef'8
28                        e'16
29                        c'8
30                    }
31                    cs'8.
32                    d'16
33                ]
34            }
35            {
36                r2
37            }
38            {
39                e'8
40                [
41                ef'8
42                d'8
43                cs'8

```

```

44         ]
45     }
46     {
47         r4
48     }
49   }
50 }
51 >>
52 } %! LilyPondFile

```

**Code Example 2.38:** Using MusicMaker with PitchHandler: RESULT



**Figure 2.1.14:** Notation from MusicMaker with two rmakers and PitchHandlers.

#### 2.1.6 ATTACHMENTHANDLERS

Along with MusicMaker, I have written a number of AttachmentHandler tools to control many other musical features. The current list of functioning tools consists of ArticulationHandler, ClefHandler, DynamicHandler, GlissandoHandler, NoteheadHandler, PitchHandler, SlurHandler, and TrillHandler. As the names imply, each of these tools contain processes for the application of various graphical elements within a score. They can be called alongside MusicMaker to create far more complex musical gestures and to handle a number of engraving issues that would typically be surmounted by hand outside of the python file. The source code for these tools is included in the appendix to this paper. With the exception of TrillHandler, each of the AttachmentHandler tools is able to be used both within MusicMaker and elsewhere in a file. TrillHandler can be used as follows:

```

1  >>> import abjad
2  >>> from TrillHandler import TrillHandler
3  >>> trill_handler = TrillHandler()
4  >>> staff = abjad.Staff(
5  ...     r'<a b>1 ~ <a b>4 \times 2/3 { a8 a8 <a b>8 } a4 a \times 2/3 { b8 a b
6  ...     }'
6  ... )

```

```

7 >>> score = abjad.Score([staff])
8 >>> trill_handler(score)
9 >>> abjad.show(score)

```

**Code Example 2.39:** Using TrillHandler

It results in the following:

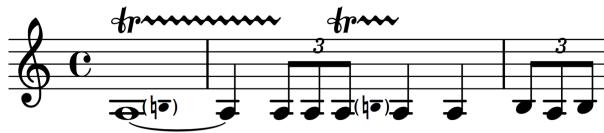
```

1 \version "2.19.82"  %! LilyPondFile
2 \language "english" %! LilyPondFile
3
4 \header { %! LilyPondFile
5     tagline = ##f
6 } %! LilyPondFile
7
8 \layout {}
9
10 \paper {}
11
12 \score { %! LilyPondFile
13     \new Score
14     <<
15         \new Staff
16         {
17             \pitchedTrill
18             a1
19             ^
20             \startTrillSpan b
21             a4
22             \times 2/3 {
23                 a8
24                 \stopTrillSpan
25                 a8
26                 \pitchedTrill
27                 a8
28                 \startTrillSpan b
29             }
30             a4
31             \stopTrillSpan
32             a4
33             \times 2/3 {
34                 b8
35                 a8
36                 b8
37             }
38         }
39     >>
40 } %! LilyPondFile

```

**Code Example 2.40:** Using TrillHandler: RESULT

and figure 2.1.15:



**Figure 2.1.15:** Trills.

The following example is a few bars for four voices exhibiting the functionality of each of the AttachmentHandlers.

```

1  >>> import abjad
2  >>> import itertools
3  >>> import os
4  >>> import pathlib
5  >>> import time
6  >>> import abjadext.rmakers
7  >>> from MusicMaker import MusicMaker
8  >>> from ArticulationHandler import ArticulationHandler
9  >>> from ClefHandler import ClefHandler
10 >>> from DynamicHandler import DynamicHandler
11 >>> from GlissandoHandler import GlissandoHandler
12 >>> from NoteheadHandler import NoteheadHandler
13 >>> from PitchHandler import PitchHandler
14 >>> from SlurHandler import SlurHandler
15 >>> from TextSpanHandler import TextSpanHandler
16
17 >>> print('Interpreting file ...')
18
19 >>> time_signatures = [
20 ...     abjad.TimeSignature(pair) for pair in [
21 ...         (4, 4),
22 ...     ]
23 ... ]
24
25 >>> bounds = abjad.mathtools.cumulative_sums([_.duration for _ in
26 ...     time_signatures])
27
27 >>> rmaker = abjadext.rmakers.TaleaRhythmMaker(
28 ...     talea=abjadext.rmakers.Talea(
29 ...         counts=[2, 1, 5, 1, 3, 3, 1, 2, ],
30 ...         denominator=16,
31 ...     ),
32 ...     beamSpecifier=abjadext.rmakers.BeamSpecifier(
33 ...         beamDivisionsTogether=True,
34 ...         beamRests=False,
35 ...     ),
36 ...     extraCountsPerDivision=[0, 0, 1, 0, -1],
37 ...     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
38 ...         leftClasses=[abjad.Note, abjad.Rest],
39 ...         leftCounts=[1, 0, 1],
40 ...     ),

```

```

41 ...     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
42 ...         trivialize=True,
43 ...         extract_trivial=True,
44 ...         rewrite_rest_filled=True,
45 ...         ),
46 ...     )
47
48 >>> articulation_handler = ArticulationHandler(
49 ...     articulation_list=['tenuto', 'staccato', 'portato', ],
50 ...     continuous=True,
51 ...     )
52
53 >>> clef_handler = ClefHandler(
54 ...     clef='bass',
55 ...     add_ottavas=True,
56 ...     # ottava_shelf=5,
57 ...     )
58
59 >>> dynamic_handler = DynamicHandler(
60 ...     starting_dynamic='f',
61 ...     hairpin='>',
62 ...     ending_dynamic='p',
63 ...     continuous=True,
64 ...     )
65
66 >>> glissando_handler = GlissandoHandler(
67 ...     line_style='dotted-line',
68 ...     )
69
70 >>> notehead_handler = NoteheadHandler(
71 ...     notehead_list=['cross', 'harmonic-mixed',
72 ...                   'diamond', 'triangle', 'slash', 'default', ],
73 ...     continuous=True,
74 ...     )
75
76 >>> pitch_handler = PitchHandler(
77 ...     pitch_list=[0, 2, 1, [3, 10], 4, 8, [7, 9], 6],
78 ...     continuous=True,
79 ...     )
80
81 >>> slur_handler = SlurHandler(
82 ...     slurs='runs',
83 ...     )
84
85 >>> text_span_handler = TextSpanHandler(
86 ...     position_list_one=['0/7', '5/7', '7/7', ],
87 ...     position_list_two=['two', 'three', 'one', ],
88 ...     position_list_three=['three', 'one', 'two', ],
89 ...     start_style_one='solid-line-with-arrow',
90 ...     start_style_two='dashed-line-with-arrow',
91 ...     stop_style_one='solid-line-with-hook',
92 ...     stop_style_two='dashed-line-with-hook',
93 ...     stop_style_three='solid-line-with-hook',
94 ...     apply_list_one_to='edges',

```

```

95 ...     apply_list_two_to='ties',
96 ...     apply_list_three_to='left_only',
97 ...     continuous=True,
98 ... )
99
100 >>> music_maker = MusicMaker(
101 ...     rmaker=rmaker,
102 ...     articulation_handler=articulation_handler,
103 ...     clef_handler=clef_handler,
104 ...     dynamic_handler=dynamic_handler,
105 ...     glissando_handler=glissando_handler,
106 ...     notehead_handler=notehead_handler,
107 ...     pitch_handler=pitch_handler,
108 ...     slur_handler=slur_handler,
109 ...     # text_span_handler=text_span_handler,
110 ...     continuous=True,
111 ... )
112
113 >>> silence_maker = abjadext.rmakers.NoteRhythmMaker(
114 ...     division_masks=[
115 ...         abjadext.rmakers.SilenceMask(
116 ...             pattern=abjad.index([0], 1),
117 ...             ),
118 ...         ],
119 ...     )
120
121 >>> class MusicSpecifier:
122
123 ...     def __init__(self, rhythm_maker, voice_name):
124 ...         self.rhythm_maker = rhythm_maker
125 ...         self.voice_name = voice_name
126
127 >>> print('Collecting timespans and rmakers ...')
128
129 >>> voice_1_timespan_list = abjad.TimespanList([
130 ...     abjad.AnnotatedTimespan(
131 ...         start_offset=start_offset,
132 ...         stop_offset=stop_offset,
133 ...         annotation=MusicSpecifier(
134 ...             rhythm_maker=rhythm_maker,
135 ...             voice_name='Voice 1',
136 ...             ),
137 ...         )
138 ...     for start_offset, stop_offset, rhythm_maker in [
139 ...         [(0, 8), (4, 8), music_maker],
140 ...         [(5, 8), (7, 8), music_maker],
141 ...         [(7, 8), (8, 8), silence_maker],
142 ...         ]
143 ...     ])
144
145 >>> voice_2_timespan_list = abjad.TimespanList([
146 ...     abjad.AnnotatedTimespan(
147 ...         start_offset=start_offset,
148 ...         stop_offset=stop_offset,

```

```

149     ...
150     ...
151     ...
152     ...
153     ...
154     ...
155     ...
156     ...
157     ...
158     ...
159     ...
160
161 >>> voice_3_timespan_list = abjad.TimespanList([
162     ...
163     abjad.AnnotatedTimespan(
164         ...
165         start_offset=start_offset,
166         ...
167         stop_offset=stop_offset,
168         ...
169         annotation=MusicSpecifier(
170             ...
171             rhythm_maker=rhythm_maker,
172             ...
173             voice_name='Voice 3',
174             ),
175     ...
176
177 >>> voice_4_timespan_list = abjad.TimespanList([
178     ...
179     abjad.AnnotatedTimespan(
180         ...
181         start_offset=start_offset,
182         ...
183         stop_offset=stop_offset,
184         ...
185         annotation=MusicSpecifier(
186             ...
187             rhythm_maker=rhythm_maker,
188             ...
189             voice_name='Voice 4',
190             ),
191     ...
192
193 >>> all_timespan_lists = {
194     ...
195     'Voice 1': voice_1_timespan_list,
196     ...
197     'Voice 2': voice_2_timespan_list,
198     ...
199     'Voice 3': voice_3_timespan_list,
200     ...
201     'Voice 4': voice_4_timespan_list,
202     ...
203 }
204
205 >>> global_timespan = abjad.Timespan(
206     ...
207     start_offset=0,
208     ...
209     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values()))

```

```

203 ... )
204
205 >>> for voice_name, timespan_list in all_timespan_lists.items():
206 ...     silences = abjad.TimespanList([global_timespan])
207 ...     silences.extend(timespan_list)
208 ...     silences.sort()
209 ...     silences.compute_logical_xor()
210 ...     for silence_timespan in silences:
211 ...         timespan_list.append(
212 ...             abjad.AnnotatedTimespan(
213 ...                 start_offset=silence_timespan.start_offset,
214 ...                 stop_offset=silence_timespan.stop_offset,
215 ...                 annotation=MusicSpecifier(
216 ...                     rhythm_maker=None,
217 ...                     voice_name=voice_name,
218 ...                 ),
219 ...             )
220 ...         )
221 ...     timespan_list.sort()
222
223 >>> for voice_name, timespan_list in all_timespan_lists.items():
224 ...     shards = timespan_list.split_at_offsets(bounds)
225 ...     split_timespan_list = abjad.TimespanList()
226 ...     for shard in shards:
227 ...         split_timespan_list.extend(shard)
228 ...     split_timespan_list.sort()
229 ...     all_timespan_lists[voice_name] = timespan_list
230
231 >>> score = abjad.Score([
232 ...     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context'),
233 ...     abjad.StaffGroup(
234 ...         [
235 abjad.Staff(
236 [abjad.Voice(name='Voice 1')], name='Staff 1', lilypond_type='Staff',
237 ),
238 abjad.Staff(
239 [abjad.Voice(name='Voice 2')], name='Staff 2', lilypond_type='Staff',
240 ),
241 abjad.Staff(
242 [abjad.Voice(name='Voice 3')], name='Staff 3', lilypond_type='Staff',
243 ),
244 abjad.Staff(
245 [abjad.Voice(name='Voice 4')], name='Staff 4', lilypond_type='Staff',
246 ),
247 ...     ],
248 ...     name='Staff Group',
249 ...   )
250 ... ])
251
252
253 >>> for time_signature in time_signatures:
254 ...     skip = abjad.Skip(1, multiplier=(time_signature))
255 ...     abjad.attach(time_signature, skip)

```

```

256     ...     score['Global Context'].append(skip)
257
258 >>> print('Making containers ...')
259
260 >>> def make_container(rhythm_maker, durations):
261     ...     selections = rhythm_maker(durations)
262     ...     container = abjad.Container([])
263     ...     container.extend(selections)
264     ...     return container
265
266 >>> def key_function(timespan):
267     ...     return timespan.annotation.rhythm_maker or silence_maker
268
269 >>> for voice_name, timespan_list in all_timespan_lists.items():
270     ...     for rhythm_maker, grouper in itertools.groupby(
271     ...         timespan_list,
272     ...         key=key_function,
273     ...     ):
274     ...         durations = [timespan.duration for timespan in grouper]
275     ...         container = make_container(rhythm_maker, durations)
276     ...         voice = score[voice_name]
277     ...         voice.append(container)
278
279 >>> print('Beaming runs ...')
280 >>> for voice in abjad.select(score['Staff Group']).components(abjad.Voice):
281     ...     for run in abjad.select(voice).runs():
282     ...         specifier = abjadext.rmakers.BeamSpecifier(
283     ...             beam_each_division=False,
284     ...             )
285     ...         specifier(run)
286     ...     abjad.beam(voice[:], beam_lone_notes=False, beam_rests=False,)
287
288 >>> print('Stopping Hairpins and Text Spans... ')
289
290 >>> for staff in abjad.iterate(score['Staff Group']).components(abjad.Staff):
291     ...     for run in abjad.select(staff).runs():
292     ...         last_leaf = run[-1]
293     ...         next_leaf = abjad.inspect(last_leaf).leaf(1)
294     ...         abjad.attach(abjad.StopHairpin(), next_leaf)
295
296 >>> print('Adding attachments ...')
297 >>> bar_line = abjad.BarLine('|||')
298 >>> metro = abjad.MetronomeMark((1, 4), 108)
299 >>> markup = abjad.Markup(r'\bold { A }')
300 >>> mark = abjad.RehearsalMark(markup=markup)
301
302 >>> def cyc(lst):
303     ...     count = 0
304     ...     while True:
305     ...         yield lst[count%len(lst)]
306     ...         count += 1
307
308 >>> instruments = cyc([
309     ...     abjad.Violin(),

```

```

310 ...     abjad.Violin(),
311 ...     abjad.Viola(),
312 ...     abjad.Cello(),
313 ... ])
314
315 >>> clefs = cyc([
316 ...     abjad.Clef('treble'),
317 ...     abjad.Clef('treble'),
318 ...     abjad.Clef('varC'),
319 ...     abjad.Clef('bass'),
320 ... ])
321
322 >>> abbreviations = cyc([
323 ...     abjad.MarginMarkup(markup=abjad.Markup('vln. I'),),
324 ...     abjad.MarginMarkup(markup=abjad.Markup('vln. II'),),
325 ...     abjad.MarginMarkup(markup=abjad.Markup('vla.'),),
326 ...     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
327 ... ])
328
329 >>> names = cyc([
330 ...     abjad.StartMarkup(markup=abjad.Markup('Violin I'),),
331 ...     abjad.StartMarkup(markup=abjad.Markup('Violin II'),),
332 ...     abjad.StartMarkup(markup=abjad.Markup('Viola'),),
333 ...     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
334 ... ])
335
336 >>> for staff in abjad.iterate(score['Staff Group']).components(abjad.Staff):
337 ...     leaf1 = abjad.select(staff).leaves()[0]
338 ...     abjad.attach(next(instruments), leaf1)
339 ...     abjad.attach(next(abbreviations), leaf1)
340 ...     abjad.attach(next(names), leaf1)
341
342 >>> for staff in abjad.select(score['Staff Group']).components(abjad.Staff)[0]:
343 ...     leaf1 = abjad.select(staff).leaves()[0]
344 ...     last_leaf = abjad.select(staff).leaves()[-1]
345 ...     abjad.attach(metro, leaf1)
346 ...     abjad.attach(bar_line, last_leaf)
347
348 >>> for staff in abjad.iterate(score['Global Context']).components(abjad.Staff):
349 ...     leaf1 = abjad.select(staff).leaves()[0]
350 ...     abjad.attach(mark, leaf1)
351
352 >>> for staff in abjad.iterate(score['Staff Group']).components(abjad.Staff):
353 ...     abjad.Instrument.transpose_from_sounding_pitch(staff)
354
355 >>> score_file = abjad.LilyPondFile.new(
356 ...     score,
357 ...     includes=['first_stylesheet.ily',
358 ...               '/Users/evansdsg2/abjad/docs/source/_stylesheets/abjad.ily'],
359 ...     ],
360 ... )
361

```

```

362 >>> abjad.SegmentMaker.comment_measure_numbers(score)
363
364 >>> abjad.show(score_file)

```

**Code Example 2.41:** Demonstration of AttachmentHandlers

It results in the following lilypond code:

```

1 \version "2.19.82"  %! LilyPondFile
2 \language "english" %! LilyPondFile
3
4 \include "first_stylesheet.ily" %!
5   LilyPondFile
6 \include "/Users/evansdsg2/abjad/docs/source/_stylesheets/abjad.ily" %!
7   LilyPondFile
8
9 \header { %! LilyPondFile
10   tagline = ##f
11 } %! LilyPondFile
12
13 \layout {}
14
15 \paper {}
16
17 \score { %! LilyPondFile
18   \new Score
19   <<
20     \context TimeSignatureContext = "Global Context"
21     {
22       % [Global Context measure 1] %! COMMENT_MEASURE_NUMBERS
23       \time 4/4
24       \mark \markup {
25         \bold
26         {
27           A
28         }
29       }
30       s1 * 1
31     }
32     \context StaffGroup = "Staff Group"
33     <<
34       \context Staff = "Staff 1"
35       \with
36       {
37         \consists Horizontal_bracket_engraver
38       }
39       {
40         \context Voice = "Voice 1"
41         {
42           %
43           % [Voice 1 measure 1] %! COMMENT_MEASURE_NUMBERS
44           \set Staff.shortInstrumentName =
45           \markup { "vln. I" }
46           \set Staff.instrumentName =

```

```

45   \markup { "Violin I" }
46   \tempo 4=108
47   \once \override Staff.NoteHead.style = #'cross
48   \clef "bass"
49   c'8
50   \f
51   - \tenuto
52   \>
53   - \tweak style #'dotted-line
54   \glissando
55   (
56   [
57   \once \override Staff.NoteHead.style = #'harmonic-
mixed
58   d'16
59   - \staccato
60   - \tweak style #'dotted-line
61   \glissando
62   ]
63   \once \override Staff.NoteHead.style = #'diamond
64   cs'4
65   ~
66   - \tweak style #'dotted-line
67   \glissando
68   \once \override Staff.NoteHead.style = #'diamond
69   cs'16
70   \p
71   - \tweak stencil #constante-hairpin
72   \<
73   )
74 }
75 {
76   r8
77   \!
78 }
79 {
80   \once \override Staff.NoteHead.style = #'triangle
81   \clef "bass"
82   cs'16
83   \f
84   - \portato
85   \>
86   - \tweak style #'dotted-line
87   \glissando
88   (
89   [
90   \ottava 1
91   \once \override Staff.NoteHead.style = #'slash
92   <ef' bf'>8.
93   \p
94   - \tenuto
95   - \tweak stencil #constante-hairpin
96   \<
97   )

```

```

98 ]
99   \ottava 0
100 }
101 {
102   r8
103   \!
104   \bar "||"
105 }
106 }
107 }
108 \context Staff = "Staff 2"
109 \with
110 {
111   \consists Horizontal_bracket_engraver
112 }
113 {
114   \context Voice = "Voice 2"
115   {
116     \times 8/9 {
117       % [Voice 2 measure 1] %! COMMENT_MEASURE_NUMBERS
118       \set Staff.shortInstrumentName =
119       \markup { "vln. II" }
120       \set Staff.instrumentName =
121       \markup { "Violin II" }
122       \ottava 1
123       \once \override Staff.NoteHead.style = #'default
124       \clef "bass"
125       <ef' bf'>8.
126       \f
127       - \staccato
128       \>
129       - \tweak style #'dotted-line
130       \glissando
131       (
132       [
133       \ottava 0
134       \ottava 1
135       \once \override Staff.NoteHead.style = #'cross
136       e'16
137       - \portato
138       - \tweak style #'dotted-line
139       \glissando
140       \ottava 0
141       \ottava 1
142       \once \override Staff.NoteHead.style = #'harmonic-
143     mixed
144     af'8
145     - \tenuto
146     - \tweak style #'dotted-line
147     \glissando
148     \ottava 0
149     \ottava 1
150     \ottava 1

```

```

151          \once \override Staff.NoteHead.style = #'diamond
152          <g' a'>8
153          - \staccato
154          - \tweak style #'dotted-line
155          \glissando
156          \ottava 0
157          \ottava 0
158          \ottava 1
159          \once \override Staff.NoteHead.style = #'triangle
160          fs'16
161          \p
162          - \portato
163          - \tweak stencil #constante-hairpin
164          \l(
165          )
166          ]
167          \ottava 0
168      }
169  }
170  {
171      r8
172      \!
173  }
174  {
175      \ottava 1
176      \once \override Staff.NoteHead.style = #'slash
177      \clef "bass"
178      fs'4
179      \f
180      - \tenuto
181      - \tweak stencil #constante-hairpin
182      \l(
183      )
184      (
185      \ottava 0
186  }
187  {
188      r8
189      \!
190  }
191  }
192 \context Staff = "Staff 3"
193 \with
194 {
195     \consists Horizontal_bracket_engraver
196 }
197 {
198     \context Voice = "Voice 3"
199     {
200         \tweak text #tuplet-number::calc-fraction-text
201         \times 8/7 {
202             % [Voice 3 measure 1] %! COMMENT_MEASURE_NUMBERS

```

```

205          \set Staff.shortInstrumentName =
206          \markup { vla. }
207          \set Staff.instrumentName =
208          \markup { Viola }
209          \ottava 1
210          \once \override Staff.NoteHead.style = #'default
211          \clef "bass"
212          fs'16
213          \f
214          - \staccato
215          \>
216          - \tweak style #'dotted-line
217          \glissando
218          (
219          [
220          \ottava 0
221          \once \override Staff.NoteHead.style = #'cross
222          c'16
223          - \portato
224          - \tweak style #'dotted-line
225          \glissando
226          \once \override Staff.NoteHead.style = #'harmonic-
227          mixed
228          d'8.
229          - \tenuto
230          - \tweak style #'dotted-line
231          \glissando
232          \once \override Staff.NoteHead.style = #'diamond
233          cs'8
234          \p
235          - \staccato
236          - \tweak stencil #constante-hairpin
237          \<
238          )
239          ]
240      }
241      {
242          r8
243          \!
244      }
245      {
246          \once \override Staff.NoteHead.style = #'triangle
247          \clef "bass"
248          cs'16
249          \f
250          - \portato
251          \>
252          - \tweak style #'dotted-line
253          \glissando
254          (
255          [
256          \ottava 1
257          \once \override Staff.NoteHead.style = #'slash

```

```

258         <ef' bf'>16
259         - \tenuto
260         - \tweak style #'dotted-line
261         \glissando
262         \ottava 0
263         \ottava 1
264         \once \override Staff.NoteHead.style = #'default
265         e'8
266         \p
267         - \staccato
268         - \tweak stencil #constante-hairpin
269         \(
270         )
271         ]
272         \ottava 0
273     }
274     {
275         r8
276         \!
277     }
278 }
279 \context Staff = "Staff 4"
280 \with
281 {
282     \consists Horizontal_bracket_engraver
283 }
284 {
285     \context Voice = "Voice 4"
286     {
287         %
288         % [Voice 4 measure 1] %! COMMENT_MEASURE_NUMBERS
289         \set Staff.shortInstrumentName =
290         \markup { vc. }
291         \set Staff.instrumentName =
292         \markup { Violoncello }
293         \ottava 1
294         \once \override Staff.NoteHead.style = #'cross
295         \clef "bass"
296         e'8
297         \f
298         - \portato
299         \>
300         - \tweak style #'dotted-line
301         \glissando
302         (
303         [
304         \ottava 0
305         \ottava 1
306         \once \override Staff.NoteHead.style = #'harmonic-
307         mixed
308         af'16
309         - \tenuto
310         - \tweak style #'dotted-line

```

```

311           \glissando
312       ]
313   \ottava 0
314   \ottava 1
315   \ottava 1
316   \once \override Staff.NoteHead.style = #'diamond
317   <g' a'>4
318   ~
319   - \tweak style #'dotted-line
320   \glissando
321   \once \override Staff.NoteHead.style = #'diamond
322   <g' a'>16
323   \p
324   - \tweak stencil #constante-hairpin
325   \l
326   )
327   \ottava 0
328   \ottava 0
329 }
330 {
331     r8
332     \!
333 }
334 {
335     \times 4/5 {
336         \ottava 1
337         \ottava 1
338         \once \override Staff.NoteHead.style = #'triangle
339         \clef "bass"
340         <g' a'>16
341         \f
342         - \staccato
343         \r
344         - \tweak style #'dotted-line
345         \glissando
346         (
347         [
348         \ottava 0
349         \ottava 0
350         \ottava 1
351         \once \override Staff.NoteHead.style = #'slash
352         fs'8.
353         - \portato
354         - \tweak style #'dotted-line
355         \glissando
356         \ottava 0
357         \once \override Staff.NoteHead.style = #'default
358         c'16
359         \p
360         - \tenuto
361         - \tweak stencil #constante-hairpin
362         \l
363         )
364     ]

```

```

365 }
366 }
367 {
368     r8
369     \!
370 }
371 }
372 }
373 >>
374 >>
375 } %! LilyPondFile

```

**Code Example 2.42:** Demonstration of AttachmentHandlers: RESULT

and figure 2.1.16:

**Figure 2.1.16:** Four voice demonstration of AttachmentHandlers.

## 2.2 EDITING SOURCE CODE

While I have written my own tools for composition with Abjad, I have also occasionally found it necessary to edit Abjad's source code in order to include features that I desire, which is possible because Abjad is open source.

### 2.2.1 CLEF.PY

Recently I edited the *Clef.py* file in the Abjad source. I did this in order to include Abjad representations of clefs that were present in the most recent update of Lilypond. The clefs in question were *varC* and *tenorvarC*, both of which are alternative “c” clefs. I wanted to add these clefs for a logical reason as well as a personal reason. The first reason is that, as much as possible, Abjad should have a representation of all of Lilypond’s features. If a composer knows that Lilypond is capable of producing a certain graphic object,<sup>56</sup> it can be very frustrating to find no way to use it in Abjad. The second reason is that these clefs more closely represent my own handwriting of c clefs than the traditional c clef.

### 2.2.2 ARTICULATION.PY

Just as with *Clef.py* I also recently edited the *Articulation.py* file. I did this at the same time and for the same reason as editing the clef file. I added Abjad representations of “halfopen,” which is a circle with a diagonal slash and “snappizzicato,” which is the common notation for a snap or “Bartok” pizzicato.

### 2.2.3 MICROTONAL EXPANSION IN ABJAD 2.21

In the summer of 2018, I undertook, with help from Ivan Moscotta, a much larger revision of Abjad’s source code. These edits were specifically centered around Abjad’s representation of pitch. At the time, the most recent version of Abjad was Abjad 2.21. Also during this summer, I attended the CCRMA Abjad workshop and I was able to discuss some of these changes with the primary maintainers of the system: Trevor Bača and Josiah Oberholtzer. We came to the conclusion that much of Abjad’s representation of microtones should be reassessed and should be open enough for composers to be able to define their own accidentals and scales. Because of this decision, the changes that I made to Abjad 2.21’s code are not available for users in the main branch of Abjad 3.1, but will hopefully be given new birth in a future release.<sup>57</sup>

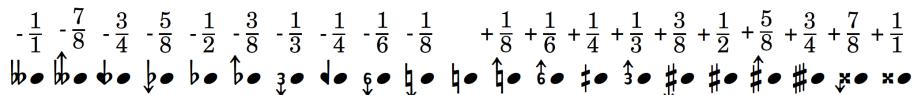
---

<sup>56</sup>These are often called grobs in Lilypond-lingo.

<sup>57</sup>However, these changes are available in my greg/dev branch of Abjad.

### 2.2.3.1 PROCESS

I decided to undertake this major revision because I wanted to compose with microtones smaller than Lilypond's and Abjad's smallest interval, which is the quarter tone. In Lilypond's font, Emmantaler, are two different kinds of quarter tones. Quarter tones written in Stein-Ellis notation and quarter tones written as traditional accidentals with an attached arrow either up or down to represent the microtonal alteration. I decided to use these arrow-based quarter tones to represent eighth tones.<sup>58</sup> In fact, there is a file buried deep within Lilypond called *Microtonal.ily* that does just this. The file must be included at the header of the Lilypond file in order to make use of the user-defined microtones. I tried to find a way to do this and had a little success, although with great difficulty. I began to wonder if it was possible to extend this to further divisions of the octave. I began to edit the default font in Lilypond to be able to represent different kinds of accidentals as well as making some slight changes to the default accidentals for my own graphic preference, generally keeping to the Stein-Zimmerman notation for quarter tones<sup>59</sup> and the Ferneyhough notation for all other microtonal alterations.<sup>60</sup> This became cumbersome and inconsistent and I looked for an alternative. Fortunately, I found the Ekmelily system.<sup>61</sup> This extension of Lilypond, written by Thomas Richter, does something similar to *Microtonal.ily*, but it also comes with an extensive font extension to allow for many kinds of microtonal representations and the ability to create user-defined scales with accidental grobs chosen by the user. This was my solution for graphically representing my new microtones. The following image in figure 2.2.1 is a representation of my own user-defined scale:



**Figure 2.2.1:** ekmelicStyle evans scale.

<sup>58</sup>This decision is informed, in part, by my familiarity with OpenMusic's accidentals.

<sup>59</sup><https://w3c.github.io/smufl/gitbook/tables/extended-stein-zimmermann-accidentals.html>

<sup>60</sup><https://w3c.github.io/smufl/gitbook/tables/other-accidentals.html>

<sup>61</sup><http://www.ekmelic-music.org/en/extra/ekmelily.htm>

### 2.2.3.2 FILE SYSTEMS AND ALTERATIONS

There were a few files in Abjad that needed to be changed in order to interface with Ekmelily via Abjad.

These files were *Accidental.py*, *NumberedPitchClass.py*, *PitchClass.py*, and *language\_pitch\_names.py*. I also edited the *language\_pitch\_names.ly* file in Lilypond. In these files I defined the name, division size, and abbreviation of each new accidental and linked these abbreviations to my user-defined scale in Ekmelily.

Making sure to always include Ekmelily and my own scale at the beginning of each Lilypond file, I was able to compose music in abjad with eighth tones, third tones, and sixth tones. All of the code alterations for this functionality is available at <https://github.com/GregoryREvans/Abjad-Microtones>.

The following image in figure 2.2.2 is a random walk generated with a pitch depth of eighth tones from my development branch of Abjad:



**Figure 2.2.2:** An eighth tone random walk.

## 2.3 CONCLUSION

In this chapter, I have described my methodology for composing in Abjad and the tools that I have written to assist in my compositional process. In the next chapter I will present the source code and scores of recent music that I have written with Abjad, all of which is available at <https://github.com/GregoryREvans>. Since the composition of these pieces, I have begun to experiment with a new organizational principle for my music. I now think of each piece as being similar to a piece of software which can make use of a number of data sets housed externally from the segment files of the composition. I now foresee a composition that is highly externalized. This will allow for greater fluidity when calling variables, functions, and lists throughout the composition process and will potentially

contribute to a greater sense of continuity throughout a piece while simplifying and reducing the amount of code written. The first piece that I am beginning to compose in this manner can be found at <https://github.com/GregoryREvans/onkos>. Because the pieces included in this appendix were written before this paradigm shift, the source code has a tendency to be overly long and at times redundant.

This page is intentionally left blank.

*I realized that one of the fundamental questions in music is its ability to make variations but preserving certain peculiarities, making it stable. We see variations; what is kept is the harmonic structure while music follows its own path. We also find constant structures, dilations and scale reductions in the music of the past [...]. Certain types of topologies are therefore obviously established.*

(2017, *Metamodels in Compositional Practices* p.85)

Francisco Guerrero



## Appendix of Source Code

CONTAINED IN THIS APPENDIX is the source code of my AttachmentHandlers as well as three of my compositions: “Cchar,” “Tianshu,” and “Four Ages of Sand.” The compilation of these materials and the scores in appendix B constitute the fruits of my recent research into computer assisted composition and engraving.

### A.1 ATTACHMENTHANDLERS

#### A.1.1 ARTICULATIONHANDLER

```
1 import abjad
2
3 class ArticulationHandler:
4
5     def __init__(
6         self,
7         articulation_list=None,
8         continuous=False,
9     ):
10    def cyc(lst):
11        if self.continuous == False:
12            self._count = 0
13        while True:
14            yield lst[self._count % len(lst)]
15            self._count += 1
16        self.articulation_list = articulation_list
17        self.continuous = continuous
18        self._cyc_articulations = cyc(articulation_list)
19        self._count = 0
20
```

```

21     def __call__(self, selections):
22         return self.add_articulations(selections)
23
24     def add_articulations(self, selections):
25         ties = abjad.select(selections).logical_ties(pitched=True)
26         for tie in ties:
27             if len(tie) == 1:
28                 if self.articulation_list != None:
29                     articulation = self._cyc_articulations
30                     abjad.attach(abjad.Articulation(next(articulation)), tie
31 [0])
32         return selections

```

**Code Example A.1:** ArticulationHandler

### A.1.2 CLEFHANDLER

```

1 import abjad
2
3 class ClefHandler:
4
5     def __init__(
6         self,
7         clef=None,
8         ottava_shelf=None,
9         add_ottavas=False,
10    ):
11        self.clef = clef
12        self.ottava_shelf = ottava_shelf
13        self.add_ottavas = add_ottavas
14
15    def __call__(self, selections):
16        return self.add_clef(selections)
17
18    def add_clef(self, selections):
19        for run in abjad.select(selections).runs():
20            ties = abjad.select(run).logical_ties(pitched=True)
21            if self.clef != None:
22                abjad.attach(abjad.Clef(self.clef), ties[0][0])
23            if self.add_ottavas == True:
24                self._add_ottavas(selections)
25        return selections
26
27    def _add_ottavas(self, selections):
28        if self.clef == 'treble':
29            if self.ottava_shelf != None:
30                shelf = self.ottava_shelf
31                for tie in abjad.select(selections).logical_ties():
32                    for pitch in abjad.inspect(tie[0]).pitches():
33                        if pitch > shelf:
34                            abjad.ottava(tie)
35            else:
36                shelf = 36

```

```

37         for tie in abjad.select(selections).logical_ties(pitched=True)
38     :
39         for pitch in abjad.inspect(tie[0]).pitches():
40             if pitch > shelf:
41                 abjad.ottava(tie)
42     if self.clef == 'alto':
43         if self.ottava_shelf != None:
44             shelf = self.ottava_shelf
45             for tie in abjad.select(selections).logical_ties(pitched=True)
46     :
47         for pitch in abjad.inspect(tie).pitches():
48             if pitch > shelf:
49                 abjad.ottava(tie)
50     else:
51         shelf = 13
52         for tie in abjad.select(selections).logical_ties(pitched=True)
53     :
54         for pitch in abjad.inspect(tie[0]).pitches():
55             if pitch > shelf:
56                 abjad.ottava(tie)
57     if self.clef == 'varC':
58         if self.ottava_shelf != None:
59             shelf = self.ottava_shelf
60             for tie in abjad.select(selections).logical_ties(pitched=True)
61     :
62         for pitch in abjad.inspect(tie[0]).pitches():
63             if pitch > shelf:
64                 abjad.ottava(tie)
65     else:
66         shelf = 13
67         for tie in abjad.select(selections).logical_ties(pitched=True)
68     :
69         for pitch in abjad.inspect(tie[0]).pitches():
70             if pitch > shelf:
71                 abjad.ottava(tie)
72     if self.clef == 'tenor':
73         if self.ottava_shelf != None:
74             shelf = self.ottava_shelf
75             for tie in abjad.select(selections).logical_ties(pitched=True)
76     :
77         for pitch in abjad.inspect(tie[0]).pitches():
78             if pitch > shelf:
79                 abjad.ottava(tie)
80     if self.clef == 'tenorvarC':
81         if self.ottava_shelf != None:
82             shelf = self.ottava_shelf
83             for tie in abjad.select(selections).logical_ties(pitched=True)

```

```

84         :
85             for pitch in abjad.inspect(tie[0]).pitches():
86                 if pitch > shelf:
87                     abjad.ottava(tie)
88             else:
89                 shelf = 10
90                 for tie in abjad.select(selections).logical_ties(pitched=True)
91                     :
92                         for pitch in abjad.inspect(tie[0]).pitches():
93                             if pitch > shelf:
94                                 abjad.ottava(tie)
95             if self.clef == 'bass':
96                 if self.ottava_shelf != None:
97                     shelf = self.ottava_shelf
98                     for tie in abjad.select(selections).logical_ties(pitched=True)
99                         :
100                         for pitch in abjad.inspect(tie[0]).pitches():
101                             if pitch > shelf:
102                                 abjad.ottava(tie)
103             else:
104                 shelf = 3
105                 for tie in abjad.select(selections).logical_ties(pitched=True)
106             :
107                 for pitch in abjad.inspect(tie[0]).pitches():
108                     if pitch > shelf:
109                         abjad.ottava(tie)
110     return(selections)

```

Code Example A.2: ClefHandler

### A.1.3 DYNAMICHANDLER

```

1 import abjad
2
3 class DynamicHandler:
4
5     def __init__(
6         self,
7         starting_dynamic=None,
8         ending_dynamic=None,
9         hairpin=None,
10        continuous=False,
11    ):
12     def cyc(lst):
13         if self.continuous == False:
14             self._count = 0
15         while True:
16             yield lst[self._count % len(lst)]
17             self._count += 1
18         self.starting_dynamic = starting_dynamic
19         self.ending_dynamic = ending_dynamic
20         self.hairpin = hairpin
21         self.continuous = continuous

```

```

22     self._cyc_dynamics = cyc([starting_dynamic, ending_dynamic])
23     self._count = 0
24
25     def __call__(self, selections):
26         return self.add_dynamics(selections)
27
28     def add_dynamics(self, selections):
29         runs = abjad.select(selections).runs()
30         ties = abjad.select(selections).logical_ties(pitched=True)
31         for run in runs:
32             if len(run) > 1:
33                 leaves = abjad.select(run).leaves()
34                 if self.starting_dynamic != None:
35                     abjad.attach(abjad.Dynamic(self.starting_dynamic), leaves
36 [0])
37                     if self.hairpin != None:
38                         abjad.attach(abjad.StartHairpin(self.hairpin), leaves[0])
39                     if self.ending_dynamic != None:
40                         abjad.attach(abjad.Dynamic(self.ending_dynamic), leaves
41 [-1])
42                         abjad.attach(abjad.StartHairpin('---'), leaves[-1])
43                 else:
44                     leaves = abjad.select(run).leaves()
45                     dynamic = next(self._cyc_dynamics)
46                     if self.starting_dynamic != None:
47                         if self.ending_dynamic != None:
48                             abjad.attach(abjad.Dynamic(dynamic), leaves[0])
49                         else:
50                             abjad.attach(abjad.Dynamic(self.starting_dynamic),
51 leaves[0])
52                         if self.starting_dynamic == None:
53                             if self.ending_dynamic != None:
54                                 abjad.attach(abjad.Dynamic(self.ending_dynamic),
55 leaves[0])
56                                 abjad.attach(abjad.StartHairpin('---'), leaves[0])
57             return selections

```

**Code Example A.3:** DynamicHandler

#### A.1.4 GLISSANDOHANDLER

```

1 import abjad
2
3 class GlissandoHandler:
4
5     def __init__(
6         self,
7         glissando_style=None,
8         line_style=None,
9     ):
10        self.glissando_style = glissando_style
11        self.line_style = line_style
12

```

```

13     def __call__(self, selections):
14         return self.add_glissando(selections)
15
16     def add_glissando(self, selections):
17         runs = abjad.select(selections).runs()
18         if self.glissando_style == 'hide_middle_note_heads':
19             if self.line_style != None:
20                 for run in runs:
21                     if len(run) > 1:
22                         abjad.glissando(run[:], abjad.tweak(self.line_style).
23 style, hide_middle_note_heads=True, )
24             else:
25                 for run in runs:
26                     if len(run) > 1:
27                         abjad.glissando(run[:], hide_middle_note_heads=True, )
28         elif self.glissando_style == 'hide_middle_stems':
29             if self.line_style != None:
30                 for run in runs:
31                     if len(run) > 1:
32                         abjad.glissando(run[:], abjad.tweak(self.line_style).
33 style, hide_middle_note_heads=True, hide_middle_stems=True, )
34             else:
35                 for run in runs:
36                     if len(run) > 1:
37                         abjad.glissando(run[:], hide_middle_note_heads=True,
38 hide_middle_stems=True, )
38             else:
39                 if self.line_style != None:
40                     for run in runs:
41                         if len(run) > 1:
42                             abjad.glissando(run[:], abjad.tweak(self.line_style).
43 style, allow_repeats=True, allow_ties=True, )
44             else:
45                 for run in runs:
46                     if len(run) > 1:
47                         abjad.glissando(run[:], allow_repeats=True, allow_ties
= True, )
48         return selections

```

**Code Example A.4:** GlissandoHandler

### A.1.5 NOTEHEADHANDLER

```

1 import abjad
2
3 class NoteheadHandler:
4
5     def __init__(
6         self,
7         notehead_list=None,
8         continuous=False,
9     ):
10        def cyc(lst):

```

```

11     if self.continuous == False:
12         self._count = 0
13     while True:
14         yield lst[self._count % len(lst)]
15         self._count += 1
16     self.notehead_list = notehead_list
17     self.continuous = continuous
18     self._cyc_noteheads = cyc(notehead_list)
19     self._count = 0
20
21 def __call__(self, selections):
22     return self.add_noteheads(selections)
23
24 def add_noteheads(self, selections):
25     if self.notehead_list != None:
26         head = self._cyc_noteheads
27         for tie in abjad.select(selections).logical_ties(pitched=True):
28             head_name = next(head)
29             string = str(r"""\once \override Staff.NoteHead.style = #!""")
30             full_string = string + head_name
31             style = abjad.LilyPondLiteral(full_string, format_slot='before
32 ',)
33             for leaf in abjad.select(tie).leaves(pitched=True):
34                 abjad.attach(style, leaf)
35     return selections

```

**Code Example A.5:** NoteheadHandler

#### A.1.6 PITCHHANDLER

```

1 import abjad
2
3 class PitchHandler:
4
5     def __init__(
6         self,
7         pitch_list=None,
8         continuous=False,
9     ):
10        def cyc(lst):
11            if self.continuous == False:
12                self._count = 0
13            while True:
14                yield lst[self._count % len(lst)]
15                self._count += 1
16            self.pitch_list = pitch_list
17            self.continuous = continuous
18            self._cyc_pitches = cyc(pitch_list)
19            self._count = 0
20
21    def __call__(self, selections):
22        return self._apply_pitches(selections, self.pitch_list)
23

```

```

24     def _collect_pitches_durations_leaves(self, logical_ties, pitches):
25         def cyc(lst):
26             if self.continuous == False:
27                 self._count = 0
28             while True:
29                 yield lst[self._count % len(lst)]
30                 self._count += 1
31         cyc_pitches = cyc(pitches)
32         pitches, durations, leaves = [[], [], []]
33         for tie in logical_ties:
34             if isinstance(tie[0], abjad.Note):
35                 pitch = next(cyc_pitches)
36                 for leaf in tie:
37                     pitches.append(pitch)
38                     durations.append(leaf.written_duration)
39                     leaves.append(leaf)
40             else:
41                 for leaf in tie:
42                     pitches.append(None)
43                     durations.append(leaf.written_duration)
44                     leaves.append(leaf)
45         return pitches, durations, leaves
46
47     def _apply_pitches(self, selections, pitches):
48         leaf_maker = abjad.LeafMaker()
49         container = abjad.Container(selections)
50         old_ties = [tie for tie in abjad.iterate(
51             container).logical_ties()]
52         pitches, durations, old_leaves = self.
53         _collect_pitches_durations_leaves(
54             old_ties, pitches)
55         new_leaves = [leaf for leaf in leaf_maker(pitches, durations)]
56         for old_leaf, new_leaf in zip(old_leaves, new_leaves):
57             indicators = abjad.inspect(old_leaf).indicators()
58             for indicator in indicators:
59                 abjad.attach(indicator, new_leaf)
60             parent = abjad.inspect(old_leaf).parentage().parent
61             parent[parent.index(old_leaf)] = new_leaf
62         return [container[:]]

```

**Code Example A.6:** PitchHandler

### A.1.7 SLURHANDLER

```

1 import abjad
2
3 class SlurHandler:
4
5     def __init__(
6         self,
7         slurs=None,
8     ):
9         self.slurs = slurs

```

```

10
11     def __call__(self, selections):
12         return self.add_slurs(selections)
13
14     def add_slurs(self, selections):
15         if self.slurs == 'selections':
16             abjad.slur(selections[:])
17         elif self.slurs == 'runs':
18             for run in abjad.select(selections).runs():
19                 abjad.slur(run[:])
20         else:
21             pass
22     return selections

```

**Code Example A.7:** SlurHandler

#### A.1.8 TRILLHANDLER

```

1 import abjad
2
3 class TrillHandler:
4
5     def __call__(self, selections):
6         return self._apply_trills(selections)
7
8     def _apply_trills(self, selections):
9         container = abjad.Container()
10        container.append(selections)
11
12        for tie in abjad.iterate(container).logical_ties(pitched=True):
13            if all(isinstance(leaf, abjad.Chord) for leaf in abjad.iterate(tie)
14).leaves()):
15                old_chord = tie[0]
16                base_pitch = old_chord.written_pitches[0]
17                trill_pitch = old_chord.written_pitches[-1]
18                interval_ = abjad.NamedInterval().from_pitch_carriers(
19                base_pitch, trill_pitch)
20                new_leaf = abjad.Note(base_pitch, old_chord.written_duration)
21
22                trill_start = abjad.LilyPondLiteral(r'\pitchedTrill',
23                format_slot='before')
24                trill_literal = abjad.LilyPondLiteral(f'\startTrillSpan {trill_pitch}', format_slot='after')
25                trill_stop = abjad.LilyPondLiteral(r'\stopTrillSpan',
26                format_slot='after')
27                abjad.attach(trill_start, new_leaf)
28                abjad.attach(trill_literal, new_leaf)
29                last_leaf = tie[-1]
30                next_leaf = abjad.inspect(last_leaf).leaf(1)
31                if next_leaf != None:
32                    abjad.attach(trill_stop, next_leaf)
33
34                indicators = abjad.inspect(old_chord).indicators()

```

```

31         for indicator in indicators:
32             abjad.attach(indicator, new_leaf)
33
34         parent = abjad.inspect(old_chord).parentage().parent
35         parent[parent.index(old_chord)] = new_leaf
36
37         tail = abjad.select(tie).leaves()[1:]
38         for leaf in tail:
39             new_tail = abjad.Note(base_pitch, leaf.written_duration)
40             parent = abjad.inspect(leaf).parentage().parent
41             parent[parent.index(leaf)] = new_tail
42             indicators = abjad.inspect(leaf).indicators()
43             for indicator in indicators:
44                 abjad.attach(indicator, new_tail)
45
46     return container[:]

```

**Code Example A.8:** TrillHandler

## A.2 CTHAR (FOR TWO CELLOS) SOURCE CODE

### A.2.1 SEGMENT

#### A.2.1.1 SEGMENT\_I

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (4, 4), (5, 4), (3, 4), (3, 4), (5, 4), (3, 4),
17         (4, 4), (4, 4), (5, 4), (4, 4), (4, 4), (4, 4),
18         (3, 4), (5, 4), (4, 4), (4, 4), (4, 4), (4, 4),
19         (4, 4), (5, 4), (3, 4), (4, 4), (3, 4), (3, 4),
20         (5, 4), (4, 4), (3, 4), (4, 4), (4, 4), (5, 4),
21         (3, 4), (5, 4), (5, 4), (4, 4), (3, 4), (3, 4),
22         (4, 4), (4, 4), (4, 4), (4, 4), (5, 4), (4, 4),
23         (5, 4), (4, 4), (4, 4), (5, 4), (5, 4),
24     ]
25 ]

```



```

74     movement = -1 if random() < 0.5 else 1
75     value = cello_random_walk_two[i-1] + movement
76     cello_random_walk_two.append(value)
77     cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
78     cello_chord_two = [-24, -11, -20, -6, -12, -6, 0, -11, -6, 4, 0, 6, 0, -11,
79         -6, -24, -8, 0, ]
80     cello_notes_two_walk = [cello_chord_two[x] for x in reduceMod17(
81         cello_random_walk_two)]
82     map_1 = [1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, ]
83     cello_notes_two = grouper(cello_notes_two_walk, map_1)
84
85     seed(3)
86     cello_random_walk_three = []
87     cello_random_walk_three.append(-1 if random() < 0.5 else 1)
88     for i in range(1, 1000):
89         movement = -1 if random() < 0.5 else 1
90         value = cello_random_walk_three[i-1] + movement
91         cello_random_walk_three.append(value)
92     cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
93     cello_chord_three = [-24, -20, -15, -14, -4, 5, 11, 19, 26, 37, 39, 42, 39,
94         37, 26, 19, 11, 5, -4, -14, -15, -20, ]
95     cello_notes_three = [cello_chord_three[x] for x in reduceMod21(
96         cello_random_walk_three)]
97
98     seed(4)
99     cello_random_walk_four = []
100    cello_random_walk_four.append(-1 if random() < 0.5 else 1)
101    for i in range(1, 2000):
102        movement = -1 if random() < 0.5 else 1
103        value = cello_random_walk_four[i-1] + movement
104        cello_random_walk_four.append(value)
105    cello_random_walk_four = [abs(x) for x in cello_random_walk_four]
106    cello_chord_four = [-17, -8, -13, -5, 5, -5, -13, -8, ]
107    map_2 = [2, 1, 2, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 2, 1, ]
108    cello_notes_four_walk = [cello_chord_four[x] for x in reduceMod7(
109        cello_random_walk_four)]
110    cello_notes_four = grouper(cello_notes_four_walk, map_2)
111
112    rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
113        talea=abjadext.rmakers.Talea(
114            counts=[7, 4, 6, 3, 5, 3, 5, 3, 6, 4],
115            denominator=32,
116            ),
117        beamSpecifier=abjadext.rmakers.BeamSpecifier(
118            beamDivisionsTogether=True,
119            beamRests=False,
120            ),
121        extraCountsPerDivision=[0, 1, 0, -1],
122        tupletSpecifier=abjadext.rmakers.TupletSpecifier(
123            trivialize=True,
124            extractTrivial=True,
125            rewriteRestFilled=True,
126            ),
127        )

```

```

123
124 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
125     talea=abjadext.rmakers.Talea(
126         counts=[1, 1, 1, 2, 1, 3, 1, 2, 3],
127         denominator=16,
128     ),
129     beamSpecifier=abjadext.rmakers.BeamSpecifier(
130         beamDivisionsTogether=True,
131         beamRests=False,
132     ),
133     extraCountsPerDivision=[1, 0, -1, 0, 1],
134     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
135         trivialize=True,
136         extractTrivial=True,
137         rewriteRestFilled=True,
138     ),
139 )
140
141 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
142     denominators=[8, 8, 16, 8, 8, 16],
143     extraCountsPerDivision=[0, 1, 0, 0, -1, 0, 1, -1],
144     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
145         trivialize=True,
146         extractTrivial=True,
147         rewriteRestFilled=True,
148     ),
149 )
150
151 attachmentHandler_one = AttachmentHandler(
152     startingDynamic='p',
153     endingDynamic='mp',
154     hairpinIndicator='--',
155     articulation='accent',
156 )
157
158 attachmentHandler_two = AttachmentHandler(
159     startingDynamic='fff',
160     endingDynamic='mf',
161     hairpinIndicator='>',
162     articulation='tenuto',
163 )
164
165 attachmentHandler_three = AttachmentHandler(
166     startingDynamic='mp',
167     endingDynamic='ff',
168     hairpinIndicator='<|',
169     articulation='',
170 )
171
172 #####cello#####
173 cellomusicmaker_one = MusicMaker(
174     rmaker=rmaker_one,
175     pitches=cello_notes_one,
176     continuous=True,

```

```

177     attachment_handler=attachment_handler_one,
178 )
179 cellomusicmaker_two = MusicMaker(
180     rmaker=rmaker_two,
181     pitches=cello_notes_two,
182     continuous=True,
183     attachment_handler=attachment_handler_two,
184 )
185 cellomusicmaker_three = MusicMaker(
186     rmaker=rmaker_three,
187     pitches=cello_notes_three,
188     continuous=True,
189     attachment_handler=attachment_handler_three,
190 )
191 cellomusicmaker_four = MusicMaker(
192     rmaker=rmaker_two,
193     pitches=cello_notes_four,
194     continuous=True,
195     attachment_handler=attachment_handler_three,
196 )
197
198 silence_maker = abjadext.rmakers.NoteRhythmMaker(
199     division_masks=[
200         abjadext.rmakers.SilenceMask(
201             pattern=abjad.index([0], 1),
202             ),
203         ],
204     )
205
206 bowmaker = MusicMaker(
207     pitches=[33, ],
208     rmaker=rmaker_two,
209     continuous=True,
210 )
211
212 class MusicSpecifier:
213
214     def __init__(self, music_maker, voice_name):
215         self.music_maker = music_maker
216         self.voice_name = voice_name
217
218     print('Collecting timespans and rmakers ...')
219     ###group one###
220 voice_1_timespan_list = abjad.TimespanList([
221     abjad.AnnotatedTimespan(
222         start_offset=start_offset,
223         stop_offset=stop_offset,
224         annotation=MusicSpecifier(
225             music_maker=music_maker,
226             voice_name='Voice 1',
227             ),
228     )
229     for start_offset, stop_offset, music_maker in [
230         [(0, 4), (4, 4), bowmaker],

```

```

231     [(4, 4), (7, 4), bowmaker],
232     [(12, 4), (15, 4), bowmaker],
233     [(15, 4), (17, 4), bowmaker],
234     [(17, 4), (20, 4), bowmaker],
235     [(23, 4), (25, 4), bowmaker],
236     [(25, 4), (27, 4), bowmaker],
237     [(27, 4), (30, 4), bowmaker],
238     [(32, 4), (36, 4), bowmaker],
239     [(43, 4), (44, 4), bowmaker],
240     [(44, 4), (48, 4), bowmaker],
241     [(48, 4), (51, 4), bowmaker],
242     [(52, 4), (56, 4), bowmaker],
243     [(56, 4), (58, 4), bowmaker],
244     [(62, 4), (64, 4), bowmaker],
245     [(68, 4), (72, 4), bowmaker],
246     [(72, 4), (76, 4), bowmaker],
247     [(76, 4), (78, 4), bowmaker],
248     [(78, 4), (81, 4), bowmaker],
249     [(82, 4), (84, 4), bowmaker],
250     [(84, 4), (87, 4), bowmaker],
251     [(88, 4), (91, 4), bowmaker],
252     [(91, 4), (93, 4), bowmaker],
253     [(94, 4), (99, 4), bowmaker],
254     [(100, 4), (103, 4), bowmaker],
255     [(103, 4), (105, 4), bowmaker],
256     [(106, 4), (110, 4), bowmaker],
257     [(110, 4), (111, 4), bowmaker],
258     [(112, 4), (114, 4), bowmaker],
259     [(114, 4), (119, 4), bowmaker],
260     [(122, 4), (126, 4), bowmaker],
261     [(128, 4), (131, 4), bowmaker],
262     [(132, 4), (134, 4), bowmaker],
263     [(139, 4), (140, 4), bowmaker],
264     [(144, 4), (146, 4), bowmaker],
265     [(146, 4), (149, 4), bowmaker],
266     [(150, 4), (153, 4), bowmaker],
267     [(157, 4), (158, 4), bowmaker],
268     [(158, 4), (162, 4), bowmaker],
269     [(165, 4), (167, 4), bowmaker],
270     [(167, 4), (169, 4), bowmaker],
271     [(174, 4), (176, 4), bowmaker],
272     [(176, 4), (177, 4), bowmaker],
273     [(181, 4), (185, 4), bowmaker],
274     [(185, 4), (186, 4), bowmaker],
275
276     ]
277   ])
278
279 voice_2_timespan_list = abjad.TimespanList([
280     abjad.AnnotatedTimespan(
281         start_offset=start_offset,
282         stop_offset=stop_offset,
283         annotation=MusicSpecifier(
284             music_maker=music_maker,

```

```

285         voice_name='Voice 2',
286     ),
287 )
288 for start_offset, stop_offset, music_maker in [
289     [(0, 4), (4, 4), cellomusicmaker_two],
290     [(4, 4), (7, 4), cellomusicmaker_one],
291     [(12, 4), (15, 4), cellomusicmaker_two],
292     [(15, 4), (17, 4), cellomusicmaker_one],
293     [(17, 4), (20, 4), cellomusicmaker_two],
294     [(23, 4), (25, 4), cellomusicmaker_two],
295     [(25, 4), (27, 4), cellomusicmaker_one],
296     [(27, 4), (30, 4), cellomusicmaker_two],
297     [(32, 4), (36, 4), cellomusicmaker_three],
298     [(43, 4), (44, 4), cellomusicmaker_two],
299     [(44, 4), (48, 4), cellomusicmaker_two],
300     [(48, 4), (51, 4), cellomusicmaker_one],
301     [(52, 4), (56, 4), cellomusicmaker_one],
302     [(56, 4), (58, 4), cellomusicmaker_two],
303     [(62, 4), (64, 4), cellomusicmaker_two],
304     [(68, 4), (72, 4), cellomusicmaker_three],
305     [(72, 4), (76, 4), cellomusicmaker_two],
306     [(76, 4), (78, 4), cellomusicmaker_three],
307     [(78, 4), (81, 4), cellomusicmaker_two],
308     [(82, 4), (84, 4), cellomusicmaker_two],
309     [(84, 4), (87, 4), cellomusicmaker_four],#
310     [(88, 4), (91, 4), cellomusicmaker_four],
311     [(91, 4), (93, 4), cellomusicmaker_one],
312     [(94, 4), (99, 4), cellomusicmaker_three],
313     [(100, 4), (103, 4), cellomusicmaker_one],
314     [(103, 4), (105, 4), cellomusicmaker_one],
315     [(106, 4), (110, 4), cellomusicmaker_four],
316     [(110, 4), (111, 4), cellomusicmaker_four],
317     [(112, 4), (114, 4), cellomusicmaker_three],
318     [(114, 4), (119, 4), cellomusicmaker_three],
319     [(122, 4), (126, 4), cellomusicmaker_one],
320     [(128, 4), (131, 4), cellomusicmaker_three],
321     [(132, 4), (134, 4), cellomusicmaker_four],
322     [(139, 4), (140, 4), cellomusicmaker_four],
323     [(144, 4), (146, 4), cellomusicmaker_four],
324     [(146, 4), (149, 4), cellomusicmaker_four],
325     [(150, 4), (153, 4), cellomusicmaker_four],#
326     [(157, 4), (158, 4), cellomusicmaker_two],
327     [(158, 4), (162, 4), cellomusicmaker_three],
328     [(165, 4), (167, 4), cellomusicmaker_two],
329     [(167, 4), (169, 4), cellomusicmaker_two],
330     [(174, 4), (176, 4), cellomusicmaker_three],
331     [(176, 4), (177, 4), cellomusicmaker_one],
332     [(181, 4), (185, 4), cellomusicmaker_two],
333     [(185, 4), (186, 4), cellomusicmaker_three],
334   ]
335 ])
336
337 ###group two###
338 voice_3_timespan_list = abjad.TimespanList([

```

```

339 abjad.AnnotatedTimespan(
340     start_offset=start_offset,
341     stop_offset=stop_offset,
342     annotation=MusicSpecifier(
343         music_maker=music_maker,
344         voice_name='Voice 3',
345     ),
346 )
347 for start_offset, stop_offset, music_maker in [
348     [(0, 4), (3, 4), bowmaker],
349     [(3, 4), (4, 4), bowmaker],
350     [(4, 4), (5, 4), bowmaker],
351     [(8, 4), (9, 4), bowmaker],
352     [(9, 4), (12, 4), bowmaker],
353     [(12, 4), (15, 4), bowmaker],
354     [(20, 4), (23, 4), bowmaker],
355     [(25, 4), (27, 4), bowmaker],
356     [(27, 4), (29, 4), bowmaker],
357     [(34, 4), (36, 4), bowmaker],
358     [(36, 4), (40, 4), bowmaker],
359     [(40, 4), (43, 4), bowmaker],
360     [(48, 4), (51, 4), bowmaker],
361     [(52, 4), (56, 4), bowmaker],
362     [(58, 4), (60, 4), bowmaker],
363     [(60, 4), (64, 4), bowmaker],
364     [(64, 4), (66, 4), bowmaker],
365     [(72, 4), (76, 4), bowmaker],
366     [(76, 4), (79, 4), bowmaker],
367     [(79, 4), (81, 4), bowmaker],
368     [(81, 4), (82, 4), bowmaker],
369     [(83, 4), (84, 4), bowmaker],
370     [(84, 4), (88, 4), bowmaker],
371     [(88, 4), (89, 4), bowmaker],
372     [(90, 4), (91, 4), bowmaker],
373     [(91, 4), (94, 4), bowmaker],
374     [(94, 4), (96, 4), bowmaker],
375     [(97, 4), (99, 4), bowmaker],
376     [(99, 4), (103, 4), bowmaker],
377     [(104, 4), (106, 4), bowmaker],
378     [(106, 4), (110, 4), bowmaker],
379     [(111, 4), (114, 4), bowmaker],
380     [(115, 4), (117, 4), bowmaker],
381     [(119, 4), (122, 4), bowmaker],
382     [(125, 4), (127, 4), bowmaker],
383     [(127, 4), (129, 4), bowmaker],
384     [(133, 4), (136, 4), bowmaker],
385     [(136, 4), (138, 4), bowmaker],
386     [(143, 4), (146, 4), bowmaker],
387     [(146, 4), (150, 4), bowmaker],
388     [(150, 4), (154, 4), bowmaker],
389     [(154, 4), (155, 4), bowmaker],
390     [(157, 4), (158, 4), bowmaker],
391     [(158, 4), (160, 4), bowmaker],
392     [(164, 4), (167, 4), bowmaker],

```

```

393     [(167, 4), (169, 4), bowmaker],
394     [(171, 4), (172, 4), bowmaker],
395     [(172, 4), (174, 4), bowmaker],
396     [(178, 4), (180, 4), bowmaker],
397     [(180, 4), (183, 4), bowmaker],
398     [(185, 4), (189, 4), bowmaker],
399
400 ]
401 ])
402
403 voice_4_timespan_list = abjad.TimespanList([
404     abjad.AnnotatedTimespan(
405         start_offset=start_offset,
406         stop_offset=stop_offset,
407         annotation=MusicSpecifier(
408             music_maker=music_maker,
409             voice_name='Voice 4',
410         ),
411     )
412     for start_offset, stop_offset, music_maker in [
413         [(0, 4), (3, 4), cellomusicmaker_one],
414         [(3, 4), (4, 4), cellomusicmaker_two],
415         [(4, 4), (5, 4), cellomusicmaker_one],
416         [(8, 4), (9, 4), cellomusicmaker_one],
417         [(9, 4), (12, 4), cellomusicmaker_three],
418         [(12, 4), (15, 4), cellomusicmaker_one],
419         [(20, 4), (23, 4), cellomusicmaker_two],
420         [(25, 4), (27, 4), cellomusicmaker_one],
421         [(27, 4), (29, 4), cellomusicmaker_one],
422         [(34, 4), (36, 4), cellomusicmaker_two],
423         [(36, 4), (40, 4), cellomusicmaker_one],
424         [(40, 4), (43, 4), cellomusicmaker_two],
425         [(48, 4), (51, 4), cellomusicmaker_two],
426         [(52, 4), (56, 4), cellomusicmaker_two],
427         [(58, 4), (60, 4), cellomusicmaker_one],
428         [(60, 4), (64, 4), cellomusicmaker_one],
429         [(64, 4), (66, 4), cellomusicmaker_three],
430         [(72, 4), (76, 4), cellomusicmaker_two],
431         [(76, 4), (79, 4), cellomusicmaker_one],
432         [(79, 4), (81, 4), cellomusicmaker_one],
433         [(81, 4), (82, 4), cellomusicmaker_three],
434         [(83, 4), (84, 4), cellomusicmaker_two],
435         [(84, 4), (88, 4), cellomusicmaker_two],
436         [(88, 4), (89, 4), cellomusicmaker_one],
437         [(90, 4), (91, 4), cellomusicmaker_one],
438         [(91, 4), (94, 4), cellomusicmaker_three],
439         [(94, 4), (96, 4), cellomusicmaker_two],
440         [(97, 4), (99, 4), cellomusicmaker_two],
441         [(99, 4), (103, 4), cellomusicmaker_one],
442         [(104, 4), (106, 4), cellomusicmaker_one],
443         [(106, 4), (110, 4), cellomusicmaker_three],
444         [(111, 4), (114, 4), cellomusicmaker_two],
445         [(115, 4), (117, 4), cellomusicmaker_four],#
446         [(119, 4), (122, 4), cellomusicmaker_four],

```

```

447     [(125, 4), (127, 4), cellomusicmaker_four],
448     [(127, 4), (129, 4), cellomusicmaker_four],
449     [(133, 4), (136, 4), cellomusicmaker_four],
450     [(136, 4), (138, 4), cellomusicmaker_four],
451     [(143, 4), (146, 4), cellomusicmaker_four],
452     [(146, 4), (150, 4), cellomusicmaker_four],
453     [(150, 4), (154, 4), cellomusicmaker_four],#
454     [(154, 4), (155, 4), cellomusicmaker_one],
455     [(157, 4), (158, 4), cellomusicmaker_three],
456     [(158, 4), (160, 4), cellomusicmaker_three],
457     [(164, 4), (167, 4), cellomusicmaker_two],
458     [(167, 4), (169, 4), cellomusicmaker_two],
459     [(171, 4), (172, 4), cellomusicmaker_three],
460     [(172, 4), (174, 4), cellomusicmaker_one],
461     [(178, 4), (180, 4), cellomusicmaker_one],
462     [(180, 4), (183, 4), cellomusicmaker_two],
463     [(185, 4), (189, 4), cellomusicmaker_two],
464     [(189, 4), (190, 4), silence_maker],
465   ]
466 ])
467
468 all_timespan_lists = {
469   'Voice 1': voice_1_timespan_list,
470   'Voice 2': voice_2_timespan_list,
471   'Voice 3': voice_3_timespan_list,
472   'Voice 4': voice_4_timespan_list,
473 }
474
475 global_timespan = abjad.Timespan(
476   start_offset=0,
477   stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
478 )
479
480 for voice_name, timespan_list in all_timespan_lists.items():
481   silences = abjad.TimespanList([global_timespan])
482   silences.extend(timespan_list)
483   silences.sort()
484   silences.compute_logical_xor()
485   for silence_timespan in silences:
486     timespan_list.append(
487       abjad.AnnotatedTimespan(
488         start_offset=silence_timespan.start_offset,
489         stop_offset=silence_timespan.stop_offset,
490         annotation=MusicSpecifier(
491           music_maker=None,
492           voice_name=voice_name,
493         ),
494       )
495     )
496   timespan_list.sort()
497
498 for voice_name, timespan_list in all_timespan_lists.items():
499   shards = timespan_list.split_at_offsets(bounds)
500   split_timespan_list = abjad.TimespanList()

```

```

501     for shard in shards:
502         split_timespan_list.extend(shard)
503     split_timespan_list.sort()
504     all_timespan_lists[voice_name] = timespan_list
505
506 score = abjad.Score([
507     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context'),
508     abjad.StaffGroup(
509         [
510             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
511             lilypond_type='BowStaff',),
512             abjad.Staff([abjad.Voice(name='Voice 5')], name='Staff 5',
513             lilypond_type='BeamStaff',),
514             abjad.Staff([abjad.Voice(name='Voice 2')], name='Staff 2',
515             lilypond_type='Staff',),
516             ],
517             name='Staff Group 1',
518         ),
519         abjad.StaffGroup(
520             [
521                 abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
522                 lilypond_type='BowStaff',),
523                 abjad.Staff([abjad.Voice(name='Voice 6')], name='Staff 6',
524                 lilypond_type='BeamStaff',),
525                 abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
526                 lilypond_type='Staff',),
527                 ],
528                 name='Staff Group 2',
529             )
530         ],
531     )
532
533 for time_signature in time_signatures:
534     skip = abjad.Skip(1, multiplier=(time_signature))
535     abjad.attach(time_signature, skip)
536     score['Global Context'].append(skip)
537
538 print('Making containers ...')
539
540 def make_container(music_maker, durations):
541     selections = music_maker(durations)
542     container = abjad.Container([])
543     container.extend(selections)
544     return container
545
546 def key_function(timespan):
547     return timespan.annotation.music_maker or silence_maker
548
549 for voice_name, timespan_list in all_timespan_lists.items():
550     for music_maker, grouper in itertools.groupby(
551         timespan_list,
552         key=key_function,
553     ):
554         durations = [timespan.duration for timespan in grouper]

```

```

549     container = make_container(music_maker, durations)
550     voice = score[voice_name]
551     voice.append(container)
552
553 print('Adding Beam Staff ...')
554 voice_1_copy = abjad.mutate(score['Voice 1']).copy()
555 score['Voice 5'].extend([voice_1_copy[:]])
556
557 voice_3_copy = abjad.mutate(score['Voice 3']).copy()
558 score['Voice 6'].extend([voice_3_copy[:]])
559
560 print('Splitting and rewriting ...')
561
562 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
563     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
564         time_signature = time_signatures[i]
565         abjad.mutate(shard).rewrite_meter(time_signature)
566
567 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
568     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
569         time_signature = time_signatures[i]
570         abjad.mutate(shard).rewrite_meter(time_signature)
571
572 print('Beaming runs ...')
573
574 for voice in abjad.select(score).components(abjad.Voice):
575     for run in abjad.select(voice).runs():
576         if 1 < len(run):
577             specifier = abjadext.rmakers.BeamSpecifier(
578                 beam_each_division=True,
579                 )
580             specifier(abjad.select(run))
581             abjad.attach(abjad.StartBeam(), run[0])
582             abjad.attach(abjad.StopBeam(), run[-1])
583
584 print('Stopping Hairpins ...')
585 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
586     for rest in abjad.iterate(staff).components(abjad.Rest):
587         previous_leaf = abjad.inspect(rest).leaf(-1)
588         if isinstance(previous_leaf, abjad.Note):
589             abjad.attach(abjad.StopHairpin(), rest)
590         elif isinstance(previous_leaf, abjad.Chord):
591             abjad.attach(abjad.StopHairpin(), rest)
592         elif isinstance(previous_leaf, abjad.Rest):
593             pass
594
595 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
596     for rest in abjad.iterate(staff).components(abjad.Rest):
597         previous_leaf = abjad.inspect(rest).leaf(-1)
598         if isinstance(previous_leaf, abjad.Note):
599             abjad.attach(abjad.StopHairpin(), rest)
600         elif isinstance(previous_leaf, abjad.Chord):
601             abjad.attach(abjad.StopHairpin(), rest)
602         elif isinstance(previous_leaf, abjad.Rest):
603             pass

```

```

603         pass
604
605
606 print('Adding attachments ...')
607 bar_line = abjad.BarLine('|.')
608 section_bar_line = abjad.BarLine('||')
609 metro = abjad.MetronomeMark((1, 8), 60)
610 markup1 = abjad.Markup(r'\bold { A }')
611 markup2 = abjad.Markup(r'\bold { B }')
612 markup3 = abjad.Markup(r'\bold { C }')
613 markup4 = abjad.Markup(r'\bold { D }')
614 markup5 = abjad.Markup(r'\bold { E }')
615 markup6 = abjad.Markup(r'\bold { F }')
616 mark1 = abjad.RehearsalMark(markup=markup1)
617 mark2 = abjad.RehearsalMark(markup=markup2)
618 mark3 = abjad.RehearsalMark(markup=markup3)
619 mark4 = abjad.RehearsalMark(markup=markup4)
620 mark5 = abjad.RehearsalMark(markup=markup5)
621 mark6 = abjad.RehearsalMark(markup=markup6)
622
623 def _apply_numerators_and_tech(staff, nums, tech):
624     numerators = cyc(nums)
625     techs = cyc(tech)
626     for logical_tie in abjad.select(staff).logical_ties(pitched=True):
627         tech = next.techs)
628         numerator = next(numerators)
629         bcp = abjad.BowContactPoint((numerator, 5))
630         technis = abjad.BowMotionTechnique(tech)
631         for note in logical_tie:
632             abjad.attach(bcp, note)
633             abjad.attach(technis, note)
634     for run in abjad.select(staff).runs():
635         abjad.bow_contact_spanner(run, omit_bow_changes=False)
636
637 for voice in abjad.select(score['Voice 1']).components(abjad.Voice):
638     seed(4)
639     nums_random_walk = []
640     nums_random_walk.append(-1 if random() < 0.5 else 1)
641     for i in range(1, 1000):
642         movement = -1 if random() < 0.5 else 1
643         value = nums_random_walk[i-1] + movement
644         nums_random_walk.append(value)
645     nums_random_walk = [abs(x) for x in nums_random_walk]
646     nums_chord = [0, 5, 3, 1, 4, 2, 5, 4, 3, 2]
647     num_list = [nums_chord[x] for x in reduceMod9(nums_random_walk)]
648     tech_list = ['ordinario', 'ordinario', 'ordinario', 'ordinario', 'circular',
649                 'circular', 'ordinario', 'ordinario', 'ordinario', 'jete', 'ordinario',
650                 'ordinario', 'ordinario', 'ordinario', 'jete', 'jete', 'jete']
651     _apply_numerators_and_tech(staff=voice, nums=num_list, tech=tech_list)
652
653 for voice in abjad.select(score['Voice 3']).components(abjad.Voice):
654     seed(5)
655     nums_random_walk = []

```

```

654     nums_random_walk.append(-1 if random() < 0.5 else 1)
655     for i in range(1, 1000):
656         movement = -1 if random() < 0.5 else 1
657         value = nums_random_walk[i-1] + movement
658         nums_random_walk.append(value)
659     nums_random_walk = [abs(x) for x in nums_random_walk]
660     nums_chord = [0, 1, 2, 3, 4, 5, 4, 3, 2, 1]
661     num_list = [nums_chord[x] for x in reduceMod9(nums_random_walk)]
662     tech_list = ['ordinario', 'ordinario', 'ordinario', 'ordinario', 'circular',
663                  ', 'circular', 'ordinario', 'ordinario', 'ordinario', 'jete', 'ordinario',
664                  ', 'ordinario', 'ordinario', 'ordinario', 'jete', 'jete', 'jete',
665                  'jete', 'jete',]
666     _apply_numerators_and_tech(staff=voice, nums=num_list, tech=tech_list)
667
668
669 def _apply_position_and_span(staff, poses):
670     positions = cyc(poses)
671     for run in abjad.select(staff).runs():
672         span = abjad.StartTextSpan(
673             left_text=abjad.Markup(next(positions)).upright(),
674             right_text=abjad.Markup(next(positions)).upright(),
675             style='dashed-line-with-arrow',
676             )
677         abjad.attach(span, run[0])
678         abjad.attach(abjad.StopTextSpan(), run[-1])
679         abjad.override(staff).text_spinner.staff_padding = 0
680
681     for voice in abjad.select(score['Voice 5']).components(abjad.Voice):
682         pos_list_1 = ['st.', 'ord.', 'sp.', 'msp.', 'ord.',]
683         _apply_position_and_span(staff=voice, poses=pos_list_1)
684
685     for voice in abjad.select(score['Voice 6']).components(abjad.Voice):
686         pos_list_2 = ['sp.', 'msp.', 'ord.', 'st.', 'ord.',]
687         _apply_position_and_span(staff=voice, poses=pos_list_2)
688
689     for voice in abjad.select(score['Voice 1']).components(abjad.Voice):
690         for run in abjad.select(voice).runs():
691             specifier = abjadext.rmakers.BeamSpecifier(
692                 beam_each_division=False,
693                 )
694             specifier(run)
695
696     for voice in abjad.select(score['Voice 3']).components(abjad.Voice):
697         for run in abjad.select(voice).runs():
698             specifier = abjadext.rmakers.BeamSpecifier(
699                 beam_each_division=False,
700                 )
701             specifier(run)
702
703 instruments1 = cyc([
704     abjad.Cello(),
705 ])
706
707 instruments2 = cyc([
708     abjad.Cello(),
709 ])

```

```

705 ])
706
707 clefs1 = cyc([
708     abjad.Clef('percussion'),
709     abjad.Clef('percussion'),
710     abjad.Clef('bass'),
711 ])
712
713 clefs2 = cyc([
714     abjad.Clef('percussion'),
715     abjad.Clef('percussion'),
716     abjad.Clef('bass'),
717 ])
718
719 abbreviations1 = cyc([
720     abjad.MarginMarkup(markup=abjad.Markup('B.H.'),),
721     abjad.MarginMarkup(markup=abjad.Markup('vc.I'),),
722     abjad.MarginMarkup(markup=abjad.Markup('L.H.'),),
723 ])
724
725 abbreviations2 = cyc([
726     abjad.MarginMarkup(markup=abjad.Markup('B.H.'),),
727     abjad.MarginMarkup(markup=abjad.Markup('vc.II'),),
728     abjad.MarginMarkup(markup=abjad.Markup('L.H.'),),
729 ])
730
731 names1 = cyc([
732     abjad.StartMarkup(markup=abjad.Markup('Bow Hand'),),
733     abjad.StartMarkup(markup=abjad.Markup('Violoncello I'),),
734     abjad.StartMarkup(markup=abjad.Markup('Left Hand'),),
735 ])
736
737 names2 = cyc([
738     abjad.StartMarkup(markup=abjad.Markup('Bow Hand'),),
739     abjad.StartMarkup(markup=abjad.Markup('Violoncello II'),),
740     abjad.StartMarkup(markup=abjad.Markup('Left Hand'),),
741 ])
742
743 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
744     leaf1 = abjad.select(staff).leaves()[0]
745     abjad.attach(next(instruments1), leaf1)
746     abjad.attach(next(abbreviations1), leaf1)
747     abjad.attach(next(names1), leaf1)
748     abjad.attach(next(clefs1), leaf1)
749
750 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
751     leaf1 = abjad.select(staff).leaves()[0]
752     abjad.attach(next(instruments2), leaf1)
753     abjad.attach(next(abbreviations2), leaf1)
754     abjad.attach(next(names2), leaf1)
755     abjad.attach(next(clefs2), leaf1)
756
757 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
758     leaf1 = abjad.select(staff).leaves()[0]

```

```

759     last_leaf = abjad.select(staff).leaves()[-1]
760     abjad.attach(metro, leaf1)
761     abjad.attach(bar_line, last_leaf)
762
763 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff)[0]:
764     leaf1 = abjad.select(staff).leaves()[0]
765     last_leaf = abjad.select(staff).leaves()[-1]
766     abjad.attach(metro, leaf1)
767     abjad.attach(bar_line, last_leaf)
768
769 for staff in abjad.iterate(score['Global Context']).components(abjad.Staff):
770     leaf1_start = abjad.select(staff).leaves()[7]
771     leaf1 = abjad.select(staff).leaves()[8]
772     abjad.attach(mark1, leaf1)
773     abjad.attach(section_bar_line, leaf1_start)
774
775 for staff in abjad.iterate(score['Global Context']).components(abjad.Staff):
776     leaf2_start = abjad.select(staff).leaves()[15]
777     leaf2 = abjad.select(staff).leaves()[16]
778     abjad.attach(mark2, leaf2)
779     abjad.attach(section_bar_line, leaf2_start)
780
781 for staff in abjad.iterate(score['Global Context']).components(abjad.Staff):
782     leaf3_start = abjad.select(staff).leaves()[23]
783     leaf3 = abjad.select(staff).leaves()[24]
784     abjad.attach(mark3, leaf3)
785     abjad.attach(section_bar_line, leaf3_start)
786
787 for staff in abjad.iterate(score['Global Context']).components(abjad.Staff):
788     leaf4_start = abjad.select(staff).leaves()[31]
789     leaf4 = abjad.select(staff).leaves()[32]
790     abjad.attach(mark4, leaf4)
791     abjad.attach(section_bar_line, leaf4_start)
792
793 for staff in abjad.iterate(score['Global Context']).components(abjad.Staff):
794     leaf5_start = abjad.select(staff).leaves()[38]
795     leaf5 = abjad.select(staff).leaves()[39]
796     abjad.attach(mark5, leaf5)
797     abjad.attach(section_bar_line, leaf5_start)
798
799 score_file = abjad.LilyPondFile.new(
800     score,
801     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
802     _stylesheets/abjad.ily'],
803 )
804
805 abjad.SegmentMaker.comment_measure_numbers(score)
806 ######
807
808 directory = '/Users/evansdsg2/Scores/cthar/cthar/Segments/Segment_I'
809 pdf_path = f'{directory}/Segment_I.pdf'
810 path = pathlib.Path('Segment_I.pdf')
811 if path.exists():
812     print(f'Removing {pdf_path} ...')

```

```

812     path.unlink()
813 time_1 = time.time()
814 print(f'Persisting {pdf_path} ...')
815 result = abjad.persist(score_file).as_pdf(pdf_path)
816 print(result[0])
817 print(result[1])
818 print(result[2])
819 success = result[3]
820 if success is False:
821     print('LilyPond failed!')
822 time_2 = time.time()
823 total_time = time_2 - time_1
824 print(f'Total time: {total_time} seconds')
825 if path.exists():
826     print(f'Opening {pdf_path} ...')
827     os.system(f'open {pdf_path}')

```

**Code Example A.9:** Cthar Segment\_I

## A.2.2 STYLESHEET

```

1 Cthar Stylesheet.
2 % 2018-07-17 19:54
3
4 \version "2.19.82"
5 \language "english"
6 #(set-default-paper-size "letterlandscape")
7 #(set-global-staff-size 10)
8 \include "ekmel.ily"
9 \ekmelicStyle evans
10
11 \header {
12     tagline = ##f
13     breakbefore = ##t
14     title = \markup \override #'(
15         font-name . "Didot"
16     ) \fontsize #15 \bold \center-column {
17         "Cthar"
18     }
19     subtitle = \markup \override #'(
20         font-name . "Didot"

```

```

21         ) \fontsize #4 \center-column {
22             "for two cellos"
23         }
24     arranger = \markup \override #'(
25         font-name . "Didot"
26     ) \fontsize #2.5 {
27         "Gregory Rowland Evans"
28     }
29 }
30
31 bowtab = {
32     \override Staff.Clef.stencil = #ly:text-interface::print
33     \override Staff.Clef.text = \markup { \general-align #Y #0.03
34     \epsfile #Y #10 #"bow_position_tablature.eps"
35 }
36 }
37
38 \layout {
39     \accidentalStyle forget
40     %\accidentalStyle modern
41     %\accidentalStyle modern-cautionary
42     %\accidentalStyle neo-modern
43     %\accidentalStyle dodecaphonic
44     indent = #5
45     %ragged-last = ##t
46     ragged-right = ##t
47     %left-margin = #15
48     \context {
49         \name TimeSignatureContext
50         \type Engraver_group
51         \numericTimeSignature
52         \consists Axis_group_engraver
53         \consists Bar_number_engraver

```

```

54      \consists Time_signature_engraver
55
56      \consists Mark_engraver
57
58      \consists Metronome_mark_engraver
59      \override BarNumber.Y-extent = #'(0 . 0)
60      \override BarNumber.Y-offset = 0
61      \override BarNumber.extra-offset = #'(-4 . 0)
62      %\override BarNumber.font-name = "Didot"
63      \override BarNumber.stencil = #(
64          make-stencil-boxer 0.1 0.7 ly:text-interface::print
65          )
66
67      \override BarNumber.font-size = 1
68      \override BarNumber.padding = 4
69      \override MetronomeMark.X-extent = #'(0 . 0)
70      \override MetronomeMark.Y-extent = #'(0 . 0)
71      \override MetronomeMark.break-align-symbols = #'(left-edge)
72      \override MetronomeMark.extra-offset = #'(0 . 4)
73      \override MetronomeMark.font-size = 10
74      \override RehearsalMark.stencil = #(
75          make-stencil-circler 0.1 0.7 ly:text-interface::print
76          )
77
78      \override RehearsalMark.X-extent = #'(0 . 0)
79      \override RehearsalMark.X-offset = 6
80      \override RehearsalMark.Y-offset = -2.25
81      \override RehearsalMark.break-align-symbols = #'(time-signature)
82      \override RehearsalMark.break-visibility = #end-of-line-invisible
83      \override RehearsalMark.font-name = "Didot"
84      \override RehearsalMark.font-size = 8
85      \override RehearsalMark.outside-staff-priority = 500
86      \override RehearsalMark.self-alignment-X = #center
87          \override TimeSignature.X-extent = #'(0 . 0)
88          \override TimeSignature.X-offset = #ly:self-alignment-interface::x-
89          aligned-on-self
90
91          \override TimeSignature.Y-extent = #'(0 . 0)

```

```

86   \override TimeSignature.Y-offset = 3
87
88   \override TimeSignature.break-align-symbol = ##f
89
90   \override TimeSignature.break-visibility = #end-of-line-invisible
91
92   \override TimeSignature.font-size = #7
93
94   \override TimeSignature.self-alignment-X = #center
95
96   \override VerticalAxisGroup.default-staff-staff-spacing = #'(
97     (basic-distance . 0) (minimum-distance . 10) (padding . 6) (stretchability
98     . 0)
99
100  )
101
102  }
103
104
105  \context {
106    \Score
107
108    \remove Bar_number_engraver
109
110    \remove Mark_engraver
111
112    \accepts TimeSignatureContext
113
114    \accepts LipStaff
115
116    \override BarLine.bar-extent = #'(-2 . 2)
117
118    \override Beam.breakable = ##t
119
120    \override Beam.concaveness = #10000
121
122    \override Glissando.breakable = ##t
123
124    \override MetronomeMark.font-size = 5
125
126    \override SpacingSpanner.strict-grace-spacing = ##t
127
128    \override SpacingSpanner.strict-note-spacing = ##t
129
130    \override SpacingSpanner.uniform-stretching = ##t
131
132    \override StaffGrouper.staff-staff-spacing = #'(
133      (basic-distance . 0) (minimum-distance . 6) (padding . 2)
134
135      )
136
137    \override TupletBracket.bracket-visibility = ##t
138
139    \override TupletBracket.minimum-length = #3
140
141    \override TupletBracket.padding = #2
142
143    \override TupletBracket.springs-and-rods = #ly:spanner::set-spacing-
144      rods
145
146    \override TupletNumber.text = #tuple-number::calc-fraction-text

```

```

117   \override TextSpanner.Y-offset = 1
118   proportionalNotationDuration = #(ly:make-moment 1 50)
119   autoBeaming = ##f
120   tupletFullLength = ##t
121 }
122
123 \context {
124   \Voice
125   \remove Forbid_line_break_engraver
126 }
127
128 \context {
129   \Staff
130   \remove Time_signature_engraver
131 }
132
133 \context {
134   \Staff
135   \name BowStaff
136   \type Engraver_group
137   \alias Staff
138   \bowtab
139   \override Beam.stencil = ##f
140   \override Dots.stencil = ##f
141   \override Flag.stencil = ##f
142   \override Glissando.bound-details.left.padding = #0.5
143   \override Glissando.bound-details.right.padding = #0.5
144   \override Glissando.thickness = #2
145   \override NoteHead.Y-offset = #-5
146   \override NoteHead.extra-offset = #'(0.05 . 0)
147   \override NoteHead.stencil = ##f
148   \override Rest.transparent = ##t
149   \override Script.staff-padding = #2
150   \override StaffSymbol.transparent = ##t
151   \override Stem.direction = #down
152   \override Stem.stencil = ##f

```

```

150     \override TimeSignature.stencil = ##f
151 
152     \override Tie.stencil = ##f
153 
154     \override TupletBracket.stencil = ##f
155 
156     \override TupletNumber.stencil = ##f
157 
158     %\RemoveEmptyStaves
159 
160 }
161 
162 
163 
164 
165 
166 
167 
168 
169 
170 
171 
172 
173 
174 
175 
176 
177 
178 
179 
180 
181 
182

```

150       \override TimeSignature.stencil = ##f  
151  
152       \override Tie.stencil = ##f  
153  
154       \override TupletBracket.stencil = ##f  
155  
156       \override TupletNumber.stencil = ##f  
157  
158       %\RemoveEmptyStaves  
159  
160 }  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182

```

183     \remove Time-signature-engraver
184 }
185 
186     \context {
187         \StaffGroup
188 
189         \accepts BowStaff
190 
191     }
192 
193 \paper {
194 
195     top-margin = 1.5\cm
196 
197     bottom-margin = 1.5\cm
198 
199     %top-margin = .90\in
200 
201     oddHeaderMarkup = \markup ""
202 
203     evenHeaderMarkup = \markup ""
204 
205     oddFooterMarkup = \markup \fill-line {
206 
207         " "
208 
209         \concat {
210 
211             "Cthar ~"
212 
213             \fontsize #2
214 
215             \fromproperty #'page:page-number-string "~ Evans"
216 
217         }
218 
219         " "
220 
221     }
222 
223     evenFooterMarkup = \markup \fill-line {
224 
225         " "
226 
227         \concat { "Cthar ~" \fontsize #2
228 
229             \fromproperty #'page:page-number-string "~ Evans"
230 
231         } ""
232 
233     }

```

215 }

### Code Example A.10: Cthar Stylesheet

## A.3 TIANSHU (FOR 12 PLAYERS) SOURCE CODE

### A.3.1 SEGMENTS

#### A.3.1.1 SEGMENT\_I

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (5, 4), (2, 4), (4, 4), (3, 4), (4, 4), (4, 4),
17         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
18         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
19         (2, 4), (3, 4), (4, 4), (3, 4), (4, 4), (3, 4),
20         (5, 4), (3, 4), (3, 4), (4, 4), (3, 4), (3, 4),
21         (4, 4), (5, 4), (4, 4), (3, 4), (5, 4), (5, 4),
22         (5, 4), (5, 4), (4, 4), (4, 4), (5, 4), (5, 4),
23         (4, 4), (4, 4), (3, 4), (4, 4), (4, 4), (3, 4),
24         (5, 4),
25     ]
26 ]
27
28 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures
29 ])
30
31 def reduceMod3(rw):
32     return [(x % 4) for x in rw]
33
34 def reduceMod5(rw):
35     return [(x % 6) for x in rw]
36
37 def reduceMod7(rw):
38     return [(x % 8) for x in rw]
```

```

39 def reduceMod9(rw):
40     return [(x % 10) for x in rw]
41
42 def reduceMod11(rw):
43     return [(x % 12) for x in rw]
44
45 def reduceMod13(rw):
46     return [(x % 14) for x in rw]
47
48 def reduceMod15(rw):
49     return [(x % 16) for x in rw]
50
51 seed(1)
52 flute_random_walk_one = []
53 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
54 for i in range(1, 1000):
55     movement = -1 if random() < 0.5 else 1
56     value = flute_random_walk_one[i-1] + movement
57     flute_random_walk_one.append(value)
58 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
59 flute_chord_one = [8, 14, 23, 27, 28, 30, 37, 30, 28, 27, 23, 14, ]
60 flute_notes_one = [flute_chord_one[x] for x in reduceMod11(
61     flute_random_walk_one)]
62
63 seed(2)
64 clarinet_random_walk_one = []
65 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
66 for i in range(1, 1000):
67     movement = -1 if random() < 0.5 else 1
68     value = clarinet_random_walk_one[i-1] + movement
69     clarinet_random_walk_one.append(value)
70 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
71 clarinet_chord_one = [-3, 5, 8, 14, 23, 27, 23, 14, 8, 5, ]
72 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod9(
73     clarinet_random_walk_one)]
74
75 seed(3)
76 bassoon_random_walk_one = []
77 bassoon_random_walk_one.append(-1 if random() < 0.5 else 1)
78 for i in range(1, 1000):
79     movement = -1 if random() < 0.5 else 1
80     value = bassoon_random_walk_one[i-1] + movement
81     bassoon_random_walk_one.append(value)
82 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
83 bassoon_chord_one = [-24, -14, -3, 5, 8, 5, -3, -14, ]
84 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod7(
85     bassoon_random_walk_one)]
86
87 seed(4)
88 horn_random_walk_one = []
89 horn_random_walk_one.append(-1 if random() < 0.5 else 1)
90 for i in range(1, 1000):
91     movement = -1 if random() < 0.5 else 1
92     value = horn_random_walk_one[i-1] + movement

```

```

90     horn_random_walk_one.append(value)
91 horn_random_walk_one = [abs(x) for x in horn_random_walk_one]
92 horn_chord_one = [-24, -14, -3, 5, 8, 5, -3, -14, ]
93 horn_notes_one = [horn_chord_one[x] for x in reduceMod7(horn_random_walk_one)]
94
95 seed(5)
96 trumpet_random_walk_one = []
97 trumpet_random_walk_one.append(-1 if random() < 0.5 else 1)
98 for i in range(1, 1000):
99     movement = -1 if random() < 0.5 else 1
100    value = trumpet_random_walk_one[i-1] + movement
101    trumpet_random_walk_one.append(value)
102 trumpet_random_walk_one = [abs(x) for x in trumpet_random_walk_one]
103 trumpet_chord_one = [-3, 5, 8, 14, 23, 14, 8, 5, ]
104 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod7(
105     trumpet_random_walk_one)]
106
107 seed(6)
108 trombone_random_walk_one = []
109 trombone_random_walk_one.append(-1 if random() < 0.5 else 1)
110 for i in range(1, 1000):
111     movement = -1 if random() < 0.5 else 1
112     value = trombone_random_walk_one[i-1] + movement
113     trombone_random_walk_one.append(value)
114 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
115 trombone_chord_one = [-14, -3, 5, -3, ]
116 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod3(
117     trombone_random_walk_one)]
118
119 seed(7)
120 tuba_random_walk_one = []
121 tuba_random_walk_one.append(-1 if random() < 0.5 else 1)
122 for i in range(1, 1000):
123     movement = -1 if random() < 0.5 else 1
124     value = tuba_random_walk_one[i-1] + movement
125     tuba_random_walk_one.append(value)
126 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
127 tuba_chord_one = [-29, -24, -14, -3, 5, -3, -14, -24, ]
128 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod7(tuba_random_walk_one)]
129
130 seed(8)
131 violin1_random_walk_one = []
132 violin1_random_walk_one.append(-1 if random() < 0.5 else 1)
133 for i in range(1, 1000):
134     movement = -1 if random() < 0.5 else 1
135     value = violin1_random_walk_one[i-1] + movement
136     violin1_random_walk_one.append(value)
137 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
138 violin1_chord_one = [-3, 5, 8, 14, 23, 27, 28, 30, 37, 30, 28, 27, 23, 14, 8,
139     5, ]
139 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod15(
140     violin1_random_walk_one)]
141
142 seed(9)

```

```

140 violin2_random_walk_one = []
141 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
142 for i in range(1, 1000):
143     movement = -1 if random() < 0.5 else 1
144     value = violin2_random_walk_one[i-1] + movement
145     violin2_random_walk_one.append(value)
146 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
147 violin2_chord_one = [-3, 5, 8, 14, 23, 27, 28, 27, 23, 14, 8, 5, ]
148 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod11(
149     violin2_random_walk_one)]
150
150 seed(10)
151 viola_random_walk_one = []
152 viola_random_walk_one.append(-1 if random() < 0.5 else 1)
153 for i in range(1, 1000):
154     movement = -1 if random() < 0.5 else 1
155     value = viola_random_walk_one[i-1] + movement
156     viola_random_walk_one.append(value)
157 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
158 viola_chord_one = [-3, 5, 8, 14, 23, 27, 28, 27, 23, 14, 8, 5, ]
159 viola_notes_one = [viola_chord_one[x] for x in reduceMod11(
160     viola_random_walk_one)]
160
161 seed(11)
162 cello_random_walk_one = []
163 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
164 for i in range(1, 1000):
165     movement = -1 if random() < 0.5 else 1
166     value = cello_random_walk_one[i-1] + movement
167     cello_random_walk_one.append(value)
168 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
169 cello_chord_one = [-24, -14, -3, 5, 8, 14, 8, 5, -3, -14]
170 cello_notes_one = [cello_chord_one[x] for x in reduceMod9(
171     cello_random_walk_one)]
172
172 seed(12)
173 bass_random_walk_one = []
174 bass_random_walk_one.append(-1 if random() < 0.5 else 1)
175 for i in range(1, 1000):
176     movement = -1 if random() < 0.5 else 1
177     value = bass_random_walk_one[i-1] + movement
178     bass_random_walk_one.append(value)
179 bass_random_walk_one = [abs(x) for x in bass_random_walk_one]
180 bass_chord_one = [-29, -24, -14, -3, -14, -24, ]
181 bass_notes_one = [bass_chord_one[x] for x in reduceMod5(bass_random_walk_one)]
182
183 flute_scale = [30, 23, 5, 23, ]
184 clarinet_scale = [23, 5, ]
185 bassoon_scale = [-24, ]
186 horn_scale = [5, ]
187 trumpet_scale = [23, ]
188 trombone_scale = [5, ]
189 tuba_scale = [-24, ]
190 violin1_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25, 24.5,

```

```

24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5, 20, 20.5, 21,
21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26, 26.5, 27, 27.5, 28,
28.5, 29, 29.5, ]
191 violin2_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14, 13.5,
13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5, 10, 10.5,
11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16, 16.5, 17, 17.5, 18,
18.5, ]
192 viola_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1, 0.5,
0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1,
1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]
193 cello_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8, -8.5, -9,
-9.5, -10, -10.5, -11, -11.5, -12, -12.5 -13, -13.5 -14, -13.5, -13,
-12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8, -7.5, -7, -6.5,
-6, -5.5, -5, -4.5, -4, -3.5, ]
194 bass_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18, -18.5, -19,
-19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5, -24, -24.5, -25,
-24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21, -20.5, -20, -19.5, -19,
-18.5, -18, -17.5, -17, -16.5, -16, -15.5, -15, -14.5, ]
195
196 seed(1)
197 flute_random_walk_two = []
198 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
199 for i in range(1, 1000):
200     movement = -1 if random() < 0.5 else 1
201     value = flute_random_walk_two[i-1] + movement
202     flute_random_walk_two.append(value)
203 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
204 flute_chord_two = [10, 16, 23, 25, 26, 25, 23, 16, ]
205 flute_notes_two = [flute_chord_two[x] for x in reduceMod7(
    flute_random_walk_two)]
206
207 seed(2)
208 clarinet_random_walk_two = []
209 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
210 for i in range(1, 1000):
211     movement = -1 if random() < 0.5 else 1
212     value = clarinet_random_walk_two[i-1] + movement
213     clarinet_random_walk_two.append(value)
214 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
215 clarinet_chord_two = [-5, 5, 10, 16, 23, 25, 26, 25, 23, 16, 10, 5, ]
216 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod11(
    clarinet_random_walk_two)]
217
218 seed(3)
219 bassoon_random_walk_two = []
220 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
221 for i in range(1, 1000):
222     movement = -1 if random() < 0.5 else 1
223     value = bassoon_random_walk_two[i-1] + movement
224     bassoon_random_walk_two.append(value)
225 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
226 bassoon_chord_two = [-24, -16, -5, 5, -5, -16, ]
227 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod5(
    bassoon_random_walk_two)]
```

```

228
229 seed(4)
230 horn_random_walk_two = []
231 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
232 for i in range(1, 1000):
233     movement = -1 if random() < 0.5 else 1
234     value = horn_random_walk_two[i-1] + movement
235     horn_random_walk_two.append(value)
236 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
237 horn_chord_two = [-16, -5, 5, 10, 5, -5, ]
238 horn_notes_two = [horn_chord_two[x] for x in reduceMod5(horn_random_walk_two)]
239
240 seed(5)
241 trumpet_random_walk_two = []
242 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)
243 for i in range(1, 1000):
244     movement = -1 if random() < 0.5 else 1
245     value = trumpet_random_walk_two[i-1] + movement
246     trumpet_random_walk_two.append(value)
247 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]
248 trumpet_chord_two = [-5, 5, 10, 16, 23, 16, 10, 5, ]
249 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod7(
250     trumpet_random_walk_two)]
251
252 seed(6)
253 trombone_random_walk_two = []
254 trombone_random_walk_two.append(-1 if random() < 0.5 else 1)
255 for i in range(1, 1000):
256     movement = -1 if random() < 0.5 else 1
257     value = trombone_random_walk_two[i-1] + movement
258     trombone_random_walk_two.append(value)
259 trombone_random_walk_two = [abs(x) for x in trombone_random_walk_two]
260 trombone_chord_two = [-16, -5, 5, -5, ]
261 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod3(
262     trombone_random_walk_two)]
263
264 seed(7)
265 tuba_random_walk_two = []
266 tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
267 for i in range(1, 1000):
268     movement = -1 if random() < 0.5 else 1
269     value = tuba_random_walk_two[i-1] + movement
270     tuba_random_walk_two.append(value)
271 tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
272 tuba_chord_two = [-27, -24, -16, -5, -16, -24, ]
273 tuba_notes_two = [tuba_chord_two[x] for x in reduceMod5(tuba_random_walk_two)]
274
275 seed(8)
276 violin1_random_walk_two = []
277 violin1_random_walk_two.append(-1 if random() < 0.5 else 1)
278 for i in range(1, 1000):
279     movement = -1 if random() < 0.5 else 1

```

```

280 violin1_random_walk_two = [abs(x) for x in violin1_random_walk_two]
281 violin1_chord_two = [-5, 5, 10, 16, 23, 25, 26, 30, 38, 30, 26, 25, 23, 16,
282   10, 5, ]
283 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod15(
284   violin1_random_walk_two)]
285
286 seed(9)
287 violin2_random_walk_two = []
288 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
289 for i in range(1, 1000):
290   movement = -1 if random() < 0.5 else 1
291   value = violin2_random_walk_two[i-1] + movement
292   violin2_random_walk_two.append(value)
293 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
294 violin2_chord_two = [-5, 5, 10, 16, 23, 25, 26, 25, 23, 16, 10, 5, ]
295 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod11(
296   violin2_random_walk_two)]
297
298 seed(10)
299 viola_random_walk_two = []
300 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
301 for i in range(1, 1000):
302   movement = -1 if random() < 0.5 else 1
303   value = viola_random_walk_two[i-1] + movement
304   viola_random_walk_two.append(value)
305 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
306 viola_chord_two = [-5, 5, 10, 16, 23, 16, 10, 5, ]
307 viola_notes_two = [viola_chord_two[x] for x in reduceMod7(
308   viola_random_walk_two)]
309
310 seed(11)
311 cello_random_walk_two = []
312 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
313 for i in range(1, 1000):
314   movement = -1 if random() < 0.5 else 1
315   value = cello_random_walk_two[i-1] + movement
316   cello_random_walk_two.append(value)
317 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
318 cello_chord_two = [-24, -16, -5, 5, 10, 16, 23, 16, 10, 5, -5, -16]
319 cello_notes_two = [cello_chord_two[x] for x in reduceMod11(
320   cello_random_walk_two)]
321
322 seed(12)
323 bass_random_walk_two = []
324 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
325 for i in range(1, 1000):
326   movement = -1 if random() < 0.5 else 1
327   value = bass_random_walk_two[i-1] + movement
328   bass_random_walk_two.append(value)
329 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
330 bass_chord_two = [-27, -24, -16, -5, -16, -24, ]
331 bass_notes_two = [bass_chord_two[x] for x in reduceMod5(bass_random_walk_two)]
332
333 rmaker_one = abjadext.rmakers.NoteRhythmMaker()

```

```

329 rmaker_two = abjadext.rmakers.EvenDivisionRhythmMaker(
330     denominators=[16, 16, 8, 16, 4, 16, 8],
331     extra_counts_per_division=[0, 1, 0, 0, -1, 0, 1, -1],
332     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
333         left_classes=[abjad.Rest],
334         left_counts=[1],
335         right_classes=[abjad.Rest],
336         right_counts=[1],
337         outer_divisions_only=True,
338         ),
339     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
340         trivialize=True,
341         extract_trivial=True,
342         rewrite_rest_filled=True,
343         ),
344     )
345 )
346
347 rmaker_three = abjadext.rmakers.TaleaRhythmMaker(
348     talea=abjadext.rmakers.Talea(
349         counts=[1, 1, 1, 2, 1, 3, 1, 4, 5],
350         denominator=16,
351         ),
352     beamSpecifier=abjadext.rmakers.BeamSpecifier(
353         beam_divisions_together=True,
354         beam_rests=False,
355         ),
356     extra_counts_per_division=[0, 1, 0, -1],
357     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
358         left_classes=[abjad.Note, abjad.Rest],
359         left_counts=[1, 0, 1],
360         ),
361     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
362         trivialize=True,
363         extract_trivial=True,
364         rewrite_rest_filled=True,
365         ),
366     )
367
368 attachment_handler_one = AttachmentHandler(
369     starting_dynamic='p',
370     ending_dynamic='mp',
371     hairpin_indicator='--',
372     articulation='accent',
373     )
374
375 attachment_handler_two = AttachmentHandler(
376     starting_dynamic='fff',
377     ending_dynamic='mf',
378     hairpin_indicator='>',
379     articulation='tenuto',
380     )
381 attachment_handler_three = AttachmentHandler(

```

```

383     starting_dynamic='mp',
384     ending_dynamic='ff',
385     hairpin_indicator='<|',
386     articulation='',
387 )
388
389 ##### oboe#####
390 flutemusicmaker_one = MusicMaker(
391     rmaker=rmaker_one,
392     pitches=flute_scale,
393     continuous=True,
394     attachment_handler=attachment_handler_one,
395 )
396 flutemusicmaker_two = MusicMaker(
397     rmaker=rmaker_two,
398     pitches=flute_notes_two,
399     continuous=True,
400     attachment_handler=attachment_handler_two,
401 )
402 flutemusicmaker_three = MusicMaker(
403     rmaker=rmaker_three,
404     pitches=flute_notes_one,
405     continuous=True,
406     attachment_handler=attachment_handler_three,
407 )
408 ##### violin1#####
409 violin1musicmaker_one = MusicMaker(
410     rmaker=rmaker_one,
411     pitches=violin1_scale,
412     continuous=True,
413     attachment_handler=attachment_handler_one,
414 )
415 violin1musicmaker_two = MusicMaker(
416     rmaker=rmaker_two,
417     pitches=violin1_notes_two,
418     continuous=True,
419     attachment_handler=attachment_handler_two,
420 )
421 violin1musicmaker_three = MusicMaker(
422     rmaker=rmaker_three,
423     pitches=violin1_notes_one,
424     continuous=True,
425     attachment_handler=attachment_handler_three,
426 )
427 ##### trumpet#####
428 trumpetmusicmaker_one = MusicMaker(
429     rmaker=rmaker_one,
430     pitches=trumpet_scale,
431     continuous=True,
432     attachment_handler=attachment_handler_one,
433 )
434 trumpetmusicmaker_two = MusicMaker(
435     rmaker=rmaker_two,
436     pitches=trumpet_notes_two,

```

```

437     continuous=True,
438     attachment_handler=attachment_handler_two,
439 )
440 trumpetmusicmaker_three = MusicMaker(
441     rmaker=rmaker_three,
442     pitches=trumpet_notes_one,
443     continuous=True,
444     attachment_handler=attachment_handler_three,
445 )
446 #####clarinet#####
447 clarinetmusicmaker_one = MusicMaker(
448     rmaker=rmaker_one,
449     pitches=clarinet_scale,
450     continuous=True,
451     attachment_handler=attachment_handler_one,
452 )
453 clarinetmusicmaker_two = MusicMaker(
454     rmaker=rmaker_two,
455     pitches=clarinet_notes_two,
456     continuous=True,
457     attachment_handler=attachment_handler_two,
458 )
459 clarinetmusicmaker_three = MusicMaker(
460     rmaker=rmaker_three,
461     pitches=clarinet_notes_one,
462     continuous=True,
463     attachment_handler=attachment_handler_three,
464 )
465 #####violin2#####
466 violin2musicmaker_one = MusicMaker(
467     rmaker=rmaker_one,
468     pitches=violin2_scale,
469     continuous=True,
470     attachment_handler=attachment_handler_one,
471 )
472 violin2musicmaker_two = MusicMaker(
473     rmaker=rmaker_two,
474     pitches=violin2_notes_two,
475     continuous=True,
476     attachment_handler=attachment_handler_two,
477 )
478 violin2musicmaker_three = MusicMaker(
479     rmaker=rmaker_three,
480     pitches=violin2_notes_one,
481     continuous=True,
482     attachment_handler=attachment_handler_three,
483 )
484 #####viola#####
485 violamusicmaker_one = MusicMaker(
486     rmaker=rmaker_one,
487     pitches=viola_scale,
488     continuous=True,
489     attachment_handler=attachment_handler_one,
490 )

```

```
491 violamusicmaker_two = MusicMaker(  
492     rmaker=rmaker_two,  
493     pitches=viola_notes_two,  
494     continuous=True,  
495     attachment_handler=attachment_handler_two,  
496 )  
497 violamusicmaker_three = MusicMaker(  
498     rmaker=rmaker_three,  
499     pitches=viola_notes_one,  
500     continuous=True,  
501     attachment_handler=attachment_handler_three,  
502 )  
503 #####bassoon#####  
504 bassoonmusicmaker_one = MusicMaker(  
505     rmaker=rmaker_one,  
506     pitches=bassoon_scale,  
507     continuous=True,  
508     attachment_handler=attachment_handler_one,  
509 )  
510 bassoonmusicmaker_two = MusicMaker(  
511     rmaker=rmaker_two,  
512     pitches=bassoon_notes_two,  
513     continuous=True,  
514     attachment_handler=attachment_handler_two,  
515 )  
516 bassoonmusicmaker_three = MusicMaker(  
517     rmaker=rmaker_three,  
518     pitches=bassoon_notes_one,  
519     continuous=True,  
520     attachment_handler=attachment_handler_three,  
521 )  
522 #####trombone#####  
523 trombonemusicmaker_one = MusicMaker(  
524     rmaker=rmaker_one,  
525     pitches=trombone_scale,  
526     continuous=True,  
527     attachment_handler=attachment_handler_one,  
528 )  
529 trombonemusicmaker_two = MusicMaker(  
530     rmaker=rmaker_two,  
531     pitches=trombone_notes_two,  
532     continuous=True,  
533     attachment_handler=attachment_handler_two,  
534 )  
535 trombonemusicmaker_three = MusicMaker(  
536     rmaker=rmaker_three,  
537     pitches=trombone_notes_one,  
538     continuous=True,  
539     attachment_handler=attachment_handler_three,  
540 )  
541 #####cello#####  
542 cellomusicmaker_one = MusicMaker(  
543     rmaker=rmaker_one,  
544     pitches=cello_scale,
```

```

545     continuous=True,
546     attachment_handler=attachment_handler_one,
547 )
548 cellomusicmaker_two = MusicMaker(
549     rmaker=rmaker_two,
550     pitches=cello_notes_two,
551     continuous=True,
552     attachment_handler=attachment_handler_two,
553 )
554 cellomusicmaker_three = MusicMaker(
555     rmaker=rmaker_three,
556     pitches=cello_notes_one,
557     continuous=True,
558     attachment_handler=attachment_handler_three,
559 )
560 #####horn#####
561 hornmusicmaker_one = MusicMaker(
562     rmaker=rmaker_one,
563     pitches=horn_scale,
564     continuous=True,
565     attachment_handler=attachment_handler_one,
566 )
567 hornmusicmaker_two = MusicMaker(
568     rmaker=rmaker_two,
569     pitches=horn_notes_two,
570     continuous=True,
571     attachment_handler=attachment_handler_two,
572 )
573 hornmusicmaker_three = MusicMaker(
574     rmaker=rmaker_three,
575     pitches=horn_notes_one,
576     continuous=True,
577     attachment_handler=attachment_handler_three,
578 )
579 #####tuba#####
580 tubamusicmaker_one = MusicMaker(
581     rmaker=rmaker_one,
582     pitches=tuba_scale,
583     continuous=True,
584     attachment_handler=attachment_handler_one,
585 )
586 tubamusicmaker_two = MusicMaker(
587     rmaker=rmaker_two,
588     pitches=tuba_notes_two,
589     continuous=True,
590     attachment_handler=attachment_handler_two,
591 )
592 tubamusicmaker_three = MusicMaker(
593     rmaker=rmaker_three,
594     pitches=tuba_notes_one,
595     continuous=True,
596     attachment_handler=attachment_handler_three,
597 )
598 #####bass#####

```

```

599 bassmusicmaker_one = MusicMaker(
600     rmaker=rmaker_one,
601     pitches=bass_scale,
602     continuous=True,
603     attachment_handler=attachment_handler_one,
604 )
605 bassmusicmaker_two = MusicMaker(
606     rmaker=rmaker_two,
607     pitches=bass_notes_two,
608     continuous=True,
609     attachment_handler=attachment_handler_two,
610 )
611 bassmusicmaker_three = MusicMaker(
612     rmaker=rmaker_three,
613     pitches=bass_notes_one,
614     continuous=True,
615     attachment_handler=attachment_handler_three,
616 )
617
618 silence_maker = abjadext.rmakers.NoteRhythmMaker(
619     division_masks=[
620         abjadext.rmakers.SilenceMask(
621             pattern=abjad.index([0], 1),
622             ),
623         ],
624     )
625
626 class MusicSpecifier:
627
628     def __init__(self, music_maker, voice_name):
629         self.music_maker = music_maker
630         self.voice_name = voice_name
631
632     print('Collecting timespans and rmakers ...')
633     ##group one###
634     voice_1_timespan_list = abjad.TimespanList([
635         abjad.AnnotatedTimespan(
636             start_offset=start_offset,
637             stop_offset=stop_offset,
638             annotation=MusicSpecifier(
639                 music_maker=music_maker,
640                 voice_name='Voice 1',
641             ),
642         )
643         for start_offset, stop_offset, music_maker in [
644             [(9, 4), (10, 4), flutemusicmaker_one],
645             [(15, 4), (18, 4), flutemusicmaker_two],
646             [(22, 4), (25, 4), flutemusicmaker_three],
647             [(27, 4), (30, 4), flutemusicmaker_one],
648             [(30, 4), (32, 4), flutemusicmaker_one],
649             [(35, 4), (39, 4), flutemusicmaker_two],
650             [(42, 4), (43, 4), flutemusicmaker_three],
651             [(43, 4), (44, 4), flutemusicmaker_three],
652             [(45, 4), (46, 4), flutemusicmaker_one],

```

```

653     [(46, 4), (50, 4), flutemusicmaker_one],
654     [(54, 4), (57, 4), flutemusicmaker_two],
655     [(59, 4), (60, 4), flutemusicmaker_three],
656     [(65, 4), (67, 4), flutemusicmaker_one],
657     [(67, 4), (69, 4), flutemusicmaker_one],
658     [(70, 4), (72, 4), flutemusicmaker_two],
659     [(72, 4), (75, 4), flutemusicmaker_two],
660     [(76, 4), (78, 4), flutemusicmaker_three],
661     [(81, 4), (82, 4), flutemusicmaker_one],
662     [(82, 4), (85, 4), flutemusicmaker_one],
663     [(90, 4), (91, 4), flutemusicmaker_two],
664     [(93, 4), (94, 4), flutemusicmaker_three],
665     [(94, 4), (96, 4), flutemusicmaker_three],
666     [(100, 4), (104, 4), flutemusicmaker_one],
667     [(104, 4), (105, 4), flutemusicmaker_one],
668     [(106, 4), (107, 4), flutemusicmaker_two],
669     [(107, 4), (108, 4), flutemusicmaker_two],
670     [(111, 4), (114, 4), flutemusicmaker_one],
671     [(114, 4), (115, 4), flutemusicmaker_one],
672     [(116, 4), (119, 4), flutemusicmaker_one],
673     [(119, 4), (120, 4), flutemusicmaker_one],
674     [(121, 4), (123, 4), flutemusicmaker_one],
675     [(123, 4), (125, 4), flutemusicmaker_one],
676     [(126, 4), (131, 4), flutemusicmaker_two],
677     [(131, 4), (133, 4), flutemusicmaker_two],
678     [(136, 4), (141, 4), flutemusicmaker_two],
679     [(148, 4), (150, 4), flutemusicmaker_two],
680     [(150, 4), (153, 4), flutemusicmaker_three],
681     [(155, 4), (159, 4), flutemusicmaker_three],
682     [(162, 4), (164, 4), flutemusicmaker_three],
683     [(168, 4), (171, 4), flutemusicmaker_three],
684     [(173, 4), (175, 4), flutemusicmaker_three],
685     [(175, 4), (177, 4), flutemusicmaker_three],
686     [(180, 4), (182, 4), flutemusicmaker_three],
687     [(186, 4), (190, 4), flutemusicmaker_three],
688     [(190, 4), (191, 4), silence_maker],
689   ]
690 ])
691
692 voice_5_timestspan_list = abjad.TimespanList([
693     abjad.AnnotatedTimespan(
694         start_offset=start_offset,
695         stop_offset=stop_offset,
696         annotation=MusicSpecifier(
697             music_maker=music_maker,
698             voice_name='Voice 5',
699         ),
700     )
701     for start_offset, stop_offset, music_maker in [
702         [(9, 4), (10, 4), trumpetmusicmaker_one],
703         [(14, 4), (18, 4), trumpetmusicmaker_two],
704         [(23, 4), (25, 4), trumpetmusicmaker_three],
705         [(27, 4), (30, 4), trumpetmusicmaker_one],
706         [(30, 4), (32, 4), trumpetmusicmaker_one],

```

```

707     [(35, 4), (39, 4), trumpetmusicmaker_two],
708     [(42, 4), (43, 4), trumpetmusicmaker_three],
709     [(43, 4), (44, 4), trumpetmusicmaker_three],
710     [(45, 4), (46, 4), trumpetmusicmaker_one],
711     [(46, 4), (50, 4), trumpetmusicmaker_one],
712     [(54, 4), (57, 4), trumpetmusicmaker_two],
713     [(59, 4), (60, 4), trumpetmusicmaker_three],
714     [(65, 4), (67, 4), trumpetmusicmaker_one],
715     [(67, 4), (69, 4), trumpetmusicmaker_one],
716     [(70, 4), (72, 4), trumpetmusicmaker_two],
717     [(72, 4), (75, 4), trumpetmusicmaker_two],
718     [(76, 4), (78, 4), trumpetmusicmaker_three],
719     [(81, 4), (82, 4), trumpetmusicmaker_one],
720     [(82, 4), (85, 4), trumpetmusicmaker_one],
721     [(90, 4), (91, 4), trumpetmusicmaker_two],
722     [(93, 4), (94, 4), trumpetmusicmaker_three],
723     [(94, 4), (96, 4), trumpetmusicmaker_three],
724     [(100, 4), (104, 4), trumpetmusicmaker_one],
725     [(104, 4), (105, 4), trumpetmusicmaker_one],
726     [(106, 4), (107, 4), trumpetmusicmaker_two],
727     [(107, 4), (108, 4), trumpetmusicmaker_two],
728     [(111, 4), (114, 4), trumpetmusicmaker_one],
729     [(114, 4), (115, 4), trumpetmusicmaker_one],
730     [(116, 4), (119, 4), trumpetmusicmaker_one],
731     [(119, 4), (120, 4), trumpetmusicmaker_one],
732     [(121, 4), (123, 4), trumpetmusicmaker_one],
733     [(123, 4), (125, 4), trumpetmusicmaker_one],
734     [(126, 4), (131, 4), trumpetmusicmaker_two],
735     [(131, 4), (133, 4), trumpetmusicmaker_two],
736     [(136, 4), (141, 4), trumpetmusicmaker_two],
737     [(148, 4), (150, 4), trumpetmusicmaker_two],
738     [(150, 4), (154, 4), trumpetmusicmaker_three],
739     [(157, 4), (159, 4), trumpetmusicmaker_three],
740     [(163, 4), (164, 4), trumpetmusicmaker_three],
741     [(164, 4), (166, 4), trumpetmusicmaker_three],
742     [(168, 4), (172, 4), trumpetmusicmaker_three],
743     [(175, 4), (177, 4), trumpetmusicmaker_three],
744     [(181, 4), (183, 4), trumpetmusicmaker_three],
745     [(183, 4), (184, 4), trumpetmusicmaker_three],
746     [(186, 4), (190, 4), trumpetmusicmaker_three],
747 ]
748 ])
749
750 voice_8_timespan_list = abjad.TimespanList([
751     abjad.AnnotatedTimespan(
752         start_offset=start_offset,
753         stop_offset=stop_offset,
754         annotation=MusicSpecifier(
755             music_maker=music_maker,
756             voice_name='Voice 8',
757         ),
758     )
759     for start_offset, stop_offset, music_maker in [
760         [(9, 4), (10, 4), violin1musicmaker_one],

```

```

761     [(14, 4), (18, 4), violin1musicmaker_two],
762     [(22, 4), (25, 4), violin1musicmaker_three],
763     [(27, 4), (30, 4), violin1musicmaker_one],
764     [(35, 4), (39, 4), violin1musicmaker_two],
765     [(42, 4), (43, 4), violin1musicmaker_three],
766     [(43, 4), (44, 4), violin1musicmaker_three],
767     [(45, 4), (46, 4), violin1musicmaker_one],
768     [(46, 4), (50, 4), violin1musicmaker_one],
769     [(54, 4), (57, 4), violin1musicmaker_two],
770     [(59, 4), (60, 4), violin1musicmaker_three],
771     [(65, 4), (67, 4), violin1musicmaker_one],
772     [(67, 4), (69, 4), violin1musicmaker_one],
773     [(70, 4), (72, 4), violin1musicmaker_two],
774     [(72, 4), (75, 4), violin1musicmaker_two],
775     [(76, 4), (78, 4), violin1musicmaker_three],
776     [(81, 4), (82, 4), violin1musicmaker_one],
777     [(82, 4), (85, 4), violin1musicmaker_one],
778     [(90, 4), (91, 4), violin1musicmaker_two],
779     [(93, 4), (94, 4), violin1musicmaker_three],
780     [(94, 4), (96, 4), violin1musicmaker_three],
781     [(100, 4), (104, 4), violin1musicmaker_one],
782     [(104, 4), (105, 4), violin1musicmaker_one],
783     [(106, 4), (107, 4), violin1musicmaker_two],
784     [(107, 4), (108, 4), violin1musicmaker_two],
785     [(111, 4), (114, 4), violin1musicmaker_one],
786     [(114, 4), (115, 4), violin1musicmaker_one],
787     [(116, 4), (119, 4), violin1musicmaker_one],
788     [(119, 4), (120, 4), violin1musicmaker_one],
789     [(121, 4), (123, 4), violin1musicmaker_one],
790     [(123, 4), (125, 4), violin1musicmaker_one],
791     [(126, 4), (131, 4), violin1musicmaker_two],
792     [(131, 4), (133, 4), violin1musicmaker_two],
793     [(136, 4), (141, 4), violin1musicmaker_two],
794     [(148, 4), (150, 4), violin1musicmaker_two],
795     [(150, 4), (152, 4), violin1musicmaker_three],
796     [(156, 4), (159, 4), violin1musicmaker_three],
797     [(161, 4), (164, 4), violin1musicmaker_three],
798     [(164, 4), (165, 4), violin1musicmaker_three],
799     [(168, 4), (170, 4), violin1musicmaker_three],
800     [(174, 4), (175, 4), violin1musicmaker_three],
801     [(175, 4), (177, 4), violin1musicmaker_three],
802     [(179, 4), (183, 4), violin1musicmaker_three],
803     [(186, 4), (190, 4), violin1musicmaker_three],
804     ]
805   ])
806
807   ###group two###
808   voice_2_timespan_list = abjad.TimespanList([
809     abjad.AnnotatedTimespan(
810       start_offset=start_offset,
811       stop_offset=stop_offset,
812       annotation=MusicSpecifier(
813         music_maker=music_maker,
814         voice_name='Voice 2',

```

```

815     ),
816
817     for start_offset, stop_offset, music_maker in [
818         [(2, 4), (5, 4), clarinetmusicmaker_one],
819         [(10, 4), (11, 4), clarinetmusicmaker_two],
820         [(11, 4), (13, 4), clarinetmusicmaker_two],
821         [(16, 4), (18, 4), clarinetmusicmaker_three],
822         [(21, 4), (22, 4), clarinetmusicmaker_one],
823         [(22, 4), (25, 4), clarinetmusicmaker_one],
824         [(35, 4), (40, 4), clarinetmusicmaker_one],
825         [(44, 4), (46, 4), clarinetmusicmaker_two],
826         [(46, 4), (47, 4), clarinetmusicmaker_two],
827         [(49, 4), (50, 4), clarinetmusicmaker_three],
828         [(55, 4), (59, 4), clarinetmusicmaker_one],
829         [(62, 4), (64, 4), clarinetmusicmaker_two],
830         [(65, 4), (67, 4), clarinetmusicmaker_three],
831         [(67, 4), (70, 4), clarinetmusicmaker_three],
832         [(70, 4), (71, 4), clarinetmusicmaker_three],
833         [(73, 4), (75, 4), clarinetmusicmaker_two],
834         [(75, 4), (76, 4), clarinetmusicmaker_two],
835         [(80, 4), (82, 4), clarinetmusicmaker_one],
836         [(82, 4), (85, 4), clarinetmusicmaker_one],
837         [(86, 4), (88, 4), clarinetmusicmaker_two],
838         [(91, 4), (94, 4), clarinetmusicmaker_three],
839         [(94, 4), (95, 4), clarinetmusicmaker_three],
840         [(100, 4), (101, 4), clarinetmusicmaker_two],
841         [(103, 4), (104, 4), clarinetmusicmaker_one],
842         [(104, 4), (106, 4), clarinetmusicmaker_one],
843         [(110, 4), (114, 4), clarinetmusicmaker_one],
844         [(115, 4), (119, 4), clarinetmusicmaker_one],
845         [(120, 4), (123, 4), clarinetmusicmaker_one],
846         [(123, 4), (124, 4), clarinetmusicmaker_one],
847         [(125, 4), (126, 4), clarinetmusicmaker_two],
848         [(129, 4), (131, 4), clarinetmusicmaker_two],
849         [(131, 4), (134, 4), clarinetmusicmaker_two],
850         [(141, 4), (144, 4), clarinetmusicmaker_two],
851         [(149, 4), (150, 4), clarinetmusicmaker_two],
852         [(155, 4), (159, 4), clarinetmusicmaker_three],
853         [(162, 4), (164, 4), clarinetmusicmaker_three],
854         [(165, 4), (168, 4), clarinetmusicmaker_three],
855         [(168, 4), (170, 4), clarinetmusicmaker_three],
856         [(174, 4), (175, 4), clarinetmusicmaker_three],
857         [(175, 4), (177, 4), clarinetmusicmaker_three],
858         [(179, 4), (180, 4), clarinetmusicmaker_three],
859         [(185, 4), (186, 4), clarinetmusicmaker_three],
860         [(186, 4), (190, 4), clarinetmusicmaker_three],
861     ]
862   ])
863
864 voice_9_timespan_list = abjad.TimespanList([
865     abjad.AnnotatedTimespan(
866         start_offset=start_offset,
867         stop_offset=stop_offset,
868         annotation=MusicSpecifier(

```

```

869     music_maker=music_maker,
870     voice_name='Voice 9',
871 ),
872 )
873 for start_offset, stop_offset, music_maker in [
874     [(2, 4), (5, 4), violin2musicmaker_one],
875     [(9, 4), (11, 4), violin2musicmaker_two],
876     [(11, 4), (13, 4), violin2musicmaker_two],
877     [(16, 4), (18, 4), violin2musicmaker_three],
878     [(21, 4), (22, 4), violin2musicmaker_one],
879     [(22, 4), (23, 4), violin2musicmaker_one],
880     [(35, 4), (40, 4), violin2musicmaker_one],
881     [(44, 4), (46, 4), violin2musicmaker_two],
882     [(46, 4), (47, 4), violin2musicmaker_two],
883     [(49, 4), (50, 4), violin2musicmaker_three],
884     [(55, 4), (59, 4), violin2musicmaker_one],
885     [(62, 4), (64, 4), violin2musicmaker_two],
886     [(65, 4), (67, 4), violin2musicmaker_three],
887     [(67, 4), (70, 4), violin2musicmaker_three],
888     [(70, 4), (71, 4), violin2musicmaker_three],
889     [(73, 4), (75, 4), violin2musicmaker_two],
890     [(75, 4), (76, 4), violin2musicmaker_two],
891     [(80, 4), (82, 4), violin2musicmaker_one],
892     [(82, 4), (85, 4), violin2musicmaker_one],
893     [(86, 4), (88, 4), violin2musicmaker_two],
894     [(91, 4), (94, 4), violin2musicmaker_three],
895     [(94, 4), (95, 4), violin2musicmaker_three],
896     [(100, 4), (101, 4), violin2musicmaker_two],
897     [(103, 4), (104, 4), violin2musicmaker_one],
898     [(104, 4), (106, 4), violin2musicmaker_one],
899     [(110, 4), (114, 4), violin2musicmaker_one],
900     [(115, 4), (119, 4), violin2musicmaker_one],
901     [(120, 4), (123, 4), violin2musicmaker_one],
902     [(123, 4), (124, 4), violin2musicmaker_one],
903     [(125, 4), (126, 4), violin2musicmaker_two],
904     [(129, 4), (131, 4), violin2musicmaker_two],
905     [(131, 4), (134, 4), violin2musicmaker_two],
906     [(141, 4), (144, 4), violin2musicmaker_two],
907     [(149, 4), (150, 4), violin2musicmaker_two],
908     [(154, 4), (157, 4), violin2musicmaker_three],
909     [(159, 4), (160, 4), violin2musicmaker_three],
910     [(165, 4), (168, 4), violin2musicmaker_three],
911     [(168, 4), (169, 4), violin2musicmaker_three],
912     [(172, 4), (174, 4), violin2musicmaker_three],
913     [(175, 4), (179, 4), violin2musicmaker_three],
914     [(179, 4), (180, 4), violin2musicmaker_three],
915     [(184, 4), (186, 4), violin2musicmaker_three],
916     [(186, 4), (190, 4), violin2musicmaker_three],
917 ]
918 ])
919
920 voice_10_timespan_list = abjad.TimespanList([
921     abjad.AnnotatedTimespan(
922         start_offset=start_offset,

```

```

923     stop_offset=stop_offset,
924     annotation=MusicSpecifier(
925         music_maker=music_maker,
926         voice_name='Voice 10',
927     ),
928 )
929 for start_offset, stop_offset, music_maker in [
930     [(2, 4), (5, 4), violamusicmaker_one],
931     [(9, 4), (11, 4), violamusicmaker_two],
932     [(11, 4), (13, 4), violamusicmaker_two],
933     [(17, 4), (18, 4), violamusicmaker_three],
934     [(21, 4), (22, 4), violamusicmaker_one],
935     [(22, 4), (25, 4), violamusicmaker_one],
936     [(29, 4), (30, 4), violamusicmaker_two],
937     [(30, 4), (32, 4), violamusicmaker_two],
938     [(35, 4), (40, 4), violamusicmaker_one],
939     [(44, 4), (46, 4), violamusicmaker_two],
940     [(46, 4), (47, 4), violamusicmaker_two],
941     [(49, 4), (50, 4), violamusicmaker_three],
942     [(55, 4), (59, 4), violamusicmaker_one],
943     [(62, 4), (64, 4), violamusicmaker_two],
944     [(65, 4), (67, 4), violamusicmaker_three],
945     [(67, 4), (70, 4), violamusicmaker_three],
946     [(70, 4), (71, 4), violamusicmaker_three],
947     [(73, 4), (75, 4), violamusicmaker_two],
948     [(75, 4), (76, 4), violamusicmaker_two],
949     [(80, 4), (82, 4), violamusicmaker_one],
950     [(82, 4), (85, 4), violamusicmaker_one],
951     [(86, 4), (88, 4), violamusicmaker_two],
952     [(91, 4), (94, 4), violamusicmaker_three],
953     [(94, 4), (95, 4), violamusicmaker_three],
954     [(100, 4), (101, 4), violamusicmaker_two],
955     [(103, 4), (104, 4), violamusicmaker_one],
956     [(104, 4), (106, 4), violamusicmaker_one],
957     [(110, 4), (114, 4), violamusicmaker_one],
958     [(115, 4), (119, 4), violamusicmaker_one],
959     [(120, 4), (123, 4), violamusicmaker_one],
960     [(123, 4), (124, 4), violamusicmaker_one],
961     [(125, 4), (126, 4), violamusicmaker_two],
962     [(129, 4), (131, 4), violamusicmaker_two],
963     [(131, 4), (134, 4), violamusicmaker_two],
964     [(141, 4), (144, 4), violamusicmaker_two],
965     [(149, 4), (150, 4), violamusicmaker_two],
966     [(153, 4), (154, 4), violamusicmaker_three],
967     [(154, 4), (155, 4), violamusicmaker_three],
968     [(156, 4), (159, 4), violamusicmaker_three],
969     [(159, 4), (161, 4), violamusicmaker_three],
970     [(165, 4), (168, 4), violamusicmaker_three],
971     [(170, 4), (171, 4), violamusicmaker_three],
972     [(176, 4), (179, 4), violamusicmaker_three],
973     [(179, 4), (180, 4), violamusicmaker_three],
974     [(183, 4), (185, 4), violamusicmaker_three],
975     [(186, 4), (190, 4), violamusicmaker_three],
976 ]

```

```

977 ])
978
979 #####group three#####
980 voice_3_timespan_list = abjad.TimespanList([
981     abjad.AnnotatedTimespan(
982         start_offset=start_offset,
983         stop_offset=stop_offset,
984         annotation=MusicSpecifier(
985             music_maker=music_maker,
986             voice_name='Voice 3',
987         ),
988     )
989     for start_offset, stop_offset, music_maker in [
990         [(7, 4), (11, 4), bassoonmusicmaker_one],
991         [(15, 4), (16, 4), bassoonmusicmaker_two],
992         [(19, 4), (22, 4), bassoonmusicmaker_three],
993         [(22, 4), (23, 4), bassoonmusicmaker_three],
994         [(27, 4), (30, 4), bassoonmusicmaker_one],
995         [(32, 4), (35, 4), bassoonmusicmaker_two],
996         [(35, 4), (36, 4), bassoonmusicmaker_three],
997         [(37, 4), (40, 4), bassoonmusicmaker_two],
998         [(40, 4), (42, 4), bassoonmusicmaker_two],
999         [(46, 4), (49, 4), bassoonmusicmaker_one],
1000         [(51, 4), (52, 4), bassoonmusicmaker_three],
1001         [(57, 4), (59, 4), bassoonmusicmaker_two],
1002         [(59, 4), (61, 4), bassoonmusicmaker_two],
1003         [(64, 4), (66, 4), bassoonmusicmaker_one],
1004         [(67, 4), (70, 4), bassoonmusicmaker_three],
1005         [(70, 4), (72, 4), bassoonmusicmaker_one],
1006         [(72, 4), (73, 4), bassoonmusicmaker_one],
1007         [(77, 4), (79, 4), bassoonmusicmaker_two],
1008         [(79, 4), (82, 4), bassoonmusicmaker_two],
1009         [(83, 4), (85, 4), bassoonmusicmaker_three],
1010         [(88, 4), (89, 4), bassoonmusicmaker_two],
1011         [(89, 4), (92, 4), bassoonmusicmaker_two],
1012         [(97, 4), (98, 4), bassoonmusicmaker_one],
1013         [(100, 4), (103, 4), bassoonmusicmaker_two],
1014         [(107, 4), (110, 4), bassoonmusicmaker_three],
1015         [(110, 4), (112, 4), bassoonmusicmaker_one],
1016         [(113, 4), (114, 4), bassoonmusicmaker_one],
1017         [(114, 4), (117, 4), bassoonmusicmaker_one],
1018         [(118, 4), (119, 4), bassoonmusicmaker_one],
1019         [(119, 4), (122, 4), bassoonmusicmaker_one],
1020         [(123, 4), (125, 4), bassoonmusicmaker_one],
1021         [(126, 4), (131, 4), bassoonmusicmaker_two],
1022         [(138, 4), (141, 4), bassoonmusicmaker_two],
1023         [(146, 4), (150, 4), bassoonmusicmaker_two],
1024         [(150, 4), (154, 4), bassoonmusicmaker_three],
1025         [(154, 4), (155, 4), bassoonmusicmaker_three],
1026         [(159, 4), (162, 4), bassoonmusicmaker_three],
1027         [(164, 4), (165, 4), bassoonmusicmaker_three],
1028         [(170, 4), (172, 4), bassoonmusicmaker_three],
1029         [(172, 4), (174, 4), bassoonmusicmaker_three],
1030         [(177, 4), (179, 4), bassoonmusicmaker_three],

```

```

1031     [(180, 4), (183, 4), bassoonmusicmaker_three],
1032     [(183, 4), (185, 4), bassoonmusicmaker_three],
1033     [(186, 4), (190, 4), bassoonmusicmaker_three],
1034   ]
1035 ])
1036
1037 voice_6_timespan_list = abjad.TimespanList([
1038     abjad.AnnotatedTimespan(
1039         start_offset=start_offset,
1040         stop_offset=stop_offset,
1041         annotation=MusicSpecifier(
1042             music_maker=music_maker,
1043             voice_name='Voice 6',
1044         ),
1045     )
1046     for start_offset, stop_offset, music_maker in [
1047         [(7, 4), (11, 4), trombonemusicmaker_one],
1048         [(14, 4), (16, 4), trombonemusicmaker_two],
1049         [(19, 4), (22, 4), trombonemusicmaker_three],
1050         [(22, 4), (23, 4), trombonemusicmaker_three],
1051         [(27, 4), (29, 4), trombonemusicmaker_one],
1052         [(35, 4), (36, 4), trombonemusicmaker_three],
1053         [(37, 4), (40, 4), trombonemusicmaker_two],
1054         [(40, 4), (42, 4), trombonemusicmaker_two],
1055         [(46, 4), (49, 4), trombonemusicmaker_one],
1056         [(51, 4), (52, 4), trombonemusicmaker_three],
1057         [(57, 4), (59, 4), trombonemusicmaker_two],
1058         [(59, 4), (61, 4), trombonemusicmaker_two],
1059         [(64, 4), (66, 4), trombonemusicmaker_one],
1060         [(67, 4), (70, 4), trombonemusicmaker_three],
1061         [(70, 4), (72, 4), trombonemusicmaker_one],
1062         [(72, 4), (73, 4), trombonemusicmaker_one],
1063         [(77, 4), (79, 4), trombonemusicmaker_two],
1064         [(79, 4), (82, 4), trombonemusicmaker_two],
1065         [(83, 4), (85, 4), trombonemusicmaker_three],
1066         [(88, 4), (89, 4), trombonemusicmaker_two],
1067         [(89, 4), (92, 4), trombonemusicmaker_two],
1068         [(97, 4), (98, 4), trombonemusicmaker_one],
1069         [(100, 4), (103, 4), trombonemusicmaker_two],
1070         [(107, 4), (110, 4), trombonemusicmaker_three],
1071         [(110, 4), (112, 4), trombonemusicmaker_one],
1072         [(113, 4), (114, 4), trombonemusicmaker_one],
1073         [(114, 4), (117, 4), trombonemusicmaker_one],
1074         [(118, 4), (119, 4), trombonemusicmaker_one],
1075         [(119, 4), (122, 4), trombonemusicmaker_one],
1076         [(123, 4), (125, 4), trombonemusicmaker_one],
1077         [(126, 4), (131, 4), trombonemusicmaker_two],
1078         [(138, 4), (141, 4), trombonemusicmaker_two],
1079         [(146, 4), (150, 4), trombonemusicmaker_two],
1080         [(150, 4), (154, 4), trombonemusicmaker_three],
1081         [(157, 4), (159, 4), trombonemusicmaker_three],
1082         [(160, 4), (164, 4), trombonemusicmaker_three],
1083         [(164, 4), (165, 4), trombonemusicmaker_three],
1084         [(169, 4), (172, 4), trombonemusicmaker_three],

```

```

1085     [(174, 4), (175, 4), trombonemusicmaker_three],
1086     [(180, 4), (183, 4), trombonemusicmaker_three],
1087     [(183, 4), (184, 4), trombonemusicmaker_three],
1088     [(186, 4), (190, 4), trombonemusicmaker_three],
1089   ]
1090 ])
1091
1092 voice_11_timespan_list = abjad.TimespanList([
1093     abjad.AnnotatedTimespan(
1094         start_offset=start_offset,
1095         stop_offset=stop_offset,
1096         annotation=MusicSpecifier(
1097             music_maker=music_maker,
1098             voice_name='Voice 11',
1099         ),
1100     )
1101     for start_offset, stop_offset, music_maker in [
1102         [(7, 4), (11, 4), cellomusicmaker_one],
1103         [(14, 4), (16, 4), cellomusicmaker_two],
1104         [(21, 4), (22, 4), cellomusicmaker_three],
1105         [(22, 4), (23, 4), cellomusicmaker_three],
1106         [(27, 4), (30, 4), cellomusicmaker_one],
1107         [(35, 4), (36, 4), cellomusicmaker_three],
1108         [(37, 4), (40, 4), cellomusicmaker_two],
1109         [(40, 4), (42, 4), cellomusicmaker_two],
1110         [(46, 4), (49, 4), cellomusicmaker_one],
1111         [(51, 4), (52, 4), cellomusicmaker_three],
1112         [(57, 4), (59, 4), cellomusicmaker_two],
1113         [(59, 4), (61, 4), cellomusicmaker_two],
1114         [(64, 4), (66, 4), cellomusicmaker_one],
1115         [(67, 4), (70, 4), cellomusicmaker_three],
1116         [(70, 4), (72, 4), cellomusicmaker_one],
1117         [(72, 4), (73, 4), cellomusicmaker_one],
1118         [(77, 4), (79, 4), cellomusicmaker_two],
1119         [(79, 4), (82, 4), cellomusicmaker_two],
1120         [(83, 4), (85, 4), cellomusicmaker_three],
1121         [(88, 4), (89, 4), cellomusicmaker_two],
1122         [(89, 4), (92, 4), cellomusicmaker_two],
1123         [(97, 4), (98, 4), cellomusicmaker_one],
1124         [(100, 4), (103, 4), cellomusicmaker_two],
1125         [(107, 4), (110, 4), cellomusicmaker_three],
1126         [(110, 4), (112, 4), cellomusicmaker_one],
1127         [(113, 4), (114, 4), cellomusicmaker_one],
1128         [(114, 4), (117, 4), cellomusicmaker_one],
1129         [(118, 4), (119, 4), cellomusicmaker_one],
1130         [(119, 4), (122, 4), cellomusicmaker_one],
1131         [(123, 4), (125, 4), cellomusicmaker_one],
1132         [(126, 4), (131, 4), cellomusicmaker_two],
1133         [(138, 4), (141, 4), cellomusicmaker_two],
1134         [(146, 4), (150, 4), cellomusicmaker_two],
1135         [(150, 4), (153, 4), cellomusicmaker_three],
1136         [(155, 4), (156, 4), cellomusicmaker_three],
1137         [(161, 4), (164, 4), cellomusicmaker_three],
1138         [(164, 4), (165, 4), cellomusicmaker_three],

```

```

1139     [(168, 4), (170, 4), cellomusicmaker_three],
1140     [(171, 4), (172, 4), cellomusicmaker_three],
1141     [(172, 4), (175, 4), cellomusicmaker_three],
1142     [(175, 4), (176, 4), cellomusicmaker_three],
1143     [(180, 4), (183, 4), cellomusicmaker_three],
1144     [(185, 4), (186, 4), cellomusicmaker_three],
1145     [(186, 4), (190, 4), cellomusicmaker_three],
1146   ]
1147 ])
1148
1149 ###group four###
1150 voice_4_timestrap_list = abjad.TimespanList([
1151     abjad.AnnotatedTimespan(
1152         start_offset=start_offset,
1153         stop_offset=stop_offset,
1154         annotation=MusicSpecifier(
1155             music_maker=music_maker,
1156             voice_name='Voice 4',
1157         ),
1158     )
1159     for start_offset, stop_offset, music_maker in [
1160         [(0, 4), (5, 4), hornmusicmaker_one],
1161         [(8, 4), (10, 4), hornmusicmaker_two],
1162         [(14, 4), (18, 4), hornmusicmaker_three],
1163         [(21, 4), (22, 4), hornmusicmaker_one],
1164         [(22, 4), (23, 4), hornmusicmaker_one],
1165         [(38, 4), (40, 4), hornmusicmaker_two],
1166         [(41, 4), (43, 4), hornmusicmaker_one],
1167         [(43, 4), (46, 4), hornmusicmaker_one],
1168         [(50, 4), (53, 4), hornmusicmaker_three],
1169         [(55, 4), (56, 4), hornmusicmaker_two],
1170         [(61, 4), (64, 4), hornmusicmaker_one],
1171         [(64, 4), (65, 4), hornmusicmaker_one],
1172         [(68, 4), (70, 4), hornmusicmaker_three],
1173         [(70, 4), (72, 4), hornmusicmaker_two],
1174         [(72, 4), (74, 4), hornmusicmaker_two],
1175         [(79, 4), (80, 4), hornmusicmaker_three],
1176         [(82, 4), (85, 4), hornmusicmaker_two],
1177         [(89, 4), (94, 4), hornmusicmaker_one],
1178         [(95, 4), (97, 4), hornmusicmaker_two],
1179         [(100, 4), (104, 4), hornmusicmaker_three],
1180         [(109, 4), (110, 4), hornmusicmaker_two],
1181         [(110, 4), (111, 4), hornmusicmaker_one],
1182         [(112, 4), (114, 4), hornmusicmaker_one],
1183         [(114, 4), (116, 4), hornmusicmaker_one],
1184         [(117, 4), (119, 4), hornmusicmaker_one],
1185         [(119, 4), (121, 4), hornmusicmaker_one],
1186         [(122, 4), (123, 4), hornmusicmaker_one],
1187         [(123, 4), (125, 4), hornmusicmaker_one],
1188         [(133, 4), (136, 4), hornmusicmaker_two],
1189         [(142, 4), (146, 4), hornmusicmaker_two],
1190         [(146, 4), (150, 4), hornmusicmaker_two],
1191         [(153, 4), (154, 4), hornmusicmaker_three],
1192         [(154, 4), (155, 4), hornmusicmaker_three],

```

```

1193     [(159, 4), (162, 4), hornmusicmaker_three],
1194     [(164, 4), (168, 4), hornmusicmaker_three],
1195     [(171, 4), (172, 4), hornmusicmaker_three],
1196     [(172, 4), (173, 4), hornmusicmaker_three],
1197     [(177, 4), (179, 4), hornmusicmaker_three],
1198     [(179, 4), (180, 4), hornmusicmaker_three],
1199     [(182, 4), (183, 4), hornmusicmaker_three],
1200     [(183, 4), (186, 4), hornmusicmaker_three],
1201     [(186, 4), (190, 4), hornmusicmaker_three],
1202 ]
1203 ])
1204
1205 voice_7_timespan_list = abjad.TimespanList([
1206     abjad.AnnotatedTimespan(
1207         start_offset=start_offset,
1208         stop_offset=stop_offset,
1209         annotation=MusicSpecifier(
1210             music_maker=music_maker,
1211             voice_name='Voice 7',
1212         ),
1213     )
1214     for start_offset, stop_offset, music_maker in [
1215         [(0, 4), (5, 4), tubamusicmaker_one],
1216         [(8, 4), (10, 4), tubamusicmaker_two],
1217         [(14, 4), (18, 4), tubamusicmaker_three],
1218         [(21, 4), (22, 4), tubamusicmaker_one],
1219         [(22, 4), (23, 4), tubamusicmaker_one],
1220         [(26, 4), (30, 4), tubamusicmaker_two],
1221         [(38, 4), (40, 4), tubamusicmaker_two],
1222         [(41, 4), (43, 4), tubamusicmaker_one],
1223         [(43, 4), (46, 4), tubamusicmaker_one],
1224         [(50, 4), (53, 4), tubamusicmaker_three],
1225         [(55, 4), (56, 4), tubamusicmaker_two],
1226         [(61, 4), (64, 4), tubamusicmaker_one],
1227         [(64, 4), (65, 4), tubamusicmaker_one],
1228         [(68, 4), (70, 4), tubamusicmaker_three],
1229         [(70, 4), (72, 4), tubamusicmaker_two],
1230         [(72, 4), (74, 4), tubamusicmaker_two],
1231         [(79, 4), (80, 4), tubamusicmaker_three],
1232         [(82, 4), (85, 4), tubamusicmaker_two],
1233         [(89, 4), (94, 4), tubamusicmaker_one],
1234         [(95, 4), (97, 4), tubamusicmaker_two],
1235         [(100, 4), (104, 4), tubamusicmaker_three],
1236         [(109, 4), (110, 4), tubamusicmaker_two],
1237         [(110, 4), (111, 4), tubamusicmaker_one],
1238         [(112, 4), (114, 4), tubamusicmaker_one],
1239         [(114, 4), (116, 4), tubamusicmaker_one],
1240         [(117, 4), (119, 4), tubamusicmaker_one],
1241         [(119, 4), (121, 4), tubamusicmaker_one],
1242         [(122, 4), (123, 4), tubamusicmaker_one],
1243         [(123, 4), (125, 4), tubamusicmaker_one],
1244         [(133, 4), (136, 4), tubamusicmaker_two],
1245         [(142, 4), (146, 4), tubamusicmaker_two],
1246         [(146, 4), (150, 4), tubamusicmaker_two],

```

```

1247     [(154, 4), (157, 4), tubamusicmaker_three],
1248     [(159, 4), (163, 4), tubamusicmaker_three],
1249     [(166, 4), (168, 4), tubamusicmaker_three],
1250     [(172, 4), (175, 4), tubamusicmaker_three],
1251     [(177, 4), (179, 4), tubamusicmaker_three],
1252     [(179, 4), (181, 4), tubamusicmaker_three],
1253     [(184, 4), (186, 4), tubamusicmaker_three],
1254     [(186, 4), (190, 4), tubamusicmaker_three],
1255 ]
1256 ])
1257
1258 voice_12_timespan_list = abjad.TimespanList([
1259     abjad.AnnotatedTimespan(
1260         start_offset=start_offset,
1261         stop_offset=stop_offset,
1262         annotation=MusicSpecifier(
1263             music_maker=music_maker,
1264             voice_name='Voice 12',
1265         ),
1266     )
1267     for start_offset, stop_offset, music_maker in [
1268         [(0, 4), (5, 4), bassmusicmaker_one],
1269         [(8, 4), (10, 4), bassmusicmaker_two],
1270         [(14, 4), (18, 4), bassmusicmaker_three],
1271         [(21, 4), (22, 4), bassmusicmaker_one],
1272         [(22, 4), (23, 4), bassmusicmaker_one],
1273         [(38, 4), (40, 4), bassmusicmaker_two],
1274         [(41, 4), (43, 4), bassmusicmaker_one],
1275         [(43, 4), (46, 4), bassmusicmaker_one],
1276         [(50, 4), (53, 4), bassmusicmaker_three],
1277         [(55, 4), (56, 4), bassmusicmaker_two],
1278         [(61, 4), (64, 4), bassmusicmaker_one],
1279         [(64, 4), (65, 4), bassmusicmaker_one],
1280         [(68, 4), (70, 4), bassmusicmaker_three],
1281         [(70, 4), (72, 4), bassmusicmaker_two],
1282         [(72, 4), (74, 4), bassmusicmaker_two],
1283         [(79, 4), (80, 4), bassmusicmaker_three],
1284         [(82, 4), (85, 4), bassmusicmaker_two],
1285         [(89, 4), (94, 4), bassmusicmaker_one],
1286         [(95, 4), (97, 4), bassmusicmaker_two],
1287         [(100, 4), (104, 4), bassmusicmaker_three],
1288         [(109, 4), (110, 4), bassmusicmaker_two],
1289         [(110, 4), (111, 4), bassmusicmaker_one],
1290         [(112, 4), (114, 4), bassmusicmaker_one],
1291         [(114, 4), (116, 4), bassmusicmaker_one],
1292         [(117, 4), (119, 4), bassmusicmaker_one],
1293         [(119, 4), (121, 4), bassmusicmaker_one],
1294         [(122, 4), (123, 4), bassmusicmaker_one],
1295         [(123, 4), (125, 4), bassmusicmaker_one],
1296         [(133, 4), (136, 4), bassmusicmaker_two],
1297         [(142, 4), (146, 4), bassmusicmaker_two],
1298         [(146, 4), (150, 4), bassmusicmaker_two],
1299         [(152, 4), (154, 4), bassmusicmaker_three],
1300         [(154, 4), (156, 4), bassmusicmaker_three],

```

```

1301     [(159, 4), (161, 4), bassmusicmaker_three],
1302     [(165, 4), (168, 4), bassmusicmaker_three],
1303     [(170, 4), (172, 4), bassmusicmaker_three],
1304     [(172, 4), (174, 4), bassmusicmaker_three],
1305     [(177, 4), (179, 4), bassmusicmaker_three],
1306     [(183, 4), (186, 4), bassmusicmaker_three],
1307     [(186, 4), (190, 4), bassmusicmaker_three],
1308 ]
1309 ])
1310
1311 all_timespan_lists = {
1312     'Voice 1': voice_1_timespan_list,
1313     'Voice 2': voice_2_timespan_list,
1314     'Voice 3': voice_3_timespan_list,
1315     'Voice 4': voice_4_timespan_list,
1316     'Voice 5': voice_5_timespan_list,
1317     'Voice 6': voice_6_timespan_list,
1318     'Voice 7': voice_7_timespan_list,
1319     'Voice 8': voice_8_timespan_list,
1320     'Voice 9': voice_9_timespan_list,
1321     'Voice 10': voice_10_timespan_list,
1322     'Voice 11': voice_11_timespan_list,
1323     'Voice 12': voice_12_timespan_list,
1324 }
1325
1326 global_timespan = abjad.Timespan(
1327     start_offset=0,
1328     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
1329 )
1330
1331 for voice_name, timespan_list in all_timespan_lists.items():
1332     silences = abjad.TimespanList([global_timespan])
1333     silences.extend(timespan_list)
1334     silences.sort()
1335     silences.compute_logical_xor()
1336     for silence_timespan in silences:
1337         timespan_list.append(
1338             abjad.AnnotatedTimespan(
1339                 start_offset=silence_timespan.start_offset,
1340                 stop_offset=silence_timespan.stop_offset,
1341                 annotation=MusicSpecifier(
1342                     music_maker=None,
1343                     voice_name=voice_name,
1344                 ),
1345             )
1346         )
1347     timespan_list.sort()
1348
1349 for voice_name, timespan_list in all_timespan_lists.items():
1350     shards = timespan_list.split_at_offsets(bounds)
1351     split_timespan_list = abjad.TimespanList()
1352     for shard in shards:
1353         split_timespan_list.extend(shard)
1354     split_timespan_list.sort()

```

```

1355     all_timespan_lists[voice_name] = timespan_list
1356
1357 score = abjad.Score([
1358     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 1')
1359     ,
1360     abjad.StaffGroup(
1361         [
1362             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
1363             lilypond_type='Staff',),
1364             abjad.Staff([abjad.Voice(name='Voice 2')], name='Staff 2',
1365             lilypond_type='Staff',),
1366             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
1367             lilypond_type='Staff',),
1368         ],
1369         name='Staff Group 1',
1370     ),
1371     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 2')
1372     ,
1373     abjad.StaffGroup(
1374         [
1375             abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
1376             lilypond_type='Staff',),
1377             abjad.Staff([abjad.Voice(name='Voice 5')], name='Staff 5',
1378             lilypond_type='Staff',),
1379             abjad.Staff([abjad.Voice(name='Voice 6')], name='Staff 6',
1380             lilypond_type='Staff',),
1381             abjad.Staff([abjad.Voice(name='Voice 7')], name='Staff 7',
1382             lilypond_type='Staff',),
1383         ],
1384         name='Staff Group 2',
1385     ),
1386     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 3')
1387     ,
1388     abjad.StaffGroup(
1389         [
1390             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1391             lilypond_type='Staff',),
1392             abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1393             lilypond_type='Staff',),
1394             abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1395             lilypond_type='Staff',),
1396             abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1397             lilypond_type='Staff',),
1398             abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1399             lilypond_type='Staff',),
1400         ],
1401         name='Staff Group 3',
1402     )
1403 ],
1404 )
1405
1406 for time_signature in time_signatures:
1407     skip = abjad.Skip(1, multiplier=(time_signature))
1408     abjad.attach(time_signature, skip)

```

```

1394     score['Global Context 1'].append(skip)
1395
1396 for time_signature in time_signatures:
1397     skip = abjad.Skip(1, multiplier=(time_signature))
1398     abjad.attach(time_signature, skip)
1399     score['Global Context 2'].append(skip)
1400
1401 for time_signature in time_signatures:
1402     skip = abjad.Skip(1, multiplier=(time_signature))
1403     abjad.attach(time_signature, skip)
1404     score['Global Context 3'].append(skip)
1405
1406 print('Making containers ...')
1407
1408 def make_container(music_maker, durations):
1409     selections = music_maker(durations)
1410     container = abjad.Container([])
1411     container.extend(selections)
1412     return container
1413
1414 def key_function(timespan):
1415     return timespan.annotation.music_maker or silence_maker
1416
1417 for voice_name, timespan_list in all_timespan_lists.items():
1418     for music_maker, grouper in itertools.groupby(
1419         timespan_list,
1420         key=key_function,
1421     ):
1422         durations = [timespan.duration for timespan in grouper]
1423         container = make_container(music_maker, durations)
1424         voice = score[voice_name]
1425         voice.append(container)
1426
1427 print('Splitting and rewriting ...')
1428
1429 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
1430     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1431         time_signature = time_signatures[i]
1432         abjad.mutate(shard).rewrite_meter(time_signature)
1433
1434 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
1435     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1436         time_signature = time_signatures[i]
1437         abjad.mutate(shard).rewrite_meter(time_signature)
1438
1439 for voice in abjad.iterate(score['Staff Group 3']).components(abjad.Voice):
1440     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1441         time_signature = time_signatures[i]
1442         abjad.mutate(shard).rewrite_meter(time_signature)
1443
1444 print('Beaming runs ...')
1445
1446 for voice in abjad.select(score).components(abjad.Voice):
1447     for run in abjad.select(voice).runs():

```

```

1448     if 1 < len(run):
1449         specifier = abjadext.rmakers.BeamSpecifier(
1450             beam_each_division=False,
1451             )
1452         specifier(run)
1453         abjad.attach(abjad.StartBeam(), run[0])
1454         abjad.attach(abjad.StopBeam(), run[-1])
1455         for leaf in run:
1456             if abjad.Duration(1, 4) <= leaf.written_duration:
1457                 continue
1458             previous_leaf = abjad.inspect(leaf).leaf(-1)
1459             next_leaf = abjad.inspect(leaf).leaf(1)
1460             if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
1461                 abjad.Duration(1, 4) <= next_leaf.written_duration):
1462                 left = previous_leaf.written_duration.flag_count
1463                 right = leaf.written_duration.flag_count
1464                 beam_count = abjad.BeamCount(
1465                     left=left,
1466                     right=right,
1467                     )
1468                 abjad.attach(beam_count, leaf)
1469                 continue
1470             if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
1471                 abjad.Duration(1, 4) <= previous_leaf.written_duration):
1472                 left = leaf.written_duration.flag_count
1473                 right = next_leaf.written_duration.flag_count
1474                 beam_count = abjad.BeamCount(
1475                     left=left,
1476                     right=right,
1477                     )
1478                 abjad.attach(beam_count, leaf)
1479
1480 print('Beautifying score ...')
1481 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1482     for selection in abjad.select(staff).components(abjad.Rest).
1483         group_by_contiguity():
1484             start_command = abjad.LilyPondLiteral(
1485                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1486                 startStaff',
1487                 format_slot='before',
1488                 )
1489             stop_command = abjad.LilyPondLiteral(
1490                 r'\stopStaff \startStaff',
1491                 format_slot='after',
1492                 )
1493             abjad.attach(start_command, selection[0])
1494             abjad.attach(stop_command, selection[-1])
1495
1496 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1497     for selection in abjad.select(staff).components(abjad.Rest).
1498         group_by_contiguity():
1499             start_command = abjad.LilyPondLiteral(
1500                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1501                 startStaff',

```

```

1498     format_slot='before',
1499 )
1500 stop_command = abjad.LilyPondLiteral(
1501     r'\stopStaff \startStaff',
1502     format_slot='after',
1503 )
1504 abjad.attach(start_command, selection[0])
1505 abjad.attach(stop_command, selection[-1])
1506
1507 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1508     for selection in abjad.select(staff).components(abjad.Rest).
1509         group_by_contiguity():
1510             start_command = abjad.LilyPondLiteral(
1511                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1512                 startStaff',
1513                 format_slot='before',
1514             )
1515             stop_command = abjad.LilyPondLiteral(
1516                 r'\stopStaff \startStaff',
1517                 format_slot='after',
1518             )
1519             abjad.attach(start_command, selection[0])
1520             abjad.attach(stop_command, selection[-1])
1521
1522 print('Stopping Hairpins ...')
1523 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1524     for rest in abjad.iterate(staff).components(abjad.Rest):
1525         previous_leaf = abjad.inspect(rest).leaf(-1)
1526         if isinstance(previous_leaf, abjad.Note):
1527             abjad.attach(abjad.StopHairpin(), rest)
1528         elif isinstance(previous_leaf, abjad.Chord):
1529             abjad.attach(abjad.StopHairpin(), rest)
1530         elif isinstance(previous_leaf, abjad.Rest):
1531             pass
1532
1533 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1534     for rest in abjad.iterate(staff).components(abjad.Rest):
1535         previous_leaf = abjad.inspect(rest).leaf(-1)
1536         if isinstance(previous_leaf, abjad.Note):
1537             abjad.attach(abjad.StopHairpin(), rest)
1538         elif isinstance(previous_leaf, abjad.Chord):
1539             abjad.attach(abjad.StopHairpin(), rest)
1540         elif isinstance(previous_leaf, abjad.Rest):
1541             pass
1542
1543 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1544     for rest in abjad.iterate(staff).components(abjad.Rest):
1545         previous_leaf = abjad.inspect(rest).leaf(-1)
1546         if isinstance(previous_leaf, abjad.Note):
1547             abjad.attach(abjad.StopHairpin(), rest)
1548         elif isinstance(previous_leaf, abjad.Chord):
1549             abjad.attach(abjad.StopHairpin(), rest)
1550         elif isinstance(previous_leaf, abjad.Rest):
1551             pass

```

```

1550
1551 print('Adding pitch material ...')
1552 def cyc(lst):
1553     count = 0
1554     while True:
1555         yield lst[count%len(lst)]
1556         count += 1
1557
1558 print('Adding attachments ...')
1559 bar_line = abjad.BarLine('|||')
1560 metro = abjad.MetronomeMark((1, 4), 108)
1561 markup1 = abjad.Markup(r'\bold { A }')
1562 markup2 = abjad.Markup(r'\bold { B }')
1563 markup3 = abjad.Markup(r'\bold { C }')
1564 markup4 = abjad.Markup(r'\bold { D }')
1565 markup5 = abjad.Markup(r'\bold { E }')
1566 markup6 = abjad.Markup(r'\bold { F }')
1567 mark1 = abjad.RehearsalMark(markup=markup1)
1568 mark2 = abjad.RehearsalMark(markup=markup2)
1569 mark3 = abjad.RehearsalMark(markup=markup3)
1570 mark4 = abjad.RehearsalMark(markup=markup4)
1571 mark5 = abjad.RehearsalMark(markup=markup5)
1572 mark6 = abjad.RehearsalMark(markup=markup6)
1573
1574 instruments1 = cyc([
1575     abjad.Flute(),
1576     abjad.ClarinetInBFlat(),
1577     abjad.Bassoon(),
1578 ])
1579
1580 instruments2 = cyc([
1581     abjad.FrenchHorn(),
1582     abjad.Trombone(),
1583     abjad.TenorTrombone(),
1584     abjad.Tuba(),
1585 ])
1586
1587 instruments3 = cyc([
1588     abjad.Violin(),
1589     abjad.Violin(),
1590     abjad.Viola(),
1591     abjad.Cello(),
1592     abjad.Contrabass(),
1593 ])
1594
1595 clefs1 = cyc([
1596     abjad.Clef('treble'),
1597     abjad.Clef('treble'),
1598     abjad.Clef('bass'),
1599 ])
1600
1601 clefs2 = cyc([
1602     abjad.Clef('bass'),
1603     abjad.Clef('treble'),

```

```

1604     abjad.Clef('bass'),
1605     abjad.Clef('bass'),
1606   ])
1607
1608 clefs3 = cyc([
1609     abjad.Clef('treble'),
1610     abjad.Clef('treble'),
1611     abjad.Clef('alto'),
1612     abjad.Clef('bass'),
1613     abjad.Clef('bass'),
1614   ])
1615
1616 abbreviations1 = cyc([
1617     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),
1618     abjad.MarginMarkup(markup=abjad.Markup('cl.'),),
1619     abjad.MarginMarkup(markup=abjad.Markup('bssn.'),),
1620   ])
1621
1622 abbreviations2 = cyc([
1623     abjad.MarginMarkup(markup=abjad.Markup('hr.'),),
1624     abjad.MarginMarkup(markup=abjad.Markup('trp.'),),
1625     abjad.MarginMarkup(markup=abjad.Markup('trmb.'),),
1626     abjad.MarginMarkup(markup=abjad.Markup('tb.'),),
1627   ])
1628
1629 abbreviations3 = cyc([
1630     abjad.MarginMarkup(markup=abjad.Markup('vln.I'),),
1631     abjad.MarginMarkup(markup=abjad.Markup('vln.II'),),
1632     abjad.MarginMarkup(markup=abjad.Markup('vla.'),),
1633     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
1634     abjad.MarginMarkup(markup=abjad.Markup('cb.'),),
1635   ])
1636
1637 names1 = cyc([
1638     abjad.StartMarkup(markup=abjad.Markup('Flute'),),
1639     abjad.StartMarkup(markup=abjad.Markup('Clarinet'),),
1640     abjad.StartMarkup(markup=abjad.Markup('Bassoon'),),
1641   ])
1642
1643 names2 = cyc([
1644     abjad.StartMarkup(markup=abjad.Markup('Horn'),),
1645     abjad.StartMarkup(markup=abjad.Markup('Trumpet'),),
1646     abjad.StartMarkup(markup=abjad.Markup('Trombone'),),
1647     abjad.StartMarkup(markup=abjad.Markup('Tuba'),),
1648   ])
1649
1650 names3 = cyc([
1651     abjad.StartMarkup(markup=abjad.Markup('Violin I'),),
1652     abjad.StartMarkup(markup=abjad.Markup('Violin II'),),
1653     abjad.StartMarkup(markup=abjad.Markup('Viola'),),
1654     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
1655     abjad.StartMarkup(markup=abjad.Markup('Contrabass'),),
1656   ])
1657

```

```

1658 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1659     leaf1 = abjad.select(staff).leaves()[0]
1660     abjad.attach(next(instruments1), leaf1)
1661     abjad.attach(next(abbreviations1), leaf1)
1662     abjad.attach(next(names1), leaf1)
1663     abjad.attach(next(clefs1), leaf1)
1664
1665 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1666     leaf1 = abjad.select(staff).leaves()[0]
1667     abjad.attach(next(instruments2), leaf1)
1668     abjad.attach(next(abbreviations2), leaf1)
1669     abjad.attach(next(names2), leaf1)
1670     abjad.attach(next(clefs2), leaf1)
1671
1672 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1673     leaf1 = abjad.select(staff).leaves()[0]
1674     abjad.attach(next(instruments3), leaf1)
1675     abjad.attach(next(abbreviations3), leaf1)
1676     abjad.attach(next(names3), leaf1)
1677     abjad.attach(next(clefs3), leaf1)
1678
1679 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
1680     leaf1 = abjad.select(staff).leaves()[0]
1681     last_leaf = abjad.select(staff).leaves()[-1]
1682     abjad.attach(metro, leaf1)
1683     abjad.attach(bar_line, last_leaf)
1684
1685 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff)[0]:
1686     leaf1 = abjad.select(staff).leaves()[0]
1687     last_leaf = abjad.select(staff).leaves()[-1]
1688     abjad.attach(metro, leaf1)
1689     abjad.attach(bar_line, last_leaf)
1690
1691 for staff in abjad.select(score['Staff Group 3']).components(abjad.Staff)[0]:
1692     leaf1 = abjad.select(staff).leaves()[0]
1693     last_leaf = abjad.select(staff).leaves()[-1]
1694     abjad.attach(metro, leaf1)
1695     abjad.attach(bar_line, last_leaf)
1696
1697 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1698     leaf1 = abjad.select(staff).leaves()[7]
1699     abjad.attach(mark1, leaf1)
1700
1701 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1702     leaf1 = abjad.select(staff).leaves()[7]
1703     abjad.attach(mark1, leaf1)
1704
1705 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1706     leaf1 = abjad.select(staff).leaves()[7]
1707     abjad.attach(mark1, leaf1)
1708
1709 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1710     leaf2 = abjad.select(staff).leaves()[16]
1711     abjad.attach(mark2, leaf2)

```

```

1712
1713 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1714     leaf2 = abjad.select(staff).leaves()[16]
1715     abjad.attach(mark2, leaf2)
1716
1717 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1718     leaf2 = abjad.select(staff).leaves()[16]
1719     abjad.attach(mark2, leaf2)
1720
1721 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1722     leaf3 = abjad.select(staff).leaves()[22]
1723     abjad.attach(mark3, leaf3)
1724
1725 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1726     leaf3 = abjad.select(staff).leaves()[22]
1727     abjad.attach(mark3, leaf3)
1728
1729 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1730     leaf3 = abjad.select(staff).leaves()[22]
1731     abjad.attach(mark3, leaf3)
1732
1733 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1734     leaf4 = abjad.select(staff).leaves()[29]
1735     abjad.attach(mark4, leaf4)
1736
1737 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1738     leaf4 = abjad.select(staff).leaves()[29]
1739     abjad.attach(mark4, leaf4)
1740
1741 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1742     leaf4 = abjad.select(staff).leaves()[29]
1743     abjad.attach(mark4, leaf4)
1744
1745 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1746     leaf5 = abjad.select(staff).leaves()[34]
1747     abjad.attach(mark5, leaf5)
1748
1749 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1750     leaf5 = abjad.select(staff).leaves()[34]
1751     abjad.attach(mark5, leaf5)
1752
1753 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1754     leaf5 = abjad.select(staff).leaves()[34]
1755     abjad.attach(mark5, leaf5)
1756
1757 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1758     leaf6 = abjad.select(staff).leaves()[39]
1759     abjad.attach(mark6, leaf6)
1760
1761 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1762     leaf6 = abjad.select(staff).leaves()[39]
1763     abjad.attach(mark6, leaf6)
1764
1765 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):

```

```

1766 leaf6 = abjad.select(staff).leaves()[39]
1767 abjad.attach(mark6, leaf6)
1768
1769 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1770     abjad.Instrument.transpose_from_sounding_pitch(staff)
1771
1772 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1773     abjad.Instrument.transpose_from_sounding_pitch(staff)
1774
1775 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1776     abjad.Instrument.transpose_from_sounding_pitch(staff)
1777
1778 score_file = abjad.LilyPondFile.new(
1779     score,
1780     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
1781     _stylesheets/abjad.ily'],
1782 )
1783
1784 abjad.SegmentMaker.comment_measure_numbers(score)
1785 ######
1786 directory = '/Users/evansdsg2/Scores//tianshu/tianshu/Segments/Segment_I'
1787 pdf_path = f'{directory}/Segment_I.pdf'
1788 path = pathlib.Path('Segment_I.pdf')
1789 if path.exists():
1790     print(f'Removing {pdf_path} ...')
1791     path.unlink()
1792 time_1 = time.time()
1793 print(f'Persisting {pdf_path} ...')
1794 result = abjad.persist(score_file).as_pdf(pdf_path)
1795 print(result[0])
1796 print(result[1])
1797 print(result[2])
1798 success = result[3]
1799 if success is False:
1800     print('LilyPond failed!')
1801 time_2 = time.time()
1802 total_time = time_2 - time_1
1803 print(f'Total time: {total_time} seconds')
1804 if path.exists():
1805     print(f'Opening {pdf_path} ...')
1806     os.system(f'open {pdf_path}')

```

**Code Example A.11:** Tianshu Segment\_I

### A.3.1.2 SEGMENT\_II

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers

```

```

7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (5, 4), (2, 4), (4, 4), (3, 4), (4, 4), (4, 4),
17         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
18         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
19         (2, 4), (3, 4), (4, 4), (3, 4), (4, 4), (3, 4),
20         (5, 4), (3, 4), (4, 4), (3, 4), (3, 4), (3, 4),
21         (4, 4), (5, 4), (4, 4), (3, 4), (5, 4), (5, 4),
22         (5, 4), (5, 4), (4, 4), (4, 4), (5, 4), (5, 4),
23         (4, 4), (4, 4), (3, 4), (4, 4), (4, 4), (3, 4),
24         (5, 4),
25     ]
26 ]
27
28 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures
29     ])
30
31 def reduceMod5(rw):
32     return [(x % 6) for x in rw]
33
34 def reduceMod6(rw):
35     return [(x % 7) for x in rw]
36
37 def reduceMod7(rw):
38     return [(x % 8) for x in rw]
39
40 def reduceMod9(rw):
41     return [(x % 10) for x in rw]
42
43 def reduceMod11(rw):
44     return [(x % 12) for x in rw]
45
46 def reduceMod15(rw):
47     return [(x % 16) for x in rw]
48
49 seed(1)
50 flute_random_walk_one = []
51 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
52 for i in range(1, 1000):
53     movement = -1 if random() < 0.5 else 1
54     value = flute_random_walk_one[i-1] + movement
55     flute_random_walk_one.append(value)
56 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
57 flute_chord_one = [0, 10, 16, 18, 25, 26, 25, 18, 16, 10]
58 flute_notes_one = [flute_chord_one[x] for x in reduceMod9(
59     flute_random_walk_one)]

```

```

59 seed(2)
60 clarinet_random_walk_one = []
61 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
62 for i in range(1, 1000):
63     movement = -1 if random() < 0.5 else 1
64     value = clarinet_random_walk_one[i-1] + movement
65     clarinet_random_walk_one.append(value)
66 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
67 clarinet_chord_one = [-5, 0, 10, 16, 18, 25, 18, 16, 10, 0 ]
68 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod9(
69     clarinet_random_walk_one)]
70
71 seed(3)
72 bassoon_random_walk_one = []
73 bassoon_random_walk_one.append(-1 if random() < 0.5 else 1)
74 for i in range(1, 1000):
75     movement = -1 if random() < 0.5 else 1
76     value = bassoon_random_walk_one[i-1] + movement
77     bassoon_random_walk_one.append(value)
78 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
79 bassoon_chord_one = [-19, -16, -5, 0, 10, 0, -5, -16 ]
80 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod7(
81     bassoon_random_walk_one)]
82
83 seed(4)
84 horn_random_walk_one = []
85 horn_random_walk_one.append(-1 if random() < 0.5 else 1)
86 for i in range(1, 1000):
87     movement = -1 if random() < 0.5 else 1
88     value = horn_random_walk_one[i-1] + movement
89     horn_random_walk_one.append(value)
90 horn_random_walk_one = [abs(x) for x in horn_random_walk_one]
91 horn_chord_one = [-19, -16, -5, 0, -5, -16 ]
92 horn_notes_one = [horn_chord_one[x] for x in reduceMod5(horn_random_walk_one)]
93
94 seed(5)
95 trumpet_random_walk_one = []
96 trumpet_random_walk_one.append(-1 if random() < 0.5 else 1)
97 for i in range(1, 1000):
98     movement = -1 if random() < 0.5 else 1
99     value = trumpet_random_walk_one[i-1] + movement
100    trumpet_random_walk_one.append(value)
101 trumpet_random_walk_one = [abs(x) for x in trumpet_random_walk_one]
102 trumpet_chord_one = [-5, 0, 10, 16, 18, 16, 10, 0 ]
103 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod7(
104     trumpet_random_walk_one)]
105
106 seed(6)
107 trombone_random_walk_one = []
108 trombone_random_walk_one.append(-1 if random() < 0.5 else 1)
109 for i in range(1, 1000):
110     movement = -1 if random() < 0.5 else 1
111     value = trombone_random_walk_one[i-1] + movement
112     trombone_random_walk_one.append(value)

```

```

110 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
111 trombone_chord_one = [-19, -16, -5, 0, -5, -16]
112 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod5(
113     trombone_random_walk_one)]
114
115 seed(7)
116 tuba_random_walk_one = []
117 tuba_random_walk_one.append(-1 if random() < 0.5 else 1)
118 for i in range(1, 1000):
119     movement = -1 if random() < 0.5 else 1
120     value = tuba_random_walk_one[i-1] + movement
121     tuba_random_walk_one.append(value)
122 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
123 tuba_chord_one = [-27, -19, -16, -5, -16, -19]
124 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod5(tuba_random_walk_one)]
125
126 seed(8)
127 violin1_random_walk_one = []
128 violin1_random_walk_one.append(-1 if random() < 0.5 else 1)
129 for i in range(1, 1000):
130     movement = -1 if random() < 0.5 else 1
131     value = violin1_random_walk_one[i-1] + movement
132     violin1_random_walk_one.append(value)
133 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
134 violin1_chord_one = [-5, 0, 10, 16, 18, 25, 26, 34, 38, 34, 26, 25, 18, 16,
135     10, 0]
136 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod15(
137     violin1_random_walk_one)]
138
139 seed(9)
140 violin2_random_walk_one = []
141 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
142 for i in range(1, 1000):
143     movement = -1 if random() < 0.5 else 1
144     value = violin2_random_walk_one[i-1] + movement
145     violin2_random_walk_one.append(value)
146 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
147 violin2_chord_one = [-5, 0, 10, 16, 18, 25, 26, 25, 18, 16, 10, 0]
148 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod11(
149     violin2_random_walk_one)]
150
151 seed(10)
152 viola_random_walk_one = []
153 viola_random_walk_one.append(-1 if random() < 0.5 else 1)
154 for i in range(1, 1000):
155     movement = -1 if random() < 0.5 else 1
156     value = viola_random_walk_one[i-1] + movement
157     viola_random_walk_one.append(value)
158 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
159 viola_chord_one = [-5, 0, 10, 16, 18, 16, 10, 0]
160 viola_notes_one = [viola_chord_one[x] for x in reduceMod7(
161     viola_random_walk_one)]
162
163 seed(11)

```

```

159 cello_random_walk_one = []
160 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
161 for i in range(1, 1000):
162     movement = -1 if random() < 0.5 else 1
163     value = cello_random_walk_one[i-1] + movement
164     cello_random_walk_one.append(value)
165 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
166 cello_chord_one = [-19, -16, -5, 0, 10, 16, 18, 16, 10, 0, -5, -16]
167 cello_notes_one = [cello_chord_one[x] for x in reduceMod11(
168     cello_random_walk_one)]
169
169 seed(12)
170 bass_random_walk_one = []
171 bass_random_walk_one.append(-1 if random() < 0.5 else 1)
172 for i in range(1, 1000):
173     movement = -1 if random() < 0.5 else 1
174     value = bass_random_walk_one[i-1] + movement
175     bass_random_walk_one.append(value)
176 bass_random_walk_one = [abs(x) for x in bass_random_walk_one]
177 bass_chord_one = [-27, -19, -16, -5, 0, -5, -19]
178 bass_notes_one = [bass_chord_one[x] for x in reduceMod6(bass_random_walk_one)]
179
180 flute_scale = [18]
181 clarinet_scale = [0]
182 bassoon_scale = [-19]
183 horn_scale = [-19]
184 trumpet_scale = [18]
185 trombone_scale = [-0]
186 tuba_scale = [-19]
187 violin1_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25, 24.5,
187     24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5, 20, 20.5, 21,
187     21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26, 26.5, 27, 27.5, 28,
187     28.5, 29, 29.5, ]
188 violin2_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14, 13.5,
188     13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5, 10, 10.5,
188     11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16, 16.5, 17, 17.5, 18,
188     18.5, ]
189 viola_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1, 0.5,
189     0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1,
189     1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]
190 cello_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8, -8.5, -9,
190     -9.5, -10, -10.5, -11, -11.5, -12, -12.5, -13, -13.5, -14, -13.5, -13,
190     -12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8, -7.5, -7, -6.5,
190     -6, -5.5, -5, -4.5, -4, -3.5, ]
191 bass_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18, -18.5, -19,
191     -19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5, -24, -24.5, -25,
191     -24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21, -20.5, -20, -19.5, -19,
191     -18.5, -18, -17.5, -17, -16.5, -16, -15.5, -15, -14.5, ]
192
193 seed(1)
194 flute_random_walk_two = []
195 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
196 for i in range(1, 1000):
197     movement = -1 if random() < 0.5 else 1

```

```

198     value = flute_random_walk_two[i-1] + movement
199     flute_random_walk_two.append(value)
200 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
201 flute_chord_two = [0, 8, 14, 18, 27, 28, 27, 18, 14, 8 ]
202 flute_notes_two = [flute_chord_two[x] for x in reduceMod9(
203     flute_random_walk_two)]
204
205 seed(2)
206 clarinet_random_walk_two = []
207 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
208 for i in range(1, 1000):
209     movement = -1 if random() < 0.5 else 1
210     value = clarinet_random_walk_two[i-1] + movement
211     clarinet_random_walk_two.append(value)
212 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
213 clarinet_chord_two = [-3, 0, 8, 14, 18, 14, 8, 0 ]
214 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod7(
215     clarinet_random_walk_two)]
216
217 seed(3)
218 bassoon_random_walk_two = []
219 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
220 for i in range(1, 1000):
221     movement = -1 if random() < 0.5 else 1
222     value = bassoon_random_walk_two[i-1] + movement
223     bassoon_random_walk_two.append(value)
224 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
225 bassoon_chord_two = [-19, -14, -3, 0, 8, 0, -3, -14 ]
226 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod7(
227     bassoon_random_walk_two)]
228
229 seed(4)
230 horn_random_walk_two = []
231 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
232 for i in range(1, 1000):
233     movement = -1 if random() < 0.5 else 1
234     value = horn_random_walk_two[i-1] + movement
235     horn_random_walk_two.append(value)
236 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
237 horn_chord_two = [-19, -14, -3, 0, 8, 0, -3, -14 ]
238 horn_notes_two = [horn_chord_two[x] for x in reduceMod7(horn_random_walk_two)]
239
240 seed(5)
241 trumpet_random_walk_two = []
242 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)
243 for i in range(1, 1000):
244     movement = -1 if random() < 0.5 else 1
245     value = trumpet_random_walk_two[i-1] + movement
246     trumpet_random_walk_two.append(value)
247 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]
248 trumpet_chord_two = [-3, 0, 8, 14, 18, 14, 8, 0 ]
249 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod7(
250     trumpet_random_walk_two)]

```

```

248 seed(6)
249 trombone_random_walk_two = []
250 trombone_random_walk_two.append(-1 if random() < 0.5 else 1)
251 for i in range(1, 1000):
252     movement = -1 if random() < 0.5 else 1
253     value = trombone_random_walk_two[i-1] + movement
254     trombone_random_walk_two.append(value)
255 trombone_random_walk_two = [abs(x) for x in trombone_random_walk_two]
256 trombone_chord_two = [-19, -14, -3, 0, -3, -14]
257 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod5(
258     trombone_random_walk_two)]
259
259 seed(7)
260 tuba_random_walk_two = []
261 tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
262 for i in range(1, 1000):
263     movement = -1 if random() < 0.5 else 1
264     value = tuba_random_walk_two[i-1] + movement
265     tuba_random_walk_two.append(value)
266 tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
267 tuba_chord_two = [-29, -19, -14, -3, -14, -19]
268 tuba_notes_two = [tuba_chord_two[x] for x in reduceMod5(tuba_random_walk_two)]
269
270 seed(8)
271 violin1_random_walk_two = []
272 violin1_random_walk_two.append(-1 if random() < 0.5 else 1)
273 for i in range(1, 1000):
274     movement = -1 if random() < 0.5 else 1
275     value = violin1_random_walk_two[i-1] + movement
276     violin1_random_walk_two.append(value)
277 violin1_random_walk_two = [abs(x) for x in violin1_random_walk_two]
278 violin1_chord_two = [36, 35, 28, 27, 18, 14, 8, 0, -3, 0, 8, 14, 18, 27, 28,
279     35]
279 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod15(
280     violin1_random_walk_two)]
280
281 seed(9)
282 violin2_random_walk_two = []
283 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
284 for i in range(1, 1000):
285     movement = -1 if random() < 0.5 else 1
286     value = violin2_random_walk_two[i-1] + movement
287     violin2_random_walk_two.append(value)
288 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
289 violin2_chord_two = [-3, 0, 8, 14, 18, 27, 18, 14, 8, 0]
290 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod9(
291     violin2_random_walk_two)]
292
292 seed(10)
293 viola_random_walk_two = []
294 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
295 for i in range(1, 1000):
296     movement = -1 if random() < 0.5 else 1
297     value = viola_random_walk_two[i-1] + movement

```

```

298     viola_random_walk_two.append(value)
299 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
300 viola_chord_two = [-3, 0, 8, 14, 18, 14, 8, 0 ]
301 viola_notes_two = [viola_chord_two[x] for x in reduceMod7(
    viola_random_walk_two)]
302
303 seed(11)
304 cello_random_walk_two = []
305 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
306 for i in range(1, 1000):
    movement = -1 if random() < 0.5 else 1
    value = cello_random_walk_two[i-1] + movement
    cello_random_walk_two.append(value)
307 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
308 cello_chord_two = [-19, -14, -3, 0, 8, 14, 18, 14, 8, 0, -3, -14 ]
309 cello_notes_two = [cello_chord_two[x] for x in reduceMod11(
    cello_random_walk_two)]
310
311 seed(12)
312 bass_random_walk_two = []
313 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
314 for i in range(1, 1000):
    movement = -1 if random() < 0.5 else 1
    value = bass_random_walk_two[i-1] + movement
    bass_random_walk_two.append(value)
315 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
316 bass_chord_two = [-29, -19, -14, -3, 0, -3, -14, -19 ]
317 bass_notes_two = [bass_chord_two[x] for x in reduceMod7(bass_random_walk_two)]
318
319 rmaker_one = abjadext.rmakers.EvenDivisionRhythmMaker(
320     denominators=[16, 16, 8, 16, 4, 16, 8],
321     extra_counts_per_division=[1, 1, 0, -1, 0, 1, -1, 0],
322     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
323         left_classes=[abjad.Rest],
324         left_counts=[2],
325         right_classes=[abjad.Rest],
326         right_counts=[1],
327         outer_divisions_only=True,
328         ),
329     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
330         trivialize=True,
331         extract_trivial=True,
332         rewrite_rest_filled=True,
333         ),
334     )
335
336 rmaker_two = abjadext.rmakers.IncisedRhythmMaker(
337     inciseSpecifier=abjadext.rmakers.InciseSpecifier(
338         prefix_talea=[-1],
339         prefix_counts=[0, 1],
340         suffix_talea=[-1],
341         suffix_counts=[1],
342         talea_denominator=16,
343         ),
344
345
346
347
348
349

```

```

350 )
351
352 rmaker_three = abjadext.rmakers.TupletRhythmMaker(
353     tuplet_ratios=[(1, 3, 2)],
354     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
355         trivialize=True,
356         extract_trivial=True,
357         rewrite_rest_filled=True,
358     ),
359 )
360
361 attachment_handler_one = AttachmentHandler(
362     starting_dynamic='mf',
363     ending_dynamic='p',
364     hairpin_indicator='>',
365     articulation='tenuto',
366 )
367
368 attachment_handler_two = AttachmentHandler(
369     starting_dynamic='mp',
370     ending_dynamic='mf',
371     hairpin_indicator='<',
372     articulation='',
373 )
374
375 attachment_handler_three = AttachmentHandler(
376     starting_dynamic='pp',
377     ending_dynamic='ff',
378     hairpin_indicator='<',
379     articulation='',
380 )
381
382 ##### oboe #####
383 flutemusicmaker_one = MusicMaker(
384     rmaker=rmaker_one,
385     pitches=flute_scale,
386     continuous=True,
387     attachment_handler=attachment_handler_one,
388 )
389 flutemusicmaker_two = MusicMaker(
390     rmaker=rmaker_two,
391     pitches=flute_notes_two,
392     continuous=True,
393     attachment_handler=attachment_handler_two,
394 )
395 flutemusicmaker_three = MusicMaker(
396     rmaker=rmaker_three,
397     pitches=flute_notes_one,
398     continuous=True,
399     attachment_handler=attachment_handler_three,
400 )
401 ##### violin1 #####
402 violin1musicmaker_one = MusicMaker(
403     rmaker=rmaker_one,

```

```
404     pitches=violin1_scale,
405     continuous=True,
406     attachment_handler=attachment_handler_one,
407 )
408 violinimusicmaker_two = MusicMaker(
409     rmaker=rmaker_two,
410     pitches=violin1_notes_two,
411     continuous=True,
412     attachment_handler=attachment_handler_two,
413 )
414 violinimusicmaker_three = MusicMaker(
415     rmaker=rmaker_three,
416     pitches=violin1_notes_one,
417     continuous=True,
418     attachment_handler=attachment_handler_three,
419 )
420 #####trumpet#####
421 trumpetmusicmaker_one = MusicMaker(
422     rmaker=rmaker_one,
423     pitches=trumpet_scale,
424     continuous=True,
425     attachment_handler=attachment_handler_one,
426 )
427 trumpetmusicmaker_two = MusicMaker(
428     rmaker=rmaker_two,
429     pitches=trumpet_notes_two,
430     continuous=True,
431     attachment_handler=attachment_handler_two,
432 )
433 trumpetmusicmaker_three = MusicMaker(
434     rmaker=rmaker_three,
435     pitches=trumpet_notes_one,
436     continuous=True,
437     attachment_handler=attachment_handler_three,
438 )
439 #####clarinet#####
440 clarinetmusicmaker_one = MusicMaker(
441     rmaker=rmaker_one,
442     pitches=clarinet_scale,
443     continuous=True,
444     attachment_handler=attachment_handler_one,
445 )
446 clarinetmusicmaker_two = MusicMaker(
447     rmaker=rmaker_two,
448     pitches=clarinet_notes_two,
449     continuous=True,
450     attachment_handler=attachment_handler_two,
451 )
452 clarinetmusicmaker_three = MusicMaker(
453     rmaker=rmaker_three,
454     pitches=clarinet_notes_one,
455     continuous=True,
456     attachment_handler=attachment_handler_three,
457 )
```

```
458 #####violin2#####
459 violin2musicmaker_one = MusicMaker(
460     rmaker=rmaker_one,
461     pitches=violin2_scale,
462     continuous=True,
463     attachment_handler=attachment_handler_one,
464 )
465 violin2musicmaker_two = MusicMaker(
466     rmaker=rmaker_two,
467     pitches=violin2_notes_two,
468     continuous=True,
469     attachment_handler=attachment_handler_two,
470 )
471 violin2musicmaker_three = MusicMaker(
472     rmaker=rmaker_three,
473     pitches=violin2_notes_one,
474     continuous=True,
475     attachment_handler=attachment_handler_three,
476 )
477 #####viola#####
478 violamusicmaker_one = MusicMaker(
479     rmaker=rmaker_one,
480     pitches=viola_scale,
481     continuous=True,
482     attachment_handler=attachment_handler_one,
483 )
484 violamusicmaker_two = MusicMaker(
485     rmaker=rmaker_two,
486     pitches=viola_notes_two,
487     continuous=True,
488     attachment_handler=attachment_handler_two,
489 )
490 violamusicmaker_three = MusicMaker(
491     rmaker=rmaker_three,
492     pitches=viola_notes_one,
493     continuous=True,
494     attachment_handler=attachment_handler_three,
495 )
496 #####bassoon#####
497 bassoonmusicmaker_one = MusicMaker(
498     rmaker=rmaker_one,
499     pitches=bassoon_scale,
500     continuous=True,
501     attachment_handler=attachment_handler_one,
502 )
503 bassoonmusicmaker_two = MusicMaker(
504     rmaker=rmaker_two,
505     pitches=bassoon_notes_two,
506     continuous=True,
507     attachment_handler=attachment_handler_two,
508 )
509 bassoonmusicmaker_three = MusicMaker(
510     rmaker=rmaker_three,
511     pitches=bassoon_notes_one,
```

```

$12     continuous=True,
$13     attachment_handler=attachment_handler_three,
$14 )
$15 #####trombone#####
$16 trombonemusicmaker_one = MusicMaker(
$17     rmaker=rmaker_one,
$18     pitches=trombone_scale,
$19     continuous=True,
$20     attachment_handler=attachment_handler_one,
$21 )
$22 trombonemusicmaker_two = MusicMaker(
$23     rmaker=rmaker_two,
$24     pitches=trombone_notes_two,
$25     continuous=True,
$26     attachment_handler=attachment_handler_two,
$27 )
$28 trombonemusicmaker_three = MusicMaker(
$29     rmaker=rmaker_three,
$30     pitches=trombone_notes_one,
$31     continuous=True,
$32     attachment_handler=attachment_handler_three,
$33 )
$34 #####cello#####
$35 cellomusicmaker_one = MusicMaker(
$36     rmaker=rmaker_one,
$37     pitches=cello_scale,
$38     continuous=True,
$39     attachment_handler=attachment_handler_one,
$40 )
$41 cellomusicmaker_two = MusicMaker(
$42     rmaker=rmaker_two,
$43     pitches=cello_notes_two,
$44     continuous=True,
$45     attachment_handler=attachment_handler_two,
$46 )
$47 cellomusicmaker_three = MusicMaker(
$48     rmaker=rmaker_three,
$49     pitches=cello_notes_one,
$50     continuous=True,
$51     attachment_handler=attachment_handler_three,
$52 )
$53 #####horn#####
$54 hornmusicmaker_one = MusicMaker(
$55     rmaker=rmaker_one,
$56     pitches=horn_scale,
$57     continuous=True,
$58     attachment_handler=attachment_handler_one,
$59 )
$60 hornmusicmaker_two = MusicMaker(
$61     rmaker=rmaker_two,
$62     pitches=horn_notes_two,
$63     continuous=True,
$64     attachment_handler=attachment_handler_two,
$65 )

```

```

566 hornmusicmaker_three = MusicMaker(
567     rmaker=rmaker_three,
568     pitches=horn_notes_one,
569     continuous=True,
570     attachment_handler=attachment_handler_three,
571 )
572 #####tuba#####
573 tubamusicmaker_one = MusicMaker(
574     rmaker=rmaker_one,
575     pitches=tuba_scale,
576     continuous=True,
577     attachment_handler=attachment_handler_one,
578 )
579 tubamusicmaker_two = MusicMaker(
580     rmaker=rmaker_two,
581     pitches=tuba_notes_two,
582     continuous=True,
583     attachment_handler=attachment_handler_two,
584 )
585 tubamusicmaker_three = MusicMaker(
586     rmaker=rmaker_three,
587     pitches=tuba_notes_one,
588     continuous=True,
589     attachment_handler=attachment_handler_three,
590 )
591 #####bass#####
592 bassmusicmaker_one = MusicMaker(
593     rmaker=rmaker_one,
594     pitches=bass_scale,
595     continuous=True,
596     attachment_handler=attachment_handler_one,
597 )
598 bassmusicmaker_two = MusicMaker(
599     rmaker=rmaker_two,
600     pitches=bass_notes_two,
601     continuous=True,
602     attachment_handler=attachment_handler_two,
603 )
604 bassmusicmaker_three = MusicMaker(
605     rmaker=rmaker_three,
606     pitches=bass_notes_one,
607     continuous=True,
608     attachment_handler=attachment_handler_three,
609 )
610
611 silence_maker = abjadext.rmakers.NoteRhythmMaker(
612     division_masks=[
613         abjadext.rmakers.SilenceMask(
614             pattern=abjad.index([0], 1),
615             ),
616         ],
617     )
618
619 class MusicSpecifier:

```

```

620
621     def __init__(self, music_maker, voice_name):
622         self.music_maker = music_maker
623         self.voice_name = voice_name
624
625     print('Collecting timespans and rmakers ...')
626     #####group one#####
627     voice_1_timespan_list = abjad.TimespanList([
628         abjad.AnnotatedTimespan(
629             start_offset=start_offset,
630             stop_offset=stop_offset,
631             annotation=MusicSpecifier(
632                 music_maker=music_maker,
633                 voice_name='Voice 1',
634             ),
635         )
636         for start_offset, stop_offset, music_maker in [
637             [(9, 4), (10, 4), flutemusicmaker_one],
638             [(15, 4), (18, 4), flutemusicmaker_two],
639             [(22, 4), (25, 4), flutemusicmaker_three],
640             [(27, 4), (30, 4), flutemusicmaker_one],
641             [(30, 4), (32, 4), flutemusicmaker_one],
642             [(35, 4), (39, 4), flutemusicmaker_two],
643             [(42, 4), (43, 4), flutemusicmaker_three],
644             [(43, 4), (44, 4), flutemusicmaker_three],
645             [(45, 4), (46, 4), flutemusicmaker_one],
646             [(46, 4), (50, 4), flutemusicmaker_one],
647             [(54, 4), (57, 4), flutemusicmaker_two],
648             [(59, 4), (60, 4), flutemusicmaker_three],
649             [(65, 4), (67, 4), flutemusicmaker_one],
650             [(67, 4), (69, 4), flutemusicmaker_one],
651             [(70, 4), (72, 4), flutemusicmaker_two],
652             [(72, 4), (75, 4), flutemusicmaker_two],
653             [(76, 4), (78, 4), flutemusicmaker_three],
654             [(81, 4), (82, 4), flutemusicmaker_one],
655             [(82, 4), (85, 4), flutemusicmaker_one],
656             [(90, 4), (91, 4), flutemusicmaker_two],
657             [(93, 4), (94, 4), flutemusicmaker_three],
658             [(94, 4), (96, 4), flutemusicmaker_three],
659             [(100, 4), (104, 4), flutemusicmaker_one],
660             [(104, 4), (105, 4), flutemusicmaker_one],
661             [(106, 4), (107, 4), flutemusicmaker_two],
662             [(107, 4), (108, 4), flutemusicmaker_two],
663             [(111, 4), (114, 4), flutemusicmaker_one],
664             [(114, 4), (115, 4), flutemusicmaker_one],
665             [(116, 4), (119, 4), flutemusicmaker_one],
666             [(119, 4), (120, 4), flutemusicmaker_one],
667             [(121, 4), (123, 4), flutemusicmaker_one],
668             [(123, 4), (125, 4), flutemusicmaker_one],
669             [(126, 4), (131, 4), flutemusicmaker_two],
670             [(131, 4), (133, 4), flutemusicmaker_two],
671             [(136, 4), (141, 4), flutemusicmaker_two],
672             [(148, 4), (150, 4), flutemusicmaker_two],
673             [(150, 4), (153, 4), flutemusicmaker_three],

```

```

674     [(155, 4), (159, 4), flutemusicmaker_three],
675     [(162, 4), (164, 4), flutemusicmaker_three],
676     [(168, 4), (171, 4), flutemusicmaker_three],
677     [(173, 4), (175, 4), flutemusicmaker_three],
678     [(175, 4), (177, 4), flutemusicmaker_three],
679     [(180, 4), (182, 4), flutemusicmaker_three],
680     [(186, 4), (190, 4), flutemusicmaker_three],
681     [(190, 4), (191, 4), silence_maker],
682 ]
683 ])
684
685 voice_5_timespan_list = abjad.TimespanList([
686     abjad.AnnotatedTimespan(
687         start_offset=start_offset,
688         stop_offset=stop_offset,
689         annotation=MusicSpecifier(
690             music_maker=music_maker,
691             voice_name='Voice 5',
692         ),
693     )
694     for start_offset, stop_offset, music_maker in [
695         [(9, 4), (10, 4), trumpetmusicmaker_one],
696         [(14, 4), (18, 4), trumpetmusicmaker_two],
697         [(23, 4), (25, 4), trumpetmusicmaker_three],
698         [(27, 4), (30, 4), trumpetmusicmaker_one],
699         [(30, 4), (32, 4), trumpetmusicmaker_one],
700         [(35, 4), (39, 4), trumpetmusicmaker_two],
701         [(42, 4), (43, 4), trumpetmusicmaker_three],
702         [(43, 4), (44, 4), trumpetmusicmaker_three],
703         [(45, 4), (46, 4), trumpetmusicmaker_one],
704         [(46, 4), (50, 4), trumpetmusicmaker_one],
705         [(54, 4), (57, 4), trumpetmusicmaker_two],
706         [(59, 4), (60, 4), trumpetmusicmaker_three],
707         [(65, 4), (67, 4), trumpetmusicmaker_one],
708         [(67, 4), (69, 4), trumpetmusicmaker_one],
709         [(70, 4), (72, 4), trumpetmusicmaker_two],
710         [(72, 4), (75, 4), trumpetmusicmaker_two],
711         [(76, 4), (78, 4), trumpetmusicmaker_three],
712         [(81, 4), (82, 4), trumpetmusicmaker_one],
713         [(82, 4), (85, 4), trumpetmusicmaker_one],
714         [(90, 4), (91, 4), trumpetmusicmaker_two],
715         [(93, 4), (94, 4), trumpetmusicmaker_three],
716         [(94, 4), (96, 4), trumpetmusicmaker_three],
717         [(100, 4), (104, 4), trumpetmusicmaker_one],
718         [(104, 4), (105, 4), trumpetmusicmaker_one],
719         [(106, 4), (107, 4), trumpetmusicmaker_two],
720         [(107, 4), (108, 4), trumpetmusicmaker_two],
721         [(111, 4), (114, 4), trumpetmusicmaker_one],
722         [(114, 4), (115, 4), trumpetmusicmaker_one],
723         [(116, 4), (119, 4), trumpetmusicmaker_one],
724         [(119, 4), (120, 4), trumpetmusicmaker_one],
725         [(121, 4), (123, 4), trumpetmusicmaker_one],
726         [(123, 4), (125, 4), trumpetmusicmaker_one],
727         [(126, 4), (131, 4), trumpetmusicmaker_two],

```

```

728     [(131, 4), (133, 4), trumpetmusicmaker_two],
729     [(136, 4), (141, 4), trumpetmusicmaker_two],
730     [(148, 4), (150, 4), trumpetmusicmaker_two],
731     [(150, 4), (154, 4), trumpetmusicmaker_three],
732     [(157, 4), (159, 4), trumpetmusicmaker_three],
733     [(163, 4), (164, 4), trumpetmusicmaker_three],
734     [(164, 4), (166, 4), trumpetmusicmaker_three],
735     [(168, 4), (172, 4), trumpetmusicmaker_three],
736     [(175, 4), (177, 4), trumpetmusicmaker_three],
737     [(181, 4), (183, 4), trumpetmusicmaker_three],
738     [(183, 4), (184, 4), trumpetmusicmaker_three],
739     [(186, 4), (190, 4), trumpetmusicmaker_three],
740   ]
741 ])
742
743 voice_8_timestspan_list = abjad.TimespanList([
744     abjad.AnnotatedTimespan(
745         start_offset=start_offset,
746         stop_offset=stop_offset,
747         annotation=MusicSpecifier(
748             music_maker=music_maker,
749             voice_name='Voice 8',
750         ),
751     )
752     for start_offset, stop_offset, music_maker in [
753         [(9, 4), (10, 4), violin1musicmaker_one],
754         [(14, 4), (18, 4), violin1musicmaker_two],
755         [(22, 4), (25, 4), violin1musicmaker_three],
756         [(27, 4), (30, 4), violin1musicmaker_one],
757         [(35, 4), (39, 4), violin1musicmaker_two],
758         [(42, 4), (43, 4), violin1musicmaker_three],
759         [(43, 4), (44, 4), violin1musicmaker_three],
760         [(45, 4), (46, 4), violin1musicmaker_one],
761         [(46, 4), (50, 4), violin1musicmaker_one],
762         [(54, 4), (57, 4), violin1musicmaker_two],
763         [(59, 4), (60, 4), violin1musicmaker_three],
764         [(65, 4), (67, 4), violin1musicmaker_one],
765         [(67, 4), (69, 4), violin1musicmaker_one],
766         [(70, 4), (72, 4), violin1musicmaker_two],
767         [(72, 4), (75, 4), violin1musicmaker_two],
768         [(76, 4), (78, 4), violin1musicmaker_three],
769         [(81, 4), (82, 4), violin1musicmaker_one],
770         [(82, 4), (85, 4), violin1musicmaker_one],
771         [(90, 4), (91, 4), violin1musicmaker_two],
772         [(93, 4), (94, 4), violin1musicmaker_three],
773         [(94, 4), (96, 4), violin1musicmaker_three],
774         [(100, 4), (104, 4), violin1musicmaker_one],
775         [(104, 4), (105, 4), violin1musicmaker_one],
776         [(106, 4), (107, 4), violin1musicmaker_two],
777         [(107, 4), (108, 4), violin1musicmaker_two],
778         [(111, 4), (114, 4), violin1musicmaker_one],
779         [(114, 4), (115, 4), violin1musicmaker_one],
780         [(116, 4), (119, 4), violin1musicmaker_one],
781         [(119, 4), (120, 4), violin1musicmaker_one],

```

```

782     [(121, 4), (123, 4), violin1musicmaker_one],
783     [(123, 4), (125, 4), violin1musicmaker_one],
784     [(126, 4), (131, 4), violin1musicmaker_two],
785     [(131, 4), (133, 4), violin1musicmaker_two],
786     [(136, 4), (141, 4), violin1musicmaker_two],
787     [(148, 4), (150, 4), violin1musicmaker_two],
788     [(150, 4), (152, 4), violin1musicmaker_three],
789     [(156, 4), (159, 4), violin1musicmaker_three],
790     [(161, 4), (164, 4), violin1musicmaker_three],
791     [(164, 4), (165, 4), violin1musicmaker_three],
792     [(168, 4), (170, 4), violin1musicmaker_three],
793     [(174, 4), (175, 4), violin1musicmaker_three],
794     [(175, 4), (177, 4), violin1musicmaker_three],
795     [(179, 4), (183, 4), violin1musicmaker_three],
796     [(186, 4), (190, 4), violin1musicmaker_three],
797   ]
798 ])
799
800 ####group two#####
801 voice_2_timespan_list = abjad.TimespanList([
802     abjad.AnnotatedTimespan(
803         start_offset=start_offset,
804         stop_offset=stop_offset,
805         annotation=MusicSpecifier(
806             music_maker=music_maker,
807             voice_name='Voice 2',
808         ),
809     )
810     for start_offset, stop_offset, music_maker in [
811         [(2, 4), (5, 4), clarinetmusicmaker_one],
812         [(10, 4), (11, 4), clarinetmusicmaker_two],
813         [(11, 4), (13, 4), clarinetmusicmaker_two],
814         [(16, 4), (18, 4), clarinetmusicmaker_three],
815         [(21, 4), (22, 4), clarinetmusicmaker_one],
816         [(22, 4), (25, 4), clarinetmusicmaker_one],
817         [(35, 4), (40, 4), clarinetmusicmaker_one],
818         [(44, 4), (46, 4), clarinetmusicmaker_two],
819         [(46, 4), (47, 4), clarinetmusicmaker_two],
820         [(49, 4), (50, 4), clarinetmusicmaker_three],
821         [(55, 4), (59, 4), clarinetmusicmaker_one],
822         [(62, 4), (64, 4), clarinetmusicmaker_two],
823         [(65, 4), (67, 4), clarinetmusicmaker_three],
824         [(67, 4), (70, 4), clarinetmusicmaker_three],
825         [(70, 4), (71, 4), clarinetmusicmaker_three],
826         [(73, 4), (75, 4), clarinetmusicmaker_two],
827         [(75, 4), (76, 4), clarinetmusicmaker_two],
828         [(80, 4), (82, 4), clarinetmusicmaker_one],
829         [(82, 4), (85, 4), clarinetmusicmaker_one],
830         [(86, 4), (88, 4), clarinetmusicmaker_two],
831         [(91, 4), (94, 4), clarinetmusicmaker_three],
832         [(94, 4), (95, 4), clarinetmusicmaker_three],
833         [(100, 4), (101, 4), clarinetmusicmaker_two],
834         [(103, 4), (104, 4), clarinetmusicmaker_one],
835         [(104, 4), (106, 4), clarinetmusicmaker_one],

```

```

836     [(110, 4), (114, 4), clarinetmusicmaker_one],
837     [(115, 4), (119, 4), clarinetmusicmaker_one],
838     [(120, 4), (123, 4), clarinetmusicmaker_one],
839     [(123, 4), (124, 4), clarinetmusicmaker_one],
840     [(125, 4), (126, 4), clarinetmusicmaker_two],
841     [(129, 4), (131, 4), clarinetmusicmaker_two],
842     [(131, 4), (134, 4), clarinetmusicmaker_two],
843     [(141, 4), (144, 4), clarinetmusicmaker_two],
844     [(149, 4), (150, 4), clarinetmusicmaker_two],
845     [(155, 4), (159, 4), clarinetmusicmaker_three],
846     [(162, 4), (164, 4), clarinetmusicmaker_three],
847     [(165, 4), (168, 4), clarinetmusicmaker_three],
848     [(168, 4), (170, 4), clarinetmusicmaker_three],
849     [(174, 4), (175, 4), clarinetmusicmaker_three],
850     [(175, 4), (177, 4), clarinetmusicmaker_three],
851     [(179, 4), (180, 4), clarinetmusicmaker_three],
852     [(185, 4), (186, 4), clarinetmusicmaker_three],
853     [(186, 4), (190, 4), clarinetmusicmaker_three],
854 ]
855 ])
856
857 voice_9_timespan_list = abjad.TimespanList([
858     abjad.AnnotatedTimespan(
859         start_offset=start_offset,
860         stop_offset=stop_offset,
861         annotation=MusicSpecifier(
862             music_maker=music_maker,
863             voice_name='Voice 9',
864         ),
865     )
866     for start_offset, stop_offset, music_maker in [
867         [(2, 4), (5, 4), violin2musicmaker_one],
868         [(9, 4), (11, 4), violin2musicmaker_two],
869         [(11, 4), (13, 4), violin2musicmaker_two],
870         [(16, 4), (18, 4), violin2musicmaker_three],
871         [(21, 4), (22, 4), violin2musicmaker_one],
872         [(22, 4), (23, 4), violin2musicmaker_one],
873         [(35, 4), (40, 4), violin2musicmaker_one],
874         [(44, 4), (46, 4), violin2musicmaker_two],
875         [(46, 4), (47, 4), violin2musicmaker_two],
876         [(49, 4), (50, 4), violin2musicmaker_three],
877         [(55, 4), (59, 4), violin2musicmaker_one],
878         [(62, 4), (64, 4), violin2musicmaker_two],
879         [(65, 4), (67, 4), violin2musicmaker_three],
880         [(67, 4), (70, 4), violin2musicmaker_three],
881         [(70, 4), (71, 4), violin2musicmaker_three],
882         [(73, 4), (75, 4), violin2musicmaker_two],
883         [(75, 4), (76, 4), violin2musicmaker_two],
884         [(80, 4), (82, 4), violin2musicmaker_one],
885         [(82, 4), (85, 4), violin2musicmaker_one],
886         [(86, 4), (88, 4), violin2musicmaker_two],
887         [(91, 4), (94, 4), violin2musicmaker_three],
888         [(94, 4), (95, 4), violin2musicmaker_three],
889         [(100, 4), (101, 4), violin2musicmaker_two],

```

```

890     [(103, 4), (104, 4), violin2musicmaker_one],
891     [(104, 4), (106, 4), violin2musicmaker_one],
892     [(110, 4), (114, 4), violin2musicmaker_one],
893     [(115, 4), (119, 4), violin2musicmaker_one],
894     [(120, 4), (123, 4), violin2musicmaker_one],
895     [(123, 4), (124, 4), violin2musicmaker_one],
896     [(125, 4), (126, 4), violin2musicmaker_two],
897     [(129, 4), (131, 4), violin2musicmaker_two],
898     [(131, 4), (134, 4), violin2musicmaker_two],
899     [(141, 4), (144, 4), violin2musicmaker_two],
900     [(149, 4), (150, 4), violin2musicmaker_two],
901     [(154, 4), (157, 4), violin2musicmaker_three],
902     [(159, 4), (160, 4), violin2musicmaker_three],
903     [(165, 4), (168, 4), violin2musicmaker_three],
904     [(168, 4), (169, 4), violin2musicmaker_three],
905     [(172, 4), (174, 4), violin2musicmaker_three],
906     [(175, 4), (179, 4), violin2musicmaker_three],
907     [(179, 4), (180, 4), violin2musicmaker_three],
908     [(184, 4), (186, 4), violin2musicmaker_three],
909     [(186, 4), (190, 4), violin2musicmaker_three],
910 ]
911 ])
912
913 voice_10_timespan_list = abjad.TimespanList([
914     abjad.AnnotatedTimespan(
915         start_offset=start_offset,
916         stop_offset=stop_offset,
917         annotation=MusicSpecifier(
918             music_maker=music_maker,
919             voice_name='Voice 10',
920         ),
921     )
922     for start_offset, stop_offset, music_maker in [
923         [(2, 4), (5, 4), violamusicmaker_one],
924         [(9, 4), (11, 4), violamusicmaker_two],
925         [(11, 4), (13, 4), violamusicmaker_two],
926         [(17, 4), (18, 4), violamusicmaker_three],
927         [(21, 4), (22, 4), violamusicmaker_one],
928         [(22, 4), (25, 4), violamusicmaker_one],
929         [(29, 4), (30, 4), violamusicmaker_two],
930         [(30, 4), (32, 4), violamusicmaker_two],
931         [(35, 4), (40, 4), violamusicmaker_one],
932         [(44, 4), (46, 4), violamusicmaker_two],
933         [(46, 4), (47, 4), violamusicmaker_two],
934         [(49, 4), (50, 4), violamusicmaker_three],
935         [(55, 4), (59, 4), violamusicmaker_one],
936         [(62, 4), (64, 4), violamusicmaker_two],
937         [(65, 4), (67, 4), violamusicmaker_three],
938         [(67, 4), (70, 4), violamusicmaker_three],
939         [(70, 4), (71, 4), violamusicmaker_three],
940         [(73, 4), (75, 4), violamusicmaker_two],
941         [(75, 4), (76, 4), violamusicmaker_two],
942         [(80, 4), (82, 4), violamusicmaker_one],
943         [(82, 4), (85, 4), violamusicmaker_one],

```

```

944     [(86, 4), (88, 4), violamusicmaker_two],
945     [(91, 4), (94, 4), violamusicmaker_three],
946     [(94, 4), (95, 4), violamusicmaker_three],
947     [(100, 4), (101, 4), violamusicmaker_two],
948     [(103, 4), (104, 4), violamusicmaker_one],
949     [(104, 4), (106, 4), violamusicmaker_one],
950     [(110, 4), (114, 4), violamusicmaker_one],
951     [(115, 4), (119, 4), violamusicmaker_one],
952     [(120, 4), (123, 4), violamusicmaker_one],
953     [(123, 4), (124, 4), violamusicmaker_one],
954     [(125, 4), (126, 4), violamusicmaker_two],
955     [(129, 4), (131, 4), violamusicmaker_two],
956     [(131, 4), (134, 4), violamusicmaker_two],
957     [(141, 4), (144, 4), violamusicmaker_two],
958     [(149, 4), (150, 4), violamusicmaker_two],
959     [(153, 4), (154, 4), violamusicmaker_three],
960     [(154, 4), (155, 4), violamusicmaker_three],
961     [(156, 4), (159, 4), violamusicmaker_three],
962     [(159, 4), (161, 4), violamusicmaker_three],
963     [(165, 4), (168, 4), violamusicmaker_three],
964     [(170, 4), (171, 4), violamusicmaker_three],
965     [(176, 4), (179, 4), violamusicmaker_three],
966     [(179, 4), (180, 4), violamusicmaker_three],
967     [(183, 4), (185, 4), violamusicmaker_three],
968     [(186, 4), (190, 4), violamusicmaker_three],
969   ]
970 ])
971
972 ###group three###
973 voice_3_timestrap_list = abjad.TimespanList([
974     abjad.AnnotatedTimespan(
975         start_offset=start_offset,
976         stop_offset=stop_offset,
977         annotation=MusicSpecifier(
978             music_maker=music_maker,
979             voice_name='Voice 3',
980         ),
981     )
982     for start_offset, stop_offset, music_maker in [
983         [(7, 4), (11, 4), bassoonmusicmaker_one],
984         [(15, 4), (16, 4), bassoonmusicmaker_two],
985         [(19, 4), (22, 4), bassoonmusicmaker_three],
986         [(22, 4), (23, 4), bassoonmusicmaker_three],
987         [(27, 4), (30, 4), bassoonmusicmaker_one],
988         [(32, 4), (35, 4), bassoonmusicmaker_two],
989         [(35, 4), (36, 4), bassoonmusicmaker_three],
990         [(37, 4), (40, 4), bassoonmusicmaker_two],
991         [(40, 4), (42, 4), bassoonmusicmaker_two],
992         [(46, 4), (49, 4), bassoonmusicmaker_one],
993         [(51, 4), (52, 4), bassoonmusicmaker_three],
994         [(57, 4), (59, 4), bassoonmusicmaker_two],
995         [(59, 4), (61, 4), bassoonmusicmaker_two],
996         [(64, 4), (66, 4), bassoonmusicmaker_one],
997         [(67, 4), (70, 4), bassoonmusicmaker_three],

```

```

998     [(70, 4), (72, 4), bassoonmusicmaker_one],
999     [(72, 4), (73, 4), bassoonmusicmaker_one],
1000    [(77, 4), (79, 4), bassoonmusicmaker_two],
1001    [(79, 4), (82, 4), bassoonmusicmaker_two],
1002    [(83, 4), (85, 4), bassoonmusicmaker_three],
1003    [(88, 4), (89, 4), bassoonmusicmaker_two],
1004    [(89, 4), (92, 4), bassoonmusicmaker_two],
1005    [(97, 4), (98, 4), bassoonmusicmaker_one],
1006    [(100, 4), (103, 4), bassoonmusicmaker_two],
1007    [(107, 4), (110, 4), bassoonmusicmaker_three],
1008    [(110, 4), (112, 4), bassoonmusicmaker_one],
1009    [(113, 4), (114, 4), bassoonmusicmaker_one],
1010    [(114, 4), (117, 4), bassoonmusicmaker_one],
1011    [(118, 4), (119, 4), bassoonmusicmaker_one],
1012    [(119, 4), (122, 4), bassoonmusicmaker_one],
1013    [(123, 4), (125, 4), bassoonmusicmaker_one],
1014    [(126, 4), (131, 4), bassoonmusicmaker_two],
1015    [(138, 4), (141, 4), bassoonmusicmaker_two],
1016    [(146, 4), (150, 4), bassoonmusicmaker_two],
1017    [(150, 4), (154, 4), bassoonmusicmaker_three],
1018    [(154, 4), (155, 4), bassoonmusicmaker_three],
1019    [(159, 4), (162, 4), bassoonmusicmaker_three],
1020    [(164, 4), (165, 4), bassoonmusicmaker_three],
1021    [(170, 4), (172, 4), bassoonmusicmaker_three],
1022    [(172, 4), (174, 4), bassoonmusicmaker_three],
1023    [(177, 4), (179, 4), bassoonmusicmaker_three],
1024    [(180, 4), (183, 4), bassoonmusicmaker_three],
1025    [(183, 4), (185, 4), bassoonmusicmaker_three],
1026    [(186, 4), (190, 4), bassoonmusicmaker_three],
1027 ]
1028 ])
1029
1030 voice_6_timestspan_list = abjad.TimespanList([
1031     abjad.AnnotatedTimespan(
1032         start_offset=start_offset,
1033         stop_offset=stop_offset,
1034         annotation=MusicSpecifier(
1035             music_maker=music_maker,
1036             voice_name='Voice 6',
1037         ),
1038     )
1039     for start_offset, stop_offset, music_maker in [
1040         [(7, 4), (11, 4), trombonemusicmaker_one],
1041         [(14, 4), (16, 4), trombonemusicmaker_two],
1042         [(19, 4), (22, 4), trombonemusicmaker_three],
1043         [(22, 4), (23, 4), trombonemusicmaker_three],
1044         [(27, 4), (29, 4), trombonemusicmaker_one],
1045         [(35, 4), (36, 4), trombonemusicmaker_three],
1046         [(37, 4), (40, 4), trombonemusicmaker_two],
1047         [(40, 4), (42, 4), trombonemusicmaker_two],
1048         [(46, 4), (49, 4), trombonemusicmaker_one],
1049         [(51, 4), (52, 4), trombonemusicmaker_three],
1050         [(57, 4), (59, 4), trombonemusicmaker_two],
1051         [(59, 4), (61, 4), trombonemusicmaker_two],

```

```

1052     [(64, 4), (66, 4), trombonemusicmaker_one],
1053     [(67, 4), (70, 4), trombonemusicmaker_three],
1054     [(70, 4), (72, 4), trombonemusicmaker_one],
1055     [(72, 4), (73, 4), trombonemusicmaker_one],
1056     [(77, 4), (79, 4), trombonemusicmaker_two],
1057     [(79, 4), (82, 4), trombonemusicmaker_two],
1058     [(83, 4), (85, 4), trombonemusicmaker_three],
1059     [(88, 4), (89, 4), trombonemusicmaker_two],
1060     [(89, 4), (92, 4), trombonemusicmaker_two],
1061     [(97, 4), (98, 4), trombonemusicmaker_one],
1062     [(100, 4), (103, 4), trombonemusicmaker_two],
1063     [(107, 4), (110, 4), trombonemusicmaker_three],
1064     [(110, 4), (112, 4), trombonemusicmaker_one],
1065     [(113, 4), (114, 4), trombonemusicmaker_one],
1066     [(114, 4), (117, 4), trombonemusicmaker_one],
1067     [(118, 4), (119, 4), trombonemusicmaker_one],
1068     [(119, 4), (122, 4), trombonemusicmaker_one],
1069     [(123, 4), (125, 4), trombonemusicmaker_one],
1070     [(126, 4), (131, 4), trombonemusicmaker_two],
1071     [(138, 4), (141, 4), trombonemusicmaker_two],
1072     [(146, 4), (150, 4), trombonemusicmaker_two],
1073     [(150, 4), (154, 4), trombonemusicmaker_three],
1074     [(157, 4), (159, 4), trombonemusicmaker_three],
1075     [(160, 4), (164, 4), trombonemusicmaker_three],
1076     [(164, 4), (165, 4), trombonemusicmaker_three],
1077     [(169, 4), (172, 4), trombonemusicmaker_three],
1078     [(174, 4), (175, 4), trombonemusicmaker_three],
1079     [(180, 4), (183, 4), trombonemusicmaker_three],
1080     [(183, 4), (184, 4), trombonemusicmaker_three],
1081     [(186, 4), (190, 4), trombonemusicmaker_three],
1082   ],
1083 ]
1084
1085 voice_11_timespan_list = abjad.TimespanList([
1086     abjad.AnnotatedTimespan(
1087         start_offset=start_offset,
1088         stop_offset=stop_offset,
1089         annotation=MusicSpecifier(
1090             music_maker=music_maker,
1091             voice_name='Voice 11',
1092         ),
1093     )
1094     for start_offset, stop_offset, music_maker in [
1095         [(7, 4), (11, 4), cellomusicmaker_one],
1096         [(14, 4), (16, 4), cellomusicmaker_two],
1097         [(21, 4), (22, 4), cellomusicmaker_three],
1098         [(22, 4), (23, 4), cellomusicmaker_three],
1099         [(27, 4), (30, 4), cellomusicmaker_one],
1100         [(35, 4), (36, 4), cellomusicmaker_three],
1101         [(37, 4), (40, 4), cellomusicmaker_two],
1102         [(40, 4), (42, 4), cellomusicmaker_two],
1103         [(46, 4), (49, 4), cellomusicmaker_one],
1104         [(51, 4), (52, 4), cellomusicmaker_three],
1105         [(57, 4), (59, 4), cellomusicmaker_two],

```

```

1106     [(59, 4), (61, 4), cellomusicmaker_two],
1107     [(64, 4), (66, 4), cellomusicmaker_one],
1108     [(67, 4), (70, 4), cellomusicmaker_three],
1109     [(70, 4), (72, 4), cellomusicmaker_one],
1110     [(72, 4), (73, 4), cellomusicmaker_one],
1111     [(77, 4), (79, 4), cellomusicmaker_two],
1112     [(79, 4), (82, 4), cellomusicmaker_two],
1113     [(83, 4), (85, 4), cellomusicmaker_three],
1114     [(88, 4), (89, 4), cellomusicmaker_two],
1115     [(89, 4), (92, 4), cellomusicmaker_two],
1116     [(97, 4), (98, 4), cellomusicmaker_one],
1117     [(100, 4), (103, 4), cellomusicmaker_two],
1118     [(107, 4), (110, 4), cellomusicmaker_three],
1119     [(110, 4), (112, 4), cellomusicmaker_one],
1120     [(113, 4), (114, 4), cellomusicmaker_one],
1121     [(114, 4), (117, 4), cellomusicmaker_one],
1122     [(118, 4), (119, 4), cellomusicmaker_one],
1123     [(119, 4), (122, 4), cellomusicmaker_one],
1124     [(123, 4), (125, 4), cellomusicmaker_one],
1125     [(126, 4), (131, 4), cellomusicmaker_two],
1126     [(138, 4), (141, 4), cellomusicmaker_two],
1127     [(146, 4), (150, 4), cellomusicmaker_two],
1128     [(150, 4), (153, 4), cellomusicmaker_three],
1129     [(155, 4), (156, 4), cellomusicmaker_three],
1130     [(161, 4), (164, 4), cellomusicmaker_three],
1131     [(164, 4), (165, 4), cellomusicmaker_three],
1132     [(168, 4), (170, 4), cellomusicmaker_three],
1133     [(171, 4), (172, 4), cellomusicmaker_three],
1134     [(172, 4), (175, 4), cellomusicmaker_three],
1135     [(175, 4), (176, 4), cellomusicmaker_three],
1136     [(180, 4), (183, 4), cellomusicmaker_three],
1137     [(185, 4), (186, 4), cellomusicmaker_three],
1138     [(186, 4), (190, 4), cellomusicmaker_three],
1139 ]
1140 ])
1141
1142 ##group four##
1143 voice_4_timespan_list = abjad.TimespanList([
1144     abjad.AnnotatedTimespan(
1145         start_offset=start_offset,
1146         stop_offset=stop_offset,
1147         annotation=MusicSpecifier(
1148             music_maker=music_maker,
1149             voice_name='Voice 4',
1150         ),
1151     )
1152     for start_offset, stop_offset, music_maker in [
1153         [(0, 4), (5, 4), hornmusicmaker_one],
1154         [(8, 4), (10, 4), hornmusicmaker_two],
1155         [(14, 4), (18, 4), hornmusicmaker_three],
1156         [(21, 4), (22, 4), hornmusicmaker_one],
1157         [(22, 4), (23, 4), hornmusicmaker_one],
1158         [(38, 4), (40, 4), hornmusicmaker_two],
1159         [(41, 4), (43, 4), hornmusicmaker_one],

```

```

1160     [(43, 4), (46, 4), hornmusicmaker_one],
1161     [(50, 4), (53, 4), hornmusicmaker_three],
1162     [(55, 4), (56, 4), hornmusicmaker_two],
1163     [(61, 4), (64, 4), hornmusicmaker_one],
1164     [(64, 4), (65, 4), hornmusicmaker_one],
1165     [(68, 4), (70, 4), hornmusicmaker_three],
1166     [(70, 4), (72, 4), hornmusicmaker_two],
1167     [(72, 4), (74, 4), hornmusicmaker_two],
1168     [(79, 4), (80, 4), hornmusicmaker_three],
1169     [(82, 4), (85, 4), hornmusicmaker_two],
1170     [(89, 4), (94, 4), hornmusicmaker_one],
1171     [(95, 4), (97, 4), hornmusicmaker_two],
1172     [(100, 4), (104, 4), hornmusicmaker_three],
1173     [(109, 4), (110, 4), hornmusicmaker_two],
1174     [(110, 4), (111, 4), hornmusicmaker_one],
1175     [(112, 4), (114, 4), hornmusicmaker_one],
1176     [(114, 4), (116, 4), hornmusicmaker_one],
1177     [(117, 4), (119, 4), hornmusicmaker_one],
1178     [(119, 4), (121, 4), hornmusicmaker_one],
1179     [(122, 4), (123, 4), hornmusicmaker_one],
1180     [(123, 4), (125, 4), hornmusicmaker_one],
1181     [(133, 4), (136, 4), hornmusicmaker_two],
1182     [(142, 4), (146, 4), hornmusicmaker_two],
1183     [(146, 4), (150, 4), hornmusicmaker_two],
1184     [(153, 4), (154, 4), hornmusicmaker_three],
1185     [(154, 4), (155, 4), hornmusicmaker_three],
1186     [(159, 4), (162, 4), hornmusicmaker_three],
1187     [(164, 4), (168, 4), hornmusicmaker_three],
1188     [(171, 4), (172, 4), hornmusicmaker_three],
1189     [(172, 4), (173, 4), hornmusicmaker_three],
1190     [(177, 4), (179, 4), hornmusicmaker_three],
1191     [(179, 4), (180, 4), hornmusicmaker_three],
1192     [(182, 4), (183, 4), hornmusicmaker_three],
1193     [(183, 4), (186, 4), hornmusicmaker_three],
1194     [(186, 4), (190, 4), hornmusicmaker_three],
1195 ]
1196 ])
1197
1198 voice_7_timespan_list = abjad.TimespanList([
1199     abjad.AnnotatedTimespan(
200         start_offset=start_offset,
201         stop_offset=stop_offset,
202         annotation=MusicSpecifier(
203             music_maker=music_maker,
204             voice_name='Voice 7',
205         ),
206     )
207     for start_offset, stop_offset, music_maker in [
208         [(0, 4), (5, 4), tubamusicmaker_one],
209         [(8, 4), (10, 4), tubamusicmaker_two],
210         [(14, 4), (18, 4), tubamusicmaker_three],
211         [(21, 4), (22, 4), tubamusicmaker_one],
212         [(22, 4), (23, 4), tubamusicmaker_one],
213         [(26, 4), (30, 4), tubamusicmaker_two],

```

```

1214     [(38, 4), (40, 4), tubamusicmaker_two],
1215     [(41, 4), (43, 4), tubamusicmaker_one],
1216     [(43, 4), (46, 4), tubamusicmaker_one],
1217     [(50, 4), (53, 4), tubamusicmaker_three],
1218     [(55, 4), (56, 4), tubamusicmaker_two],
1219     [(61, 4), (64, 4), tubamusicmaker_one],
1220     [(64, 4), (65, 4), tubamusicmaker_one],
1221     [(68, 4), (70, 4), tubamusicmaker_three],
1222     [(70, 4), (72, 4), tubamusicmaker_two],
1223     [(72, 4), (74, 4), tubamusicmaker_two],
1224     [(79, 4), (80, 4), tubamusicmaker_three],
1225     [(82, 4), (85, 4), tubamusicmaker_two],
1226     [(89, 4), (94, 4), tubamusicmaker_one],
1227     [(95, 4), (97, 4), tubamusicmaker_two],
1228     [(100, 4), (104, 4), tubamusicmaker_three],
1229     [(109, 4), (110, 4), tubamusicmaker_two],
1230     [(110, 4), (111, 4), tubamusicmaker_one],
1231     [(112, 4), (114, 4), tubamusicmaker_one],
1232     [(114, 4), (116, 4), tubamusicmaker_one],
1233     [(117, 4), (119, 4), tubamusicmaker_one],
1234     [(119, 4), (121, 4), tubamusicmaker_one],
1235     [(122, 4), (123, 4), tubamusicmaker_one],
1236     [(123, 4), (125, 4), tubamusicmaker_one],
1237     [(133, 4), (136, 4), tubamusicmaker_two],
1238     [(142, 4), (146, 4), tubamusicmaker_two],
1239     [(146, 4), (150, 4), tubamusicmaker_two],
1240     [(154, 4), (157, 4), tubamusicmaker_three],
1241     [(159, 4), (163, 4), tubamusicmaker_three],
1242     [(166, 4), (168, 4), tubamusicmaker_three],
1243     [(172, 4), (175, 4), tubamusicmaker_three],
1244     [(177, 4), (179, 4), tubamusicmaker_three],
1245     [(179, 4), (181, 4), tubamusicmaker_three],
1246     [(184, 4), (186, 4), tubamusicmaker_three],
1247     [(186, 4), (190, 4), tubamusicmaker_three],
1248 ]
1249 ])
1250
1251 voice_12_timespan_list = abjad.TimespanList([
1252     abjad.AnnotatedTimespan(
1253         start_offset=start_offset,
1254         stop_offset=stop_offset,
1255         annotation=MusicSpecifier(
1256             music_maker=music_maker,
1257             voice_name='Voice 12',
1258         ),
1259     )
1260     for start_offset, stop_offset, music_maker in [
1261         [(0, 4), (5, 4), bassmusicmaker_one],
1262         [(8, 4), (10, 4), bassmusicmaker_two],
1263         [(14, 4), (18, 4), bassmusicmaker_three],
1264         [(21, 4), (22, 4), bassmusicmaker_one],
1265         [(22, 4), (23, 4), bassmusicmaker_one],
1266         [(38, 4), (40, 4), bassmusicmaker_two],
1267         [(41, 4), (43, 4), bassmusicmaker_one],

```

```

1268     [(43, 4), (46, 4), bassmusicmaker_one],
1269     [(50, 4), (53, 4), bassmusicmaker_three],
1270     [(55, 4), (56, 4), bassmusicmaker_two],
1271     [(61, 4), (64, 4), bassmusicmaker_one],
1272     [(64, 4), (65, 4), bassmusicmaker_one],
1273     [(68, 4), (70, 4), bassmusicmaker_three],
1274     [(70, 4), (72, 4), bassmusicmaker_two],
1275     [(72, 4), (74, 4), bassmusicmaker_two],
1276     [(79, 4), (80, 4), bassmusicmaker_three],
1277     [(82, 4), (85, 4), bassmusicmaker_two],
1278     [(89, 4), (94, 4), bassmusicmaker_one],
1279     [(95, 4), (97, 4), bassmusicmaker_two],
1280     [(100, 4), (104, 4), bassmusicmaker_three],
1281     [(109, 4), (110, 4), bassmusicmaker_two],
1282     [(110, 4), (111, 4), bassmusicmaker_one],
1283     [(112, 4), (114, 4), bassmusicmaker_one],
1284     [(114, 4), (116, 4), bassmusicmaker_one],
1285     [(117, 4), (119, 4), bassmusicmaker_one],
1286     [(119, 4), (121, 4), bassmusicmaker_one],
1287     [(122, 4), (123, 4), bassmusicmaker_one],
1288     [(123, 4), (125, 4), bassmusicmaker_one],
1289     [(133, 4), (136, 4), bassmusicmaker_two],
1290     [(142, 4), (146, 4), bassmusicmaker_two],
1291     [(146, 4), (150, 4), bassmusicmaker_two],
1292     [(152, 4), (154, 4), bassmusicmaker_three],
1293     [(154, 4), (156, 4), bassmusicmaker_three],
1294     [(159, 4), (161, 4), bassmusicmaker_three],
1295     [(165, 4), (168, 4), bassmusicmaker_three],
1296     [(170, 4), (172, 4), bassmusicmaker_three],
1297     [(172, 4), (174, 4), bassmusicmaker_three],
1298     [(177, 4), (179, 4), bassmusicmaker_three],
1299     [(183, 4), (186, 4), bassmusicmaker_three],
1300     [(186, 4), (190, 4), bassmusicmaker_three],
1301   ]
1302 ])
1303
1304 all_timespan_lists = {
1305   'Voice 1': voice_1_timespan_list,
1306   'Voice 2': voice_2_timespan_list,
1307   'Voice 3': voice_3_timespan_list,
1308   'Voice 4': voice_4_timespan_list,
1309   'Voice 5': voice_5_timespan_list,
1310   'Voice 6': voice_6_timespan_list,
1311   'Voice 7': voice_7_timespan_list,
1312   'Voice 8': voice_8_timespan_list,
1313   'Voice 9': voice_9_timespan_list,
1314   'Voice 10': voice_10_timespan_list,
1315   'Voice 11': voice_11_timespan_list,
1316   'Voice 12': voice_12_timespan_list,
1317 }
1318
1319 global_timespan = abjad.Timespan(
1320   start_offset=0,
1321   stop_offset=max(_.stop_offset for _ in all_timespan_lists.values()))

```



```

1367     ],
1368     name='Staff Group 2',
1369 ),
1370 abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 3')
,
1371 abjad.StaffGroup(
1372 [
1373     abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1374 lilypond_type='Staff',),
1375     abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1376 lilypond_type='Staff',),
1377     abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1378 lilypond_type='Staff',),
1379     abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1380 lilypond_type='Staff',),
1381     abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1382 lilypond_type='Staff',),
1383 ],
1384     name='Staff Group 3',
1385 )
1386 ]
,
1387 )
1388
1389 for time_signature in time_signatures:
1390     skip = abjad.Skip(1, multiplier=(time_signature))
1391     abjad.attach(time_signature, skip)
1392     score['Global Context 1'].append(skip)
1393
1394 for time_signature in time_signatures:
1395     skip = abjad.Skip(1, multiplier=(time_signature))
1396     abjad.attach(time_signature, skip)
1397     score['Global Context 2'].append(skip)
1398
1399 for time_signature in time_signatures:
1400     skip = abjad.Skip(1, multiplier=(time_signature))
1401     abjad.attach(time_signature, skip)
1402     score['Global Context 3'].append(skip)
1403
1404 print('Making containers ...')
1405
1406 def make_container(music_maker, durations):
1407     selections = music_maker(durations)
1408     container = abjad.Container([])
1409     container.extend(selections)
1410     return container
1411
1412 def key_function(timespan):
1413     return timespan.annotation.music_maker or silence_maker
1414
1415 for voice_name, timespan_list in all_timespan_lists.items():
1416     for music_maker, grouper in itertools.groupby(
1417         timespan_list,
1418         key=key_function,
1419     ):
1420

```

```

1415     durations = [timespan.duration for timespan in grouper]
1416     container = make_container(music_maker, durations)
1417     voice = score[voice_name]
1418     voice.append(container)
1419
1420 print('Splitting and rewriting ...')
1421
1422 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
1423     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1424         time_signature = time_signatures[i]
1425         abjad.mutate(shard).rewrite_meter(time_signature)
1426
1427 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
1428     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1429         time_signature = time_signatures[i]
1430         abjad.mutate(shard).rewrite_meter(time_signature)
1431
1432 for voice in abjad.iterate(score['Staff Group 3']).components(abjad.Voice):
1433     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1434         time_signature = time_signatures[i]
1435         abjad.mutate(shard).rewrite_meter(time_signature)
1436
1437 print('Beaming runs ...')
1438
1439 for voice in abjad.select(score).components(abjad.Voice):
1440     for run in abjad.select(voice).runs():
1441         if 1 < len(run):
1442             specifier = abjadext.rmakers.BeamSpecifier(
1443                 beam_each_division=False,
1444                 )
1445             specifier(run)
1446             abjad.attach(abjad.StartBeam(), run[0])
1447             abjad.attach(abjad.StopBeam(), run[-1])
1448             for leaf in run:
1449                 if abjad.Duration(1, 4) <= leaf.written_duration:
1450                     continue
1451                 previous_leaf = abjad.inspect(leaf).leaf(-1)
1452                 next_leaf = abjad.inspect(leaf).leaf(1)
1453                 if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
1454                     abjad.Duration(1, 4) <= next_leaf.written_duration):
1455                     left = previous_leaf.written_duration.flag_count
1456                     right = leaf.written_duration.flag_count
1457                     beam_count = abjad.BeamCount(
1458                         left=left,
1459                         right=right,
1460                         )
1461                     abjad.attach(beam_count, leaf)
1462                     continue
1463                 if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
1464                     abjad.Duration(1, 4) <= previous_leaf.written_duration):
1465                     left = leaf.written_duration.flag_count
1466                     right = next_leaf.written_duration.flag_count
1467                     beam_count = abjad.BeamCount(
1468                         left=left,

```

```

1469             right=right,
1470         )
1471     abjad.attach(beam_count, leaf)
1472
1473 print('Beautifying score ...')
1474 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1475     for selection in abjad.select(staff).components(abjad.Rest).
1476         group_by_contiguity():
1477             start_command = abjad.LilyPondLiteral(
1478                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1479                 startStaff',
1480                 format_slot='before',
1481             )
1482             stop_command = abjad.LilyPondLiteral(
1483                 r'\stopStaff \startStaff',
1484                 format_slot='after',
1485             )
1486             abjad.attach(start_command, selection[0])
1487             abjad.attach(stop_command, selection[-1])
1488
1489 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1490     for selection in abjad.select(staff).components(abjad.Rest).
1491         group_by_contiguity():
1492             start_command = abjad.LilyPondLiteral(
1493                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1494                 startStaff',
1495                 format_slot='before',
1496             )
1497             stop_command = abjad.LilyPondLiteral(
1498                 r'\stopStaff \startStaff',
1499                 format_slot='after',
1500             )
1501             abjad.attach(start_command, selection[0])
1502             abjad.attach(stop_command, selection[-1])
1503
1504 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1505     for selection in abjad.select(staff).components(abjad.Rest).
1506         group_by_contiguity():
1507             start_command = abjad.LilyPondLiteral(
1508                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1509                 startStaff',
1510                 format_slot='before',
1511             )
1512             stop_command = abjad.LilyPondLiteral(
1513                 r'\stopStaff \startStaff',
1514                 format_slot='after',
1515             )
1516             abjad.attach(start_command, selection[0])
1517             abjad.attach(stop_command, selection[-1])
1518
1519 print('Stopping Hairpins ...')
1520 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1521     for rest in abjad.iterate(staff).components(abjad.Rest):
1522         previous_leaf = abjad.inspect(rest).leaf(-1)

```

```

1517     if isinstance(previous_leaf, abjad.Note):
1518         abjad.attach(abjad.StopHairpin(), rest)
1519     elif isinstance(previous_leaf, abjad.Chord):
1520         abjad.attach(abjad.StopHairpin(), rest)
1521     elif isinstance(previous_leaf, abjad.Rest):
1522         pass
1523
1524 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1525     for rest in abjad.iterate(staff).components(abjad.Rest):
1526         previous_leaf = abjad.inspect(rest).leaf(-1)
1527         if isinstance(previous_leaf, abjad.Note):
1528             abjad.attach(abjad.StopHairpin(), rest)
1529         elif isinstance(previous_leaf, abjad.Chord):
1530             abjad.attach(abjad.StopHairpin(), rest)
1531         elif isinstance(previous_leaf, abjad.Rest):
1532             pass
1533
1534 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1535     for rest in abjad.iterate(staff).components(abjad.Rest):
1536         previous_leaf = abjad.inspect(rest).leaf(-1)
1537         if isinstance(previous_leaf, abjad.Note):
1538             abjad.attach(abjad.StopHairpin(), rest)
1539         elif isinstance(previous_leaf, abjad.Chord):
1540             abjad.attach(abjad.StopHairpin(), rest)
1541         elif isinstance(previous_leaf, abjad.Rest):
1542             pass
1543
1544 print('Adding pitch material ...')
1545 def cyc(lst):
1546     count = 0
1547     while True:
1548         yield lst[count%len(lst)]
1549         count += 1
1550
1551
1552 print('Adding attachments ...')
1553 bar_line = abjad.BarLine('|||')
1554 metro = abjad.MetronomeMark((1, 4), 90)
1555 markup1 = abjad.Markup(r'\bold { G }')
1556 markup2 = abjad.Markup(r'\bold { H }')
1557 markup3 = abjad.Markup(r'\bold { I }')
1558 markup4 = abjad.Markup(r'\bold { J }')
1559 markup5 = abjad.Markup(r'\bold { K }')
1560 markup6 = abjad.Markup(r'\bold { L }')
1561 mark1 = abjad.RehearsalMark(markup=markup1)
1562 mark2 = abjad.RehearsalMark(markup=markup2)
1563 mark3 = abjad.RehearsalMark(markup=markup3)
1564 mark4 = abjad.RehearsalMark(markup=markup4)
1565 mark5 = abjad.RehearsalMark(markup=markup5)
1566 mark6 = abjad.RehearsalMark(markup=markup6)
1567
1568 instruments1 = cyc([
1569     abjad.Flute(),
1570     abjad.ClarinetInBFlat(),

```

```

1571     abjad.Bassoon(),  

1572   ])  

1573  

1574 instruments2 = cyc([  

1575     abjad.FrenchHorn(),  

1576     abjad.Trumpet(),  

1577     abjad.TenorTrombone(),  

1578     abjad.Tuba(),  

1579   ])  

1580  

1581 instruments3 = cyc([  

1582     abjad.Violin(),  

1583     abjad.Violin(),  

1584     abjad.Viola(),  

1585     abjad.Cello(),  

1586     abjad.Contrabass(),  

1587   ])  

1588  

1589 clefs1 = cyc([  

1590     abjad.Clef('treble'),  

1591     abjad.Clef('treble'),  

1592     abjad.Clef('bass'),  

1593   ])  

1594  

1595 clefs2 = cyc([  

1596     abjad.Clef('treble'),  

1597     abjad.Clef('treble'),  

1598     abjad.Clef('bass'),  

1599     abjad.Clef('bass'),  

1600   ])  

1601  

1602 clefs3 = cyc([  

1603     abjad.Clef('treble'),  

1604     abjad.Clef('treble'),  

1605     abjad.Clef('alto'),  

1606     abjad.Clef('bass'),  

1607     abjad.Clef('bass'),  

1608   ])  

1609  

1610 abbreviations1 = cyc([  

1611     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),  

1612     abjad.MarginMarkup(markup=abjad.Markup('cl.'),),  

1613     abjad.MarginMarkup(markup=abjad.Markup('bssn.'),),  

1614   ])  

1615  

1616 abbreviations2 = cyc([  

1617     abjad.MarginMarkup(markup=abjad.Markup('hr.'),),  

1618     abjad.MarginMarkup(markup=abjad.Markup('trp.'),),  

1619     abjad.MarginMarkup(markup=abjad.Markup('trmb.'),),  

1620     abjad.MarginMarkup(markup=abjad.Markup('tb.'),),  

1621   ])  

1622  

1623 abbreviations3 = cyc([  

1624     abjad.MarginMarkup(markup=abjad.Markup('vln.I'),),

```

```

1625 abjad.MarginMarkup(markup=abjad.Markup('vln.II'),),
1626 abjad.MarginMarkup(markup=abjad.Markup('vla.'),),
1627 abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
1628 abjad.MarginMarkup(markup=abjad.Markup('cb.'),),
1629 ])
1630
1631 names1 = cyc([
1632     abjad.StartMarkup(markup=abjad.Markup('Flute'),),
1633     abjad.StartMarkup(markup=abjad.Markup('Clarinet'),),
1634     abjad.StartMarkup(markup=abjad.Markup('Bassoon'),),
1635 ])
1636
1637 names2 = cyc([
1638     abjad.StartMarkup(markup=abjad.Markup('Horn'),),
1639     abjad.StartMarkup(markup=abjad.Markup('Trumpet'),),
1640     abjad.StartMarkup(markup=abjad.Markup('Trombone'),),
1641     abjad.StartMarkup(markup=abjad.Markup('Tuba'),),
1642 ])
1643
1644 names3 = cyc([
1645     abjad.StartMarkup(markup=abjad.Markup('Violin I'),),
1646     abjad.StartMarkup(markup=abjad.Markup('Violin II'),),
1647     abjad.StartMarkup(markup=abjad.Markup('Viola'),),
1648     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
1649     abjad.StartMarkup(markup=abjad.Markup('Contrabass'),),
1650 ])
1651
1652 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1653     leaf1 = abjad.select(staff).leaves()[0]
1654     abjad.attach(next(instruments1), leaf1)
1655     abjad.attach(next(abbreviations1), leaf1)
1656     abjad.attach(next(names1), leaf1)
1657     abjad.attach(next(clefs1), leaf1)
1658
1659 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1660     leaf1 = abjad.select(staff).leaves()[0]
1661     abjad.attach(next(instruments2), leaf1)
1662     abjad.attach(next(abbreviations2), leaf1)
1663     abjad.attach(next(names2), leaf1)
1664     abjad.attach(next(clefs2), leaf1)
1665
1666 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1667     leaf1 = abjad.select(staff).leaves()[0]
1668     abjad.attach(next(instruments3), leaf1)
1669     abjad.attach(next(abbreviations3), leaf1)
1670     abjad.attach(next(names3), leaf1)
1671     abjad.attach(next(clefs3), leaf1)
1672
1673 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
1674     leaf1 = abjad.select(staff).leaves()[0]
1675     last_leaf = abjad.select(staff).leaves()[-1]
1676     abjad.attach(metro, leaf1)
1677     abjad.attach(bar_line, last_leaf)
1678

```

```

1679 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff)[0]:
1680     leaf1 = abjad.select(staff).leaves()[0]
1681     last_leaf = abjad.select(staff).leaves()[-1]
1682     abjad.attach(metro, leaf1)
1683     abjad.attach(bar_line, last_leaf)
1684
1685 for staff in abjad.select(score['Staff Group 3']).components(abjad.Staff)[0]:
1686     leaf1 = abjad.select(staff).leaves()[0]
1687     last_leaf = abjad.select(staff).leaves()[-1]
1688     abjad.attach(metro, leaf1)
1689     abjad.attach(bar_line, last_leaf)
1690
1691 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1692     leaf1 = abjad.select(staff).leaves()[7]
1693     abjad.attach(mark1, leaf1)
1694
1695 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1696     leaf1 = abjad.select(staff).leaves()[7]
1697     abjad.attach(mark1, leaf1)
1698
1699 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1700     leaf1 = abjad.select(staff).leaves()[7]
1701     abjad.attach(mark1, leaf1)
1702
1703 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1704     leaf2 = abjad.select(staff).leaves()[16]
1705     abjad.attach(mark2, leaf2)
1706
1707 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1708     leaf2 = abjad.select(staff).leaves()[16]
1709     abjad.attach(mark2, leaf2)
1710
1711 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1712     leaf2 = abjad.select(staff).leaves()[16]
1713     abjad.attach(mark2, leaf2)
1714
1715 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1716     leaf3 = abjad.select(staff).leaves()[22]
1717     abjad.attach(mark3, leaf3)
1718
1719 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1720     leaf3 = abjad.select(staff).leaves()[22]
1721     abjad.attach(mark3, leaf3)
1722
1723 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1724     leaf3 = abjad.select(staff).leaves()[22]
1725     abjad.attach(mark3, leaf3)
1726
1727 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1728     leaf4 = abjad.select(staff).leaves()[29]
1729     abjad.attach(mark4, leaf4)
1730
1731 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1732     leaf4 = abjad.select(staff).leaves()[29]

```

```

1733     abjad.attach(mark4, leaf4)
1734
1735 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1736     leaf4 = abjad.select(staff).leaves()[29]
1737     abjad.attach(mark4, leaf4)
1738
1739 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1740     leaf5 = abjad.select(staff).leaves()[34]
1741     abjad.attach(mark5, leaf5)
1742
1743 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1744     leaf5 = abjad.select(staff).leaves()[34]
1745     abjad.attach(mark5, leaf5)
1746
1747 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1748     leaf5 = abjad.select(staff).leaves()[34]
1749     abjad.attach(mark5, leaf5)
1750
1751 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1752     leaf6 = abjad.select(staff).leaves()[39]
1753     abjad.attach(mark6, leaf6)
1754
1755 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1756     leaf6 = abjad.select(staff).leaves()[39]
1757     abjad.attach(mark6, leaf6)
1758
1759 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1760     leaf6 = abjad.select(staff).leaves()[39]
1761     abjad.attach(mark6, leaf6)
1762
1763 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1764     abjad.Instrument.transpose_from_sounding_pitch(staff)
1765
1766 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1767     abjad.Instrument.transpose_from_sounding_pitch(staff)
1768
1769 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1770     abjad.Instrument.transpose_from_sounding_pitch(staff)
1771
1772 score_file = abjad.LilyPondFile.new(
1773     score,
1774     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
1775     _stylesheets/abjad.ily'],
1776 )
1777
1778 abjad.SegmentMaker.comment_measure_numbers(score)
1779 ######
1780 directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/Segment_II'
1781 pdf_path = f'{directory}/Segment_II.pdf'
1782 path = pathlib.Path('Segment_II.pdf')
1783 if path.exists():
1784     print(f'Removing {pdf_path} ...')
1785     path.unlink()

```

```
1786 time_1 = time.time()
1787 print(f'Persisting {pdf_path} ...')
1788 result = abjad.persist(score_file).as_pdf(pdf_path)
1789 print(result[0])
1790 print(result[1])
1791 print(result[2])
1792 success = result[3]
1793 if success is False:
1794     print('LilyPond failed!')
1795 time_2 = time.time()
1796 total_time = time_2 - time_1
1797 print(f'Total time: {total_time} seconds')
1798 if path.exists():
1799     print(f'Opening {pdf_path} ...')
1800     os.system(f'open {pdf_path}')
```

### **Code Example A.12: Tianshu Segment\_II**

### A.3.1.3 SEGMENT\_III

```
1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed

11
12 print('Interpreting file ...')

13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (5, 4), (2, 4), (4, 4), (3, 4), (4, 4), (4, 4),
17         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
18         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
19         (2, 4), (3, 4), (4, 4), (3, 4), (4, 4), (3, 4),
20         (5, 4), (3, 4), (3, 4), (4, 4), (3, 4), (3, 4),
21         (4, 4), (5, 4), (4, 4), (3, 4), (5, 4), (5, 4),
22         (5, 4), (5, 4), (4, 4), (4, 4), (5, 4), (5, 4),
23         (4, 4), (4, 4), (3, 4), (4, 4), (4, 4), (3, 4),
24         (5, 4),
25     ]
26 ]
27
28 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures])
29
30 def reduceMod3(rw):
31     return [(x % 4) for x in rw]
32
```

```

33 def reduceMod5(rw):
34     return [(x % 6) for x in rw]
35
36 def reduceMod7(rw):
37     return [(x % 8) for x in rw]
38
39 def reduceMod9(rw):
40     return [(x % 10) for x in rw]
41
42 def reduceMod11(rw):
43     return [(x % 12) for x in rw]
44
45 def reduceMod13(rw):
46     return [(x % 14) for x in rw]
47
48 seed(1)
49 flute_random_walk_one = []
50 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
51 for i in range(1, 1000):
52     movement = -1 if random() < 0.5 else 1
53     value = flute_random_walk_one[i-1] + movement
54     flute_random_walk_one.append(value)
55 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
56 flute_chord_one = [0, 6, 8, 15, 23, 15, 8, 6, ]
57 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
58     flute_random_walk_one)]
59
59 seed(2)
60 clarinet_random_walk_one = []
61 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
62 for i in range(1, 1000):
63     movement = -1 if random() < 0.5 else 1
64     value = clarinet_random_walk_one[i-1] + movement
65     clarinet_random_walk_one.append(value)
66 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
67 clarinet_chord_one = [-10, 0, 6, 8, 15, 23, 15, 8, 6, 0, ]
68 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod9(
69     clarinet_random_walk_one)]
70
70 seed(3)
71 bassoon_random_walk_one = []
72 bassoon_random_walk_one.append(-1 if random() < 0.5 else 1)
73 for i in range(1, 1000):
74     movement = -1 if random() < 0.5 else 1
75     value = bassoon_random_walk_one[i-1] + movement
76     bassoon_random_walk_one.append(value)
77 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
78 bassoon_chord_one = [-19, -15, -10, 0, -10, -15, ]
79 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod5(
80     bassoon_random_walk_one)]
81
81 seed(4)
82 horn_random_walk_one = []
83 horn_random_walk_one.append(-1 if random() < 0.5 else 1)

```

```

84 for i in range(1, 1000):
85     movement = -1 if random() < 0.5 else 1
86     value = horn_random_walk_one[i-1] + movement
87     horn_random_walk_one.append(value)
88 horn_random_walk_one = [abs(x) for x in horn_random_walk_one]
89 horn_chord_one = [-19, -15, -10, 0, 6, 0, -10, -15, ]
90 horn_notes_one = [horn_chord_one[x] for x in reduceMod7(horn_random_walk_one)]
91
92 seed(5)
93 trumpet_random_walk_one = []
94 trumpet_random_walk_one.append(-1 if random() < 0.5 else 1)
95 for i in range(1, 1000):
96     movement = -1 if random() < 0.5 else 1
97     value = trumpet_random_walk_one[i-1] + movement
98     trumpet_random_walk_one.append(value)
99 trumpet_random_walk_one = [abs(x) for x in trumpet_random_walk_one]
100 trumpet_chord_one = [0, 6, 8, 15, 8, 6, ]
101 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod5(
102     trumpet_random_walk_one)]
103
104 seed(6)
105 trombone_random_walk_one = []
106 trombone_random_walk_one.append(-1 if random() < 0.5 else 1)
107 for i in range(1, 1000):
108     movement = -1 if random() < 0.5 else 1
109     value = trombone_random_walk_one[i-1] + movement
110     trombone_random_walk_one.append(value)
111 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
112 trombone_chord_one = [-19, -15, -10, 0, -10, -15, ]
113 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod5(
114     trombone_random_walk_one)]
115
116 seed(7)
117 tuba_random_walk_one = []
118 tuba_random_walk_one.append(-1 if random() < 0.5 else 1)
119 for i in range(1, 1000):
120     movement = -1 if random() < 0.5 else 1
121     value = tuba_random_walk_one[i-1] + movement
122     tuba_random_walk_one.append(value)
123 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
124 tuba_chord_one = [-29, -20, -19, -15, -10, -15, -19, -20, ]
125 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod7(tuba_random_walk_one)]
126
127 seed(8)
128 violin1_random_walk_one = []
129 violin1_random_walk_one.append(-1 if random() < 0.5 else 1)
130 for i in range(1, 1000):
131     movement = -1 if random() < 0.5 else 1
132     value = violin1_random_walk_one[i-1] + movement
133     violin1_random_walk_one.append(value)
134 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
135 violin1_chord_one = [0, 6, 8, 15, 23, 34, 37, 34, 23, 15, 8, 6, ]
136 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod11(
137     violin1_random_walk_one)]

```

```

135
136 seed(9)
137 violin2_random_walk_one = []
138 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
139 for i in range(1, 1000):
140     movement = -1 if random() < 0.5 else 1
141     value = violin2_random_walk_one[i-1] + movement
142     violin2_random_walk_one.append(value)
143 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
144 violin2_chord_one = [0, 6, 8, 15, 23, 15, 8, 6, ]
145 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod7(
146     violin2_random_walk_one)]
147
148 seed(10)
149 viola_random_walk_one = []
150 viola_random_walk_one.append(-1 if random() < 0.5 else 1)
151 for i in range(1, 1000):
152     movement = -1 if random() < 0.5 else 1
153     value = viola_random_walk_one[i-1] + movement
154     viola_random_walk_one.append(value)
155 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
156 viola_chord_one = [-10, 0, 6, 8, 15, 8, 6, 0, ]
157 viola_notes_one = [viola_chord_one[x] for x in reduceMod7(
158     viola_random_walk_one)]
159
160 seed(11)
161 cello_random_walk_one = []
162 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
163 for i in range(1, 1000):
164     movement = -1 if random() < 0.5 else 1
165     value = cello_random_walk_one[i-1] + movement
166     cello_random_walk_one.append(value)
167 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
168 cello_chord_one = [-20, -19, -15, -10, 0, 6, 8, 15, 8, 6, 0, -10, -15, -19, ]
169 cello_notes_one = [cello_chord_one[x] for x in reduceMod13(
170     cello_random_walk_one)]
171
172 seed(12)
173 bass_random_walk_one = []
174 bass_random_walk_one.append(-1 if random() < 0.5 else 1)
175 for i in range(1, 1000):
176     movement = -1 if random() < 0.5 else 1
177     value = bass_random_walk_one[i-1] + movement
178     bass_random_walk_one.append(value)
179 bass_random_walk_one = [abs(x) for x in bass_random_walk_one]
180 bass_chord_one = [-29, -20, -19, -15, -10, 0, -10, -15, -19, -20, ]
181 bass_notes_one = [bass_chord_one[x] for x in reduceMod9(bass_random_walk_one)]
182
183 flute_scale = [37, ]
184 clarinet_scale = [8, ]
185 bassoon_scale = [8, ]
186 horn_scale = [8, ]
187 trumpet_scale = [0, ]
188 trombone_scale = [-10, ]

```

```

186 tuba_scale = [-19, ]
187 violin1_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25, 24.5,
188   24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5, 20, 20.5, 21,
189   21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26, 26.5, 27, 27.5, 28,
190   28.5, 29, 29.5, ]
191 violin2_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14, 13.5,
192   13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5, 10, 10.5,
193   11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16, 16.5, 17, 17.5, 18,
194   18.5, ]
195 viola_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1, 0.5,
196   0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1,
197   1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]
198 cello_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8, -8.5, -9,
199   -9.5, -10, -10.5, -11, -11.5, -12, -12.5, -13, -13.5, -14, -13.5, -13,
200   -12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8, -7.5, -7, -6.5,
201   -6, -5.5, -5, -4.5, -4, -3.5, ]
202 bass_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18, -18.5, -19,
203   -19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5, -24, -24.5, -25,
204   -24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21, -20.5, -20, -19.5, -19,
205   -18.5, -18, -17.5, -17, -16.5, -16, -15.5, -15, -14.5, ]
206
207 seed(1)
208 flute_random_walk_two = []
209 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
210 for i in range(1, 1000):
211     movement = -1 if random() < 0.5 else 1
212     value = flute_random_walk_two[i-1] + movement
213     flute_random_walk_two.append(value)
214 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
215 flute_chord_two = [2, 8, 11, 22, 30, 22, 11, 8, ]
216 flute_notes_two = [flute_chord_two[x] for x in reduceMod7(
217     flute_random_walk_two)]
218
219 seed(2)
220 clarinet_random_walk_two = []
221 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
222 for i in range(1, 1000):
223     movement = -1 if random() < 0.5 else 1
224     value = clarinet_random_walk_two[i-1] + movement
225     clarinet_random_walk_two.append(value)
226 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
227 clarinet_chord_two = [-8, -7, 2, 8, 11, 22, 11, 8, 2, -7, ]
228 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod9(
229     clarinet_random_walk_two)]
230
231 seed(3)
232 bassoon_random_walk_two = []
233 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
234 for i in range(1, 1000):
235     movement = -1 if random() < 0.5 else 1
236     value = bassoon_random_walk_two[i-1] + movement
237     bassoon_random_walk_two.append(value)
238 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
239 bassoon_chord_two = [-17, -12, -8, -7, 2, -7, -8, -12, ]

```

```

224 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod7(
225     bassoon_random_walk_two)]
226
226 seed(4)
227 horn_random_walk_two = []
228 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
229 for i in range(1, 1000):
230     movement = -1 if random() < 0.5 else 1
231     value = horn_random_walk_two[i-1] + movement
232     horn_random_walk_two.append(value)
233 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
234 horn_chord_two = [-17, -12, -8, -7, 2, -7, -8, -12, ]
235 horn_notes_two = [horn_chord_two[x] for x in reduceMod7(horn_random_walk_two)]
236
236 seed(5)
237 trumpet_random_walk_two = []
238 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)
239 for i in range(1, 1000):
240     movement = -1 if random() < 0.5 else 1
241     value = trumpet_random_walk_two[i-1] + movement
242     trumpet_random_walk_two.append(value)
243 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]
244 trumpet_chord_two = [2, 8, 11, 8, ]
245 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod3(
246     trumpet_random_walk_two)]
247
248 seed(6)
249 trombone_random_walk_two = []
250 trombone_random_walk_two.append(-1 if random() < 0.5 else 1)
251 for i in range(1, 1000):
252     movement = -1 if random() < 0.5 else 1
253     value = trombone_random_walk_two[i-1] + movement
254     trombone_random_walk_two.append(value)
255 trombone_random_walk_two = [abs(x) for x in trombone_random_walk_two]
256 trombone_chord_two = [-17, -12, -8, -7, 2, -7, -8, -12, ]
257 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod7(
258     trombone_random_walk_two)]
259
259 seed(7)
260 tuba_random_walk_two = []
261 tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
262 for i in range(1, 1000):
263     movement = -1 if random() < 0.5 else 1
264     value = tuba_random_walk_two[i-1] + movement
265     tuba_random_walk_two.append(value)
266 tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
267 tuba_chord_two = [-27, -17, -12, -8, -7, -8, -12, -17, ]
268 tuba_notes_two = [tuba_chord_two[x] for x in reduceMod7(tuba_random_walk_two)]
269
270 seed(8)
271 violin1_random_walk_two = []
272 violin1_random_walk_two.append(-1 if random() < 0.5 else 1)
273 for i in range(1, 1000):
274     movement = -1 if random() < 0.5 else 1

```

```

275     value = violin1_random_walk_two[i-1] + movement
276     violin1_random_walk_two.append(value)
277 violin1_random_walk_two = [abs(x) for x in violin1_random_walk_two]
278 violin1_chord_two = [2, 8, 11, 22, 30, 37, 39, 37, 30, 22, 11, 8, ]
279 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod11(
280     violin1_random_walk_two)]
281
282 seed(9)
283 violin2_random_walk_two = []
284 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
285 for i in range(1, 1000):
286     movement = -1 if random() < 0.5 else 1
287     value = violin2_random_walk_two[i-1] + movement
288     violin2_random_walk_two.append(value)
289 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
290 violin2_chord_two = [2, 8, 11, 22, 30, 22, 11, 8, ]
291 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod7(
292     violin2_random_walk_two)]
293
294 seed(10)
295 viola_random_walk_two = []
296 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
297 for i in range(1, 1000):
298     movement = -1 if random() < 0.5 else 1
299     value = viola_random_walk_two[i-1] + movement
300     viola_random_walk_two.append(value)
301 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
302 viola_chord_two = [-12, -8, -7, 2, 8, 11, 22, 11, 8, 2, -7, -8, ]
303 viola_notes_two = [viola_chord_two[x] for x in reduceMod11(
304     viola_random_walk_two)]
305
306 seed(11)
307 cello_random_walk_two = []
308 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
309 for i in range(1, 1000):
310     movement = -1 if random() < 0.5 else 1
311     value = cello_random_walk_two[i-1] + movement
312     cello_random_walk_two.append(value)
313 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
314 cello_chord_two = [-17, -12, -8, -7, 2, 8, 2, -7, -8, -12, ]
315 cello_notes_two = [cello_chord_two[x] for x in reduceMod9(
316     cello_random_walk_two)]
317
318 seed(12)
319 bass_random_walk_two = []
320 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
321 for i in range(1, 1000):
322     movement = -1 if random() < 0.5 else 1
323     value = bass_random_walk_two[i-1] + movement
324     bass_random_walk_two.append(value)
325 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
326 bass_chord_two = [-27, -17, -12, -8, -7, 2, -7, -8, -12, -17, ]
327 bass_notes_two = [bass_chord_two[x] for x in reduceMod9(bass_random_walk_two)]

```

```

325 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
326     talea=abjadext.rmakers.Talea(
327         counts=[6, 2, 4, 2, 6, 1, 12, 8, 10, ],
328         denominator=16,
329     ),
330     beamSpecifier=abjadext.rmakers.BeamSpecifier(
331         beamDivisionsTogether=True,
332         beamRests=False,
333     ),
334     extraCountsPerDivision=[0, 1, 0, -1],
335     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
336         leftClasses=[abjad.Note, abjad.Rest],
337         leftCounts=[1, 0, 1],
338     ),
339     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
340         trivialize=True,
341         extractTrivial=True,
342         rewriteRestFilled=True,
343     ),
344 )
345
346 rmaker_two = abjadext.rmakers.NoteRhythmMaker()
347
348 rmaker_three = abjadext.rmakers.IncisedRhythmMaker(
349     inciseSpecifier=abjadext.rmakers.InciseSpecifier(
350         prefixTalea=[-1],
351         prefixCounts=[0, 1, 1, 0],
352         suffixTalea=[-1],
353         suffixCounts=[1, 0],
354         taleaDenominator=16,
355     ),
356 )
357
358 attachmentHandler_one = AttachmentHandler(
359     startingDynamic='mp',
360     endingDynamic='f',
361     hairpinIndicator='<|',
362     articulation='',
363 )
364
365 attachmentHandler_two = AttachmentHandler(
366     startingDynamic='mp',
367     endingDynamic='mf',
368     hairpinIndicator='--',
369     articulation='tenuto',
370 )
371
372 attachmentHandler_three = AttachmentHandler(
373     startingDynamic='mf',
374     endingDynamic='f',
375     hairpinIndicator='--',
376     articulation='marcato',
377 )
378

```

```
379 #####oboe#####
380 flutemusicmaker_one = MusicMaker(
381     rmaker=rmaker_one,
382     pitches=flute_scale,
383     continuous=True,
384     attachment_handler=attachment_handler_one,
385 )
386 flutemusicmaker_two = MusicMaker(
387     rmaker=rmaker_two,
388     pitches=flute_notes_two,
389     continuous=True,
390     attachment_handler=attachment_handler_two,
391 )
392 flutemusicmaker_three = MusicMaker(
393     rmaker=rmaker_three,
394     pitches=flute_notes_one,
395     continuous=True,
396     attachment_handler=attachment_handler_three,
397 )
398 #####violin1#####
399 violin1musicmaker_one = MusicMaker(
400     rmaker=rmaker_one,
401     pitches=violin1_scale,
402     continuous=True,
403     attachment_handler=attachment_handler_one,
404 )
405 violin1musicmaker_two = MusicMaker(
406     rmaker=rmaker_two,
407     pitches=violin1_notes_two,
408     continuous=True,
409     attachment_handler=attachment_handler_two,
410 )
411 violin1musicmaker_three = MusicMaker(
412     rmaker=rmaker_three,
413     pitches=violin1_notes_one,
414     continuous=True,
415     attachment_handler=attachment_handler_three,
416 )
417 #####trumpet#####
418 trumpetmusicmaker_one = MusicMaker(
419     rmaker=rmaker_one,
420     pitches=trumpet_scale,
421     continuous=True,
422     attachment_handler=attachment_handler_one,
423 )
424 trumpetmusicmaker_two = MusicMaker(
425     rmaker=rmaker_two,
426     pitches=trumpet_notes_two,
427     continuous=True,
428     attachment_handler=attachment_handler_two,
429 )
430 trumpetmusicmaker_three = MusicMaker(
431     rmaker=rmaker_three,
432     pitches=trumpet_notes_one,
```

```
433     continuous=True,
434     attachment_handler=attachment_handler_three,
435 )
436 #####clarinet#####
437 clarinetmusicmaker_one = MusicMaker(
438     rmaker=rmaker_one,
439     pitches=clarinet_scale,
440     continuous=True,
441     attachment_handler=attachment_handler_one,
442 )
443 clarinetmusicmaker_two = MusicMaker(
444     rmaker=rmaker_two,
445     pitches=clarinet_notes_two,
446     continuous=True,
447     attachment_handler=attachment_handler_two,
448 )
449 clarinetmusicmaker_three = MusicMaker(
450     rmaker=rmaker_three,
451     pitches=clarinet_notes_one,
452     continuous=True,
453     attachment_handler=attachment_handler_three,
454 )
455 #####violin2#####
456 violin2musicmaker_one = MusicMaker(
457     rmaker=rmaker_one,
458     pitches=violin2_scale,
459     continuous=True,
460     attachment_handler=attachment_handler_one,
461 )
462 violin2musicmaker_two = MusicMaker(
463     rmaker=rmaker_two,
464     pitches=violin2_notes_two,
465     continuous=True,
466     attachment_handler=attachment_handler_two,
467 )
468 violin2musicmaker_three = MusicMaker(
469     rmaker=rmaker_three,
470     pitches=violin2_notes_one,
471     continuous=True,
472     attachment_handler=attachment_handler_three,
473 )
474 #####viola#####
475 violamusicmaker_one = MusicMaker(
476     rmaker=rmaker_one,
477     pitches=viola_scale,
478     continuous=True,
479     attachment_handler=attachment_handler_one,
480 )
481 violamusicmaker_two = MusicMaker(
482     rmaker=rmaker_two,
483     pitches=viola_notes_two,
484     continuous=True,
485     attachment_handler=attachment_handler_two,
486 )
```

```

487 violamusicmaker_three = MusicMaker(
488     rmaker=rmaker_three,
489     pitches=viola_notes_one,
490     continuous=True,
491     attachment_handler=attachment_handler_three,
492 )
493 #####bassoon#####
494 bassoonmusicmaker_one = MusicMaker(
495     rmaker=rmaker_one,
496     pitches=bassoon_scale,
497     continuous=True,
498     attachment_handler=attachment_handler_one,
499 )
500 bassoonmusicmaker_two = MusicMaker(
501     rmaker=rmaker_two,
502     pitches=bassoon_notes_two,
503     continuous=True,
504     attachment_handler=attachment_handler_two,
505 )
506 bassoonmusicmaker_three = MusicMaker(
507     rmaker=rmaker_three,
508     pitches=bassoon_notes_one,
509     continuous=True,
510     attachment_handler=attachment_handler_three,
511 )
512 #####trombone#####
513 trombonemusicmaker_one = MusicMaker(
514     rmaker=rmaker_one,
515     pitches=trombone_scale,
516     continuous=True,
517     attachment_handler=attachment_handler_one,
518 )
519 trombonemusicmaker_two = MusicMaker(
520     rmaker=rmaker_two,
521     pitches=trombone_notes_two,
522     continuous=True,
523     attachment_handler=attachment_handler_two,
524 )
525 trombonemusicmaker_three = MusicMaker(
526     rmaker=rmaker_three,
527     pitches=trombone_notes_one,
528     continuous=True,
529     attachment_handler=attachment_handler_three,
530 )
531 #####cello#####
532 cellomusicmaker_one = MusicMaker(
533     rmaker=rmaker_one,
534     pitches=cello_scale,
535     continuous=True,
536     attachment_handler=attachment_handler_one,
537 )
538 cellomusicmaker_two = MusicMaker(
539     rmaker=rmaker_two,
540     pitches=cello_notes_two,

```

```

541     continuous=True,
542     attachment_handler=attachment_handler_two,
543 )
544 cellomusicmaker_three = MusicMaker(
545     rmaker=rmaker_three,
546     pitches=cello_notes_one,
547     continuous=True,
548     attachment_handler=attachment_handler_three,
549 )
550 #####horn#####
551 hornmusicmaker_one = MusicMaker(
552     rmaker=rmaker_one,
553     pitches=horn_scale,
554     continuous=True,
555     attachment_handler=attachment_handler_one,
556 )
557 hornmusicmaker_two = MusicMaker(
558     rmaker=rmaker_two,
559     pitches=horn_notes_two,
560     continuous=True,
561     attachment_handler=attachment_handler_two,
562 )
563 hornmusicmaker_three = MusicMaker(
564     rmaker=rmaker_three,
565     pitches=horn_notes_one,
566     continuous=True,
567     attachment_handler=attachment_handler_three,
568 )
569 #####tuba#####
570 tubamusicmaker_one = MusicMaker(
571     rmaker=rmaker_one,
572     pitches=tuba_scale,
573     continuous=True,
574     attachment_handler=attachment_handler_one,
575 )
576 tubamusicmaker_two = MusicMaker(
577     rmaker=rmaker_two,
578     pitches=tuba_notes_two,
579     continuous=True,
580     attachment_handler=attachment_handler_two,
581 )
582 tubamusicmaker_three = MusicMaker(
583     rmaker=rmaker_three,
584     pitches=tuba_notes_one,
585     continuous=True,
586     attachment_handler=attachment_handler_three,
587 )
588 #####bass#####
589 bassmusicmaker_one = MusicMaker(
590     rmaker=rmaker_one,
591     pitches=bass_scale,
592     continuous=True,
593     attachment_handler=attachment_handler_one,
594 )

```

```

595 bassmusicmaker_two = MusicMaker(
596     rmaker=rmaker_two,
597     pitches=bass_notes_two,
598     continuous=True,
599     attachment_handler=attachment_handler_two,
600 )
601 bassmusicmaker_three = MusicMaker(
602     rmaker=rmaker_three,
603     pitches=bass_notes_one,
604     continuous=True,
605     attachment_handler=attachment_handler_three,
606 )
607
608 silence_maker = abjadext.rmakers.NoteRhythmMaker(
609     division_masks=[
610         abjadext.rmakers.SilenceMask(
611             pattern=abjad.index([0], 1),
612             ),
613         ],
614     )
615
616
617 class MusicSpecifier:
618
619     def __init__(self, music_maker, voice_name):
620         self.music_maker = music_maker
621         self.voice_name = voice_name
622
623
624 print('Collecting timespans and rmakers ...')
625 ###group one###
626 voice_1_timespan_list = abjad.TimespanList([
627     abjad.AnnotatedTimespan(
628         start_offset=start_offset,
629         stop_offset=stop_offset,
630         annotation=MusicSpecifier(
631             music_maker=music_maker,
632             voice_name='Voice 1',
633             ),
634         )
635     for start_offset, stop_offset, music_maker in [
636         [(9, 4), (10, 4), flutemusicmaker_one],
637         [(15, 4), (18, 4), flutemusicmaker_two],
638         [(22, 4), (25, 4), flutemusicmaker_three],
639         [(27, 4), (30, 4), flutemusicmaker_one],
640         [(30, 4), (32, 4), flutemusicmaker_one],
641         [(35, 4), (39, 4), flutemusicmaker_two],
642         [(42, 4), (43, 4), flutemusicmaker_three],
643         [(43, 4), (44, 4), flutemusicmaker_three],
644         [(45, 4), (46, 4), flutemusicmaker_one],
645         [(46, 4), (50, 4), flutemusicmaker_one],
646         [(54, 4), (57, 4), flutemusicmaker_two],
647         [(59, 4), (60, 4), flutemusicmaker_three],
648         [(65, 4), (67, 4), flutemusicmaker_one],

```

```

649     [(67, 4), (69, 4), flutemusicmaker_one],
650     [(70, 4), (72, 4), flutemusicmaker_two],
651     [(72, 4), (75, 4), flutemusicmaker_two],
652     [(76, 4), (78, 4), flutemusicmaker_three],
653     [(81, 4), (82, 4), flutemusicmaker_one],
654     [(82, 4), (85, 4), flutemusicmaker_one],
655     [(90, 4), (91, 4), flutemusicmaker_two],
656     [(93, 4), (94, 4), flutemusicmaker_three],
657     [(94, 4), (96, 4), flutemusicmaker_three],
658     [(100, 4), (104, 4), flutemusicmaker_one],
659     [(104, 4), (105, 4), flutemusicmaker_one],
660     [(106, 4), (107, 4), flutemusicmaker_two],
661     [(107, 4), (108, 4), flutemusicmaker_two],
662     [(111, 4), (114, 4), flutemusicmaker_one],
663     [(114, 4), (115, 4), flutemusicmaker_one],
664     [(116, 4), (119, 4), flutemusicmaker_one],
665     [(119, 4), (120, 4), flutemusicmaker_one],
666     [(121, 4), (123, 4), flutemusicmaker_one],
667     [(123, 4), (125, 4), flutemusicmaker_one],
668     [(126, 4), (131, 4), flutemusicmaker_two],
669     [(131, 4), (133, 4), flutemusicmaker_two],
670     [(136, 4), (141, 4), flutemusicmaker_two],
671     [(148, 4), (150, 4), flutemusicmaker_two],
672     [(150, 4), (153, 4), flutemusicmaker_three],
673     [(155, 4), (159, 4), flutemusicmaker_three],
674     [(162, 4), (164, 4), flutemusicmaker_three],
675     [(168, 4), (171, 4), flutemusicmaker_three],
676     [(173, 4), (175, 4), flutemusicmaker_three],
677     [(175, 4), (177, 4), flutemusicmaker_three],
678     [(180, 4), (182, 4), flutemusicmaker_three],
679     [(186, 4), (190, 4), flutemusicmaker_three],
680     [(190, 4), (191, 4), silence_maker],
681 ]
682 ])
683
684 voice_5_timespan_list = abjad.TimespanList([
685     abjad.AnnotatedTimespan(
686         start_offset=start_offset,
687         stop_offset=stop_offset,
688         annotation=MusicSpecifier(
689             music_maker=music_maker,
690             voice_name='Voice 5',
691         ),
692     )
693     for start_offset, stop_offset, music_maker in [
694         [(9, 4), (10, 4), trumpetmusicmaker_one],
695         [(14, 4), (18, 4), trumpetmusicmaker_two],
696         [(23, 4), (25, 4), trumpetmusicmaker_three],
697         [(27, 4), (30, 4), trumpetmusicmaker_one],
698         [(30, 4), (32, 4), trumpetmusicmaker_one],
699         [(35, 4), (39, 4), trumpetmusicmaker_two],
700         [(42, 4), (43, 4), trumpetmusicmaker_three],
701         [(43, 4), (44, 4), trumpetmusicmaker_three],
702         [(45, 4), (46, 4), trumpetmusicmaker_one],

```

```

703     [(46, 4), (50, 4), trumpetmusicmaker_one],
704     [(54, 4), (57, 4), trumpetmusicmaker_two],
705     [(59, 4), (60, 4), trumpetmusicmaker_three],
706     [(65, 4), (67, 4), trumpetmusicmaker_one],
707     [(67, 4), (69, 4), trumpetmusicmaker_one],
708     [(70, 4), (72, 4), trumpetmusicmaker_two],
709     [(72, 4), (75, 4), trumpetmusicmaker_two],
710     [(76, 4), (78, 4), trumpetmusicmaker_three],
711     [(81, 4), (82, 4), trumpetmusicmaker_one],
712     [(82, 4), (85, 4), trumpetmusicmaker_one],
713     [(90, 4), (91, 4), trumpetmusicmaker_two],
714     [(93, 4), (94, 4), trumpetmusicmaker_three],
715     [(94, 4), (96, 4), trumpetmusicmaker_three],
716     [(100, 4), (104, 4), trumpetmusicmaker_one],
717     [(104, 4), (105, 4), trumpetmusicmaker_one],
718     [(106, 4), (107, 4), trumpetmusicmaker_two],
719     [(107, 4), (108, 4), trumpetmusicmaker_two],
720     [(111, 4), (114, 4), trumpetmusicmaker_one],
721     [(114, 4), (115, 4), trumpetmusicmaker_one],
722     [(116, 4), (119, 4), trumpetmusicmaker_one],
723     [(119, 4), (120, 4), trumpetmusicmaker_one],
724     [(121, 4), (123, 4), trumpetmusicmaker_one],
725     [(123, 4), (125, 4), trumpetmusicmaker_one],
726     [(126, 4), (131, 4), trumpetmusicmaker_two],
727     [(131, 4), (133, 4), trumpetmusicmaker_two],
728     [(136, 4), (141, 4), trumpetmusicmaker_two],
729     [(148, 4), (150, 4), trumpetmusicmaker_two],
730     [(150, 4), (154, 4), trumpetmusicmaker_three],
731     [(157, 4), (159, 4), trumpetmusicmaker_three],
732     [(163, 4), (164, 4), trumpetmusicmaker_three],
733     [(164, 4), (166, 4), trumpetmusicmaker_three],
734     [(168, 4), (172, 4), trumpetmusicmaker_three],
735     [(175, 4), (177, 4), trumpetmusicmaker_three],
736     [(181, 4), (183, 4), trumpetmusicmaker_three],
737     [(183, 4), (184, 4), trumpetmusicmaker_three],
738     [(186, 4), (190, 4), trumpetmusicmaker_three],
739   ]
740 ])
741
742 voice_8_timestspan_list = abjad.TimespanList([
743     abjad.AnnotatedTimespan(
744         start_offset=start_offset,
745         stop_offset=stop_offset,
746         annotation=MusicSpecifier(
747             music_maker=music_maker,
748             voice_name='Voice 8',
749         ),
750     )
751     for start_offset, stop_offset, music_maker in [
752         [(9, 4), (10, 4), violin1musicmaker_one],
753         [(14, 4), (18, 4), violin1musicmaker_two],
754         [(22, 4), (25, 4), violin1musicmaker_three],
755         [(27, 4), (30, 4), violin1musicmaker_one],
756         [(35, 4), (39, 4), violin1musicmaker_two],

```

```

757     [(42, 4), (43, 4), violin1musicmaker_three],
758     [(43, 4), (44, 4), violin1musicmaker_three],
759     [(45, 4), (46, 4), violin1musicmaker_one],
760     [(46, 4), (50, 4), violin1musicmaker_one],
761     [(54, 4), (57, 4), violin1musicmaker_two],
762     [(59, 4), (60, 4), violin1musicmaker_three],
763     [(65, 4), (67, 4), violin1musicmaker_one],
764     [(67, 4), (69, 4), violin1musicmaker_one],
765     [(70, 4), (72, 4), violin1musicmaker_two],
766     [(72, 4), (75, 4), violin1musicmaker_two],
767     [(76, 4), (78, 4), violin1musicmaker_three],
768     [(81, 4), (82, 4), violin1musicmaker_one],
769     [(82, 4), (85, 4), violin1musicmaker_one],
770     [(90, 4), (91, 4), violin1musicmaker_two],
771     [(93, 4), (94, 4), violin1musicmaker_three],
772     [(94, 4), (96, 4), violin1musicmaker_three],
773     [(100, 4), (104, 4), violin1musicmaker_one],
774     [(104, 4), (105, 4), violin1musicmaker_one],
775     [(106, 4), (107, 4), violin1musicmaker_two],
776     [(107, 4), (108, 4), violin1musicmaker_two],
777     [(111, 4), (114, 4), violin1musicmaker_one],
778     [(114, 4), (115, 4), violin1musicmaker_one],
779     [(116, 4), (119, 4), violin1musicmaker_one],
780     [(119, 4), (120, 4), violin1musicmaker_one],
781     [(121, 4), (123, 4), violin1musicmaker_one],
782     [(123, 4), (125, 4), violin1musicmaker_one],
783     [(126, 4), (131, 4), violin1musicmaker_two],
784     [(131, 4), (133, 4), violin1musicmaker_two],
785     [(136, 4), (141, 4), violin1musicmaker_two],
786     [(148, 4), (150, 4), violin1musicmaker_two],
787     [(150, 4), (152, 4), violin1musicmaker_three],
788     [(156, 4), (159, 4), violin1musicmaker_three],
789     [(161, 4), (164, 4), violin1musicmaker_three],
790     [(164, 4), (165, 4), violin1musicmaker_three],
791     [(168, 4), (170, 4), violin1musicmaker_three],
792     [(174, 4), (175, 4), violin1musicmaker_three],
793     [(175, 4), (177, 4), violin1musicmaker_three],
794     [(179, 4), (183, 4), violin1musicmaker_three],
795     [(186, 4), (190, 4), violin1musicmaker_three],
796 ]
797 ])
798
799 ###group two###
800 voice_2_timespan_list = abjad.TimespanList([
801     abjad.AnnotatedTimespan(
802         start_offset=start_offset,
803         stop_offset=stop_offset,
804         annotation=MusicSpecifier(
805             music_maker=music_maker,
806             voice_name='Voice 2',
807         ),
808     )
809     for start_offset, stop_offset, music_maker in [
810         [(2, 4), (5, 4), clarinetmusicmaker_one],

```

```

811     [(10, 4), (11, 4), clarinetmusicmaker_two],
812     [(11, 4), (13, 4), clarinetmusicmaker_two],
813     [(16, 4), (18, 4), clarinetmusicmaker_three],
814     [(21, 4), (22, 4), clarinetmusicmaker_one],
815     [(22, 4), (25, 4), clarinetmusicmaker_one],
816     [(35, 4), (40, 4), clarinetmusicmaker_one],
817     [(44, 4), (46, 4), clarinetmusicmaker_two],
818     [(46, 4), (47, 4), clarinetmusicmaker_two],
819     [(49, 4), (50, 4), clarinetmusicmaker_three],
820     [(55, 4), (59, 4), clarinetmusicmaker_one],
821     [(62, 4), (64, 4), clarinetmusicmaker_two],
822     [(65, 4), (67, 4), clarinetmusicmaker_three],
823     [(67, 4), (70, 4), clarinetmusicmaker_three],
824     [(70, 4), (71, 4), clarinetmusicmaker_three],
825     [(73, 4), (75, 4), clarinetmusicmaker_two],
826     [(75, 4), (76, 4), clarinetmusicmaker_two],
827     [(80, 4), (82, 4), clarinetmusicmaker_one],
828     [(82, 4), (85, 4), clarinetmusicmaker_one],
829     [(86, 4), (88, 4), clarinetmusicmaker_two],
830     [(91, 4), (94, 4), clarinetmusicmaker_three],
831     [(94, 4), (95, 4), clarinetmusicmaker_three],
832     [(100, 4), (101, 4), clarinetmusicmaker_two],
833     [(103, 4), (104, 4), clarinetmusicmaker_one],
834     [(104, 4), (106, 4), clarinetmusicmaker_one],
835     [(110, 4), (114, 4), clarinetmusicmaker_one],
836     [(115, 4), (119, 4), clarinetmusicmaker_one],
837     [(120, 4), (123, 4), clarinetmusicmaker_one],
838     [(123, 4), (124, 4), clarinetmusicmaker_one],
839     [(125, 4), (126, 4), clarinetmusicmaker_two],
840     [(129, 4), (131, 4), clarinetmusicmaker_two],
841     [(131, 4), (134, 4), clarinetmusicmaker_two],
842     [(141, 4), (144, 4), clarinetmusicmaker_two],
843     [(149, 4), (150, 4), clarinetmusicmaker_two],
844     [(155, 4), (159, 4), clarinetmusicmaker_three],
845     [(162, 4), (164, 4), clarinetmusicmaker_three],
846     [(165, 4), (168, 4), clarinetmusicmaker_three],
847     [(168, 4), (170, 4), clarinetmusicmaker_three],
848     [(174, 4), (175, 4), clarinetmusicmaker_three],
849     [(175, 4), (177, 4), clarinetmusicmaker_three],
850     [(179, 4), (180, 4), clarinetmusicmaker_three],
851     [(185, 4), (186, 4), clarinetmusicmaker_three],
852     [(186, 4), (190, 4), clarinetmusicmaker_three],
853     ]
854   ])
855
856 voice_9_timespan_list = abjad.TimespanList([
857     abjad.AnnotatedTimespan(
858         start_offset=start_offset,
859         stop_offset=stop_offset,
860         annotation=MusicSpecifier(
861             music_maker=music_maker,
862             voice_name='Voice 9',
863         ),
864     )

```

```

865     for start_offset, stop_offset, music_maker in [
866         [(2, 4), (5, 4), violin2musicmaker_one],
867         [(9, 4), (11, 4), violin2musicmaker_two],
868         [(11, 4), (13, 4), violin2musicmaker_two],
869         [(16, 4), (18, 4), violin2musicmaker_three],
870         [(21, 4), (22, 4), violin2musicmaker_one],
871         [(22, 4), (23, 4), violin2musicmaker_one],
872         [(35, 4), (40, 4), violin2musicmaker_one],
873         [(44, 4), (46, 4), violin2musicmaker_two],
874         [(46, 4), (47, 4), violin2musicmaker_two],
875         [(49, 4), (50, 4), violin2musicmaker_three],
876         [(55, 4), (59, 4), violin2musicmaker_one],
877         [(62, 4), (64, 4), violin2musicmaker_two],
878         [(65, 4), (67, 4), violin2musicmaker_three],
879         [(67, 4), (70, 4), violin2musicmaker_three],
880         [(70, 4), (71, 4), violin2musicmaker_three],
881         [(73, 4), (75, 4), violin2musicmaker_two],
882         [(75, 4), (76, 4), violin2musicmaker_two],
883         [(80, 4), (82, 4), violin2musicmaker_one],
884         [(82, 4), (85, 4), violin2musicmaker_one],
885         [(86, 4), (88, 4), violin2musicmaker_two],
886         [(91, 4), (94, 4), violin2musicmaker_three],
887         [(94, 4), (95, 4), violin2musicmaker_three],
888         [(100, 4), (101, 4), violin2musicmaker_two],
889         [(103, 4), (104, 4), violin2musicmaker_one],
890         [(104, 4), (106, 4), violin2musicmaker_one],
891         [(110, 4), (114, 4), violin2musicmaker_one],
892         [(115, 4), (119, 4), violin2musicmaker_one],
893         [(120, 4), (123, 4), violin2musicmaker_one],
894         [(123, 4), (124, 4), violin2musicmaker_one],
895         [(125, 4), (126, 4), violin2musicmaker_two],
896         [(129, 4), (131, 4), violin2musicmaker_two],
897         [(131, 4), (134, 4), violin2musicmaker_two],
898         [(141, 4), (144, 4), violin2musicmaker_two],
899         [(149, 4), (150, 4), violin2musicmaker_two],
900         [(154, 4), (157, 4), violin2musicmaker_three],
901         [(159, 4), (160, 4), violin2musicmaker_three],
902         [(165, 4), (168, 4), violin2musicmaker_three],
903         [(168, 4), (169, 4), violin2musicmaker_three],
904         [(172, 4), (174, 4), violin2musicmaker_three],
905         [(175, 4), (179, 4), violin2musicmaker_three],
906         [(179, 4), (180, 4), violin2musicmaker_three],
907         [(184, 4), (186, 4), violin2musicmaker_three],
908         [(186, 4), (190, 4), violin2musicmaker_three],
909     ]
910 ])
911
912 voice_10_timespan_list = abjad.TimespanList([
913     abjad.AnnotatedTimespan(
914         start_offset=start_offset,
915         stop_offset=stop_offset,
916         annotation=MusicSpecifier(
917             music_maker=music_maker,
918             voice_name='Voice 10',

```

```

919     ),
920
921     for start_offset, stop_offset, music_maker in [
922         [(2, 4), (5, 4), violamusicmaker_one],
923         [(9, 4), (11, 4), violamusicmaker_two],
924         [(11, 4), (13, 4), violamusicmaker_two],
925         [(17, 4), (18, 4), violamusicmaker_three],
926         [(21, 4), (22, 4), violamusicmaker_one],
927         [(22, 4), (25, 4), violamusicmaker_one],
928         [(29, 4), (30, 4), violamusicmaker_two],
929         [(30, 4), (32, 4), violamusicmaker_two],
930         [(35, 4), (40, 4), violamusicmaker_one],
931         [(44, 4), (46, 4), violamusicmaker_two],
932         [(46, 4), (47, 4), violamusicmaker_two],
933         [(49, 4), (50, 4), violamusicmaker_three],
934         [(55, 4), (59, 4), violamusicmaker_one],
935         [(62, 4), (64, 4), violamusicmaker_two],
936         [(65, 4), (67, 4), violamusicmaker_three],
937         [(67, 4), (70, 4), violamusicmaker_three],
938         [(70, 4), (71, 4), violamusicmaker_three],
939         [(73, 4), (75, 4), violamusicmaker_two],
940         [(75, 4), (76, 4), violamusicmaker_two],
941         [(80, 4), (82, 4), violamusicmaker_one],
942         [(82, 4), (85, 4), violamusicmaker_one],
943         [(86, 4), (88, 4), violamusicmaker_two],
944         [(91, 4), (94, 4), violamusicmaker_three],
945         [(94, 4), (95, 4), violamusicmaker_three],
946         [(100, 4), (101, 4), violamusicmaker_two],
947         [(103, 4), (104, 4), violamusicmaker_one],
948         [(104, 4), (106, 4), violamusicmaker_one],
949         [(110, 4), (114, 4), violamusicmaker_one],
950         [(115, 4), (119, 4), violamusicmaker_one],
951         [(120, 4), (123, 4), violamusicmaker_one],
952         [(123, 4), (124, 4), violamusicmaker_one],
953         [(125, 4), (126, 4), violamusicmaker_two],
954         [(129, 4), (131, 4), violamusicmaker_two],
955         [(131, 4), (134, 4), violamusicmaker_two],
956         [(141, 4), (144, 4), violamusicmaker_two],
957         [(149, 4), (150, 4), violamusicmaker_two],
958         [(153, 4), (154, 4), violamusicmaker_three],
959         [(154, 4), (155, 4), violamusicmaker_three],
960         [(156, 4), (159, 4), violamusicmaker_three],
961         [(159, 4), (161, 4), violamusicmaker_three],
962         [(165, 4), (168, 4), violamusicmaker_three],
963         [(170, 4), (171, 4), violamusicmaker_three],
964         [(176, 4), (179, 4), violamusicmaker_three],
965         [(179, 4), (180, 4), violamusicmaker_three],
966         [(183, 4), (185, 4), violamusicmaker_three],
967         [(186, 4), (190, 4), violamusicmaker_three],
968     ]
969 ])
970
971 ####group three#####
972 voice_3_timespan_list = abjad.TimespanList([

```

```

973     abjad.AnnotatedTimespan(
974         start_offset=start_offset,
975         stop_offset=stop_offset,
976         annotation=MusicSpecifier(
977             music_maker=music_maker,
978             voice_name='Voice 3',
979         ),
980     )
981     for start_offset, stop_offset, music_maker in [
982         [(7, 4), (11, 4), bassoonmusicmaker_one],
983         [(15, 4), (16, 4), bassoonmusicmaker_two],
984         [(19, 4), (22, 4), bassoonmusicmaker_three],
985         [(22, 4), (23, 4), bassoonmusicmaker_three],
986         [(27, 4), (30, 4), bassoonmusicmaker_one],
987         [(32, 4), (35, 4), bassoonmusicmaker_two],
988         [(35, 4), (36, 4), bassoonmusicmaker_three],
989         [(37, 4), (40, 4), bassoonmusicmaker_two],
990         [(40, 4), (42, 4), bassoonmusicmaker_two],
991         [(46, 4), (49, 4), bassoonmusicmaker_one],
992         [(51, 4), (52, 4), bassoonmusicmaker_three],
993         [(57, 4), (59, 4), bassoonmusicmaker_two],
994         [(59, 4), (61, 4), bassoonmusicmaker_two],
995         [(64, 4), (66, 4), bassoonmusicmaker_one],
996         [(67, 4), (70, 4), bassoonmusicmaker_three],
997         [(70, 4), (72, 4), bassoonmusicmaker_one],
998         [(72, 4), (73, 4), bassoonmusicmaker_one],
999         [(77, 4), (79, 4), bassoonmusicmaker_two],
1000        [(79, 4), (82, 4), bassoonmusicmaker_two],
1001        [(83, 4), (85, 4), bassoonmusicmaker_three],
1002        [(88, 4), (89, 4), bassoonmusicmaker_two],
1003        [(89, 4), (92, 4), bassoonmusicmaker_two],
1004        [(97, 4), (98, 4), bassoonmusicmaker_one],
1005        [(100, 4), (103, 4), bassoonmusicmaker_two],
1006        [(107, 4), (110, 4), bassoonmusicmaker_three],
1007        [(110, 4), (112, 4), bassoonmusicmaker_one],
1008        [(113, 4), (114, 4), bassoonmusicmaker_one],
1009        [(114, 4), (117, 4), bassoonmusicmaker_one],
1010        [(118, 4), (119, 4), bassoonmusicmaker_one],
1011        [(119, 4), (122, 4), bassoonmusicmaker_one],
1012        [(123, 4), (125, 4), bassoonmusicmaker_one],
1013        [(126, 4), (131, 4), bassoonmusicmaker_two],
1014        [(138, 4), (141, 4), bassoonmusicmaker_two],
1015        [(146, 4), (150, 4), bassoonmusicmaker_two],
1016        [(150, 4), (154, 4), bassoonmusicmaker_three],
1017        [(154, 4), (155, 4), bassoonmusicmaker_three],
1018        [(159, 4), (162, 4), bassoonmusicmaker_three],
1019        [(164, 4), (165, 4), bassoonmusicmaker_three],
1020        [(170, 4), (172, 4), bassoonmusicmaker_three],
1021        [(172, 4), (174, 4), bassoonmusicmaker_three],
1022        [(177, 4), (179, 4), bassoonmusicmaker_three],
1023        [(180, 4), (183, 4), bassoonmusicmaker_three],
1024        [(183, 4), (185, 4), bassoonmusicmaker_three],
1025        [(186, 4), (190, 4), bassoonmusicmaker_three],
1026    ]

```

```

1027 ])
1028
1029 voice_6_timespan_list = abjad.TimespanList([
1030     abjad.AnnotatedTimespan(
1031         start_offset=start_offset,
1032         stop_offset=stop_offset,
1033         annotation=MusicSpecifier(
1034             music_maker=music_maker,
1035             voice_name='Voice 6',
1036         ),
1037     )
1038     for start_offset, stop_offset, music_maker in [
1039         [(7, 4), (11, 4), trombonemusicmaker_one],
1040         [(14, 4), (16, 4), trombonemusicmaker_two],
1041         [(19, 4), (22, 4), trombonemusicmaker_three],
1042         [(22, 4), (23, 4), trombonemusicmaker_three],
1043         [(27, 4), (29, 4), trombonemusicmaker_one],
1044         [(35, 4), (36, 4), trombonemusicmaker_three],
1045         [(37, 4), (40, 4), trombonemusicmaker_two],
1046         [(40, 4), (42, 4), trombonemusicmaker_two],
1047         [(46, 4), (49, 4), trombonemusicmaker_one],
1048         [(51, 4), (52, 4), trombonemusicmaker_three],
1049         [(57, 4), (59, 4), trombonemusicmaker_two],
1050         [(59, 4), (61, 4), trombonemusicmaker_two],
1051         [(64, 4), (66, 4), trombonemusicmaker_one],
1052         [(67, 4), (70, 4), trombonemusicmaker_three],
1053         [(70, 4), (72, 4), trombonemusicmaker_one],
1054         [(72, 4), (73, 4), trombonemusicmaker_one],
1055         [(77, 4), (79, 4), trombonemusicmaker_two],
1056         [(79, 4), (82, 4), trombonemusicmaker_two],
1057         [(83, 4), (85, 4), trombonemusicmaker_three],
1058         [(88, 4), (89, 4), trombonemusicmaker_two],
1059         [(89, 4), (92, 4), trombonemusicmaker_two],
1060         [(97, 4), (98, 4), trombonemusicmaker_one],
1061         [(100, 4), (103, 4), trombonemusicmaker_two],
1062         [(107, 4), (110, 4), trombonemusicmaker_three],
1063         [(110, 4), (112, 4), trombonemusicmaker_one],
1064         [(113, 4), (114, 4), trombonemusicmaker_one],
1065         [(114, 4), (117, 4), trombonemusicmaker_one],
1066         [(118, 4), (119, 4), trombonemusicmaker_one],
1067         [(119, 4), (122, 4), trombonemusicmaker_one],
1068         [(123, 4), (125, 4), trombonemusicmaker_one],
1069         [(126, 4), (131, 4), trombonemusicmaker_two],
1070         [(138, 4), (141, 4), trombonemusicmaker_two],
1071         [(146, 4), (150, 4), trombonemusicmaker_two],
1072         [(150, 4), (154, 4), trombonemusicmaker_three],
1073         [(157, 4), (159, 4), trombonemusicmaker_three],
1074         [(160, 4), (164, 4), trombonemusicmaker_three],
1075         [(164, 4), (165, 4), trombonemusicmaker_three],
1076         [(169, 4), (172, 4), trombonemusicmaker_three],
1077         [(174, 4), (175, 4), trombonemusicmaker_three],
1078         [(180, 4), (183, 4), trombonemusicmaker_three],
1079         [(183, 4), (184, 4), trombonemusicmaker_three],
1080         [(186, 4), (190, 4), trombonemusicmaker_three],

```

```

1081     ]
1082   ])
1083
1084 voice_11_timespan_list = abjad.TimespanList([
1085     abjad.AnnotatedTimespan(
1086         start_offset=start_offset,
1087         stop_offset=stop_offset,
1088         annotation=MusicSpecifier(
1089             music_maker=music_maker,
1090             voice_name='Voice 11',
1091         ),
1092     )
1093     for start_offset, stop_offset, music_maker in [
1094         [(7, 4), (11, 4), cellomusicmaker_one],
1095         [(14, 4), (16, 4), cellomusicmaker_two],
1096         [(21, 4), (22, 4), cellomusicmaker_three],
1097         [(22, 4), (23, 4), cellomusicmaker_three],
1098         [(27, 4), (30, 4), cellomusicmaker_one],
1099         [(35, 4), (36, 4), cellomusicmaker_three],
1100         [(37, 4), (40, 4), cellomusicmaker_two],
1101         [(40, 4), (42, 4), cellomusicmaker_two],
1102         [(46, 4), (49, 4), cellomusicmaker_one],
1103         [(51, 4), (52, 4), cellomusicmaker_three],
1104         [(57, 4), (59, 4), cellomusicmaker_two],
1105         [(59, 4), (61, 4), cellomusicmaker_two],
1106         [(64, 4), (66, 4), cellomusicmaker_one],
1107         [(67, 4), (70, 4), cellomusicmaker_three],
1108         [(70, 4), (72, 4), cellomusicmaker_one],
1109         [(72, 4), (73, 4), cellomusicmaker_one],
1110         [(77, 4), (79, 4), cellomusicmaker_two],
1111         [(79, 4), (82, 4), cellomusicmaker_two],
1112         [(83, 4), (85, 4), cellomusicmaker_three],
1113         [(88, 4), (89, 4), cellomusicmaker_two],
1114         [(89, 4), (92, 4), cellomusicmaker_two],
1115         [(97, 4), (98, 4), cellomusicmaker_one],
1116         [(100, 4), (103, 4), cellomusicmaker_two],
1117         [(107, 4), (110, 4), cellomusicmaker_three],
1118         [(110, 4), (112, 4), cellomusicmaker_one],
1119         [(113, 4), (114, 4), cellomusicmaker_one],
1120         [(114, 4), (117, 4), cellomusicmaker_one],
1121         [(118, 4), (119, 4), cellomusicmaker_one],
1122         [(119, 4), (122, 4), cellomusicmaker_one],
1123         [(123, 4), (125, 4), cellomusicmaker_one],
1124         [(126, 4), (131, 4), cellomusicmaker_two],
1125         [(138, 4), (141, 4), cellomusicmaker_two],
1126         [(146, 4), (150, 4), cellomusicmaker_two],
1127         [(150, 4), (153, 4), cellomusicmaker_three],
1128         [(155, 4), (156, 4), cellomusicmaker_three],
1129         [(161, 4), (164, 4), cellomusicmaker_three],
1130         [(164, 4), (165, 4), cellomusicmaker_three],
1131         [(168, 4), (170, 4), cellomusicmaker_three],
1132         [(171, 4), (172, 4), cellomusicmaker_three],
1133         [(172, 4), (175, 4), cellomusicmaker_three],
1134         [(175, 4), (176, 4), cellomusicmaker_three],

```

```

1135     [(180, 4), (183, 4), cellomusicmaker_three],
1136     [(185, 4), (186, 4), cellomusicmaker_three],
1137     [(186, 4), (190, 4), cellomusicmaker_three],
1138 ]
1139 ])
1140
1141 ###group four###
1142 voice_4_timespan_list = abjad.TimespanList([
1143     abjad.AnnotatedTimespan(
1144         start_offset=start_offset,
1145         stop_offset=stop_offset,
1146         annotation=MusicSpecifier(
1147             music_maker=music_maker,
1148             voice_name='Voice 4',
1149         ),
1150     )
1151     for start_offset, stop_offset, music_maker in [
1152         [(0, 4), (5, 4), hornmusicmaker_one],
1153         [(8, 4), (10, 4), hornmusicmaker_two],
1154         [(14, 4), (18, 4), hornmusicmaker_three],
1155         [(21, 4), (22, 4), hornmusicmaker_one],
1156         [(22, 4), (23, 4), hornmusicmaker_one],
1157         [(38, 4), (40, 4), hornmusicmaker_two],
1158         [(41, 4), (43, 4), hornmusicmaker_one],
1159         [(43, 4), (46, 4), hornmusicmaker_one],
1160         [(50, 4), (53, 4), hornmusicmaker_three],
1161         [(55, 4), (56, 4), hornmusicmaker_two],
1162         [(61, 4), (64, 4), hornmusicmaker_one],
1163         [(64, 4), (65, 4), hornmusicmaker_one],
1164         [(68, 4), (70, 4), hornmusicmaker_three],
1165         [(70, 4), (72, 4), hornmusicmaker_two],
1166         [(72, 4), (74, 4), hornmusicmaker_two],
1167         [(79, 4), (80, 4), hornmusicmaker_three],
1168         [(82, 4), (85, 4), hornmusicmaker_two],
1169         [(89, 4), (94, 4), hornmusicmaker_one],
1170         [(95, 4), (97, 4), hornmusicmaker_two],
1171         [(100, 4), (104, 4), hornmusicmaker_three],
1172         [(109, 4), (110, 4), hornmusicmaker_two],
1173         [(110, 4), (111, 4), hornmusicmaker_one],
1174         [(112, 4), (114, 4), hornmusicmaker_one],
1175         [(114, 4), (116, 4), hornmusicmaker_one],
1176         [(117, 4), (119, 4), hornmusicmaker_one],
1177         [(119, 4), (121, 4), hornmusicmaker_one],
1178         [(122, 4), (123, 4), hornmusicmaker_one],
1179         [(123, 4), (125, 4), hornmusicmaker_one],
1180         [(133, 4), (136, 4), hornmusicmaker_two],
1181         [(142, 4), (146, 4), hornmusicmaker_two],
1182         [(146, 4), (150, 4), hornmusicmaker_two],
1183         [(153, 4), (154, 4), hornmusicmaker_three],
1184         [(154, 4), (155, 4), hornmusicmaker_three],
1185         [(159, 4), (162, 4), hornmusicmaker_three],
1186         [(164, 4), (168, 4), hornmusicmaker_three],
1187         [(171, 4), (172, 4), hornmusicmaker_three],
1188         [(172, 4), (173, 4), hornmusicmaker_three],

```

```

1189     [(177, 4), (179, 4), hornmusicmaker_three],
1190     [(179, 4), (180, 4), hornmusicmaker_three],
1191     [(182, 4), (183, 4), hornmusicmaker_three],
1192     [(183, 4), (186, 4), hornmusicmaker_three],
1193     [(186, 4), (190, 4), hornmusicmaker_three],
1194 ]
1195 ])
1196
1197 voice_7_timestspan_list = abjad.TimespanList([
1198     abjad.AnnotatedTimespan(
1199         start_offset=start_offset,
1200         stop_offset=stop_offset,
1201         annotation=MusicSpecifier(
1202             music_maker=music_maker,
1203             voice_name='Voice 7',
1204         ),
1205     )
1206     for start_offset, stop_offset, music_maker in [
1207         [(0, 4), (5, 4), tubamusicmaker_one],
1208         [(8, 4), (10, 4), tubamusicmaker_two],
1209         [(14, 4), (18, 4), tubamusicmaker_three],
1210         [(21, 4), (22, 4), tubamusicmaker_one],
1211         [(22, 4), (23, 4), tubamusicmaker_one],
1212         [(26, 4), (30, 4), tubamusicmaker_two],
1213         [(38, 4), (40, 4), tubamusicmaker_two],
1214         [(41, 4), (43, 4), tubamusicmaker_one],
1215         [(43, 4), (46, 4), tubamusicmaker_one],
1216         [(50, 4), (53, 4), tubamusicmaker_three],
1217         [(55, 4), (56, 4), tubamusicmaker_two],
1218         [(61, 4), (64, 4), tubamusicmaker_one],
1219         [(64, 4), (65, 4), tubamusicmaker_one],
1220         [(68, 4), (70, 4), tubamusicmaker_three],
1221         [(70, 4), (72, 4), tubamusicmaker_two],
1222         [(72, 4), (74, 4), tubamusicmaker_two],
1223         [(79, 4), (80, 4), tubamusicmaker_three],
1224         [(82, 4), (85, 4), tubamusicmaker_two],
1225         [(89, 4), (94, 4), tubamusicmaker_one],
1226         [(95, 4), (97, 4), tubamusicmaker_two],
1227         [(100, 4), (104, 4), tubamusicmaker_three],
1228         [(109, 4), (110, 4), tubamusicmaker_two],
1229         [(110, 4), (111, 4), tubamusicmaker_one],
1230         [(112, 4), (114, 4), tubamusicmaker_one],
1231         [(114, 4), (116, 4), tubamusicmaker_one],
1232         [(117, 4), (119, 4), tubamusicmaker_one],
1233         [(119, 4), (121, 4), tubamusicmaker_one],
1234         [(122, 4), (123, 4), tubamusicmaker_one],
1235         [(123, 4), (125, 4), tubamusicmaker_one],
1236         [(133, 4), (136, 4), tubamusicmaker_two],
1237         [(142, 4), (146, 4), tubamusicmaker_two],
1238         [(146, 4), (150, 4), tubamusicmaker_two],
1239         [(154, 4), (157, 4), tubamusicmaker_three],
1240         [(159, 4), (163, 4), tubamusicmaker_three],
1241         [(166, 4), (168, 4), tubamusicmaker_three],
1242         [(172, 4), (175, 4), tubamusicmaker_three],

```

```

1243     [(177, 4), (179, 4), tubamusicmaker_three],
1244     [(179, 4), (181, 4), tubamusicmaker_three],
1245     [(184, 4), (186, 4), tubamusicmaker_three],
1246     [(186, 4), (190, 4), tubamusicmaker_three],
1247 ]
1248 ])
1249
1250 voice_12_timespan_list = abjad.TimespanList([
1251     abjad.AnnotatedTimespan(
1252         start_offset=start_offset,
1253         stop_offset=stop_offset,
1254         annotation=MusicSpecifier(
1255             music_maker=music_maker,
1256             voice_name='Voice 12',
1257         ),
1258     )
1259     for start_offset, stop_offset, music_maker in [
1260         [(0, 4), (5, 4), bassmusicmaker_one],
1261         [(8, 4), (10, 4), bassmusicmaker_two],
1262         [(14, 4), (18, 4), bassmusicmaker_three],
1263         [(21, 4), (22, 4), bassmusicmaker_one],
1264         [(22, 4), (23, 4), bassmusicmaker_one],
1265         [(38, 4), (40, 4), bassmusicmaker_two],
1266         [(41, 4), (43, 4), bassmusicmaker_one],
1267         [(43, 4), (46, 4), bassmusicmaker_one],
1268         [(50, 4), (53, 4), bassmusicmaker_three],
1269         [(55, 4), (56, 4), bassmusicmaker_two],
1270         [(61, 4), (64, 4), bassmusicmaker_one],
1271         [(64, 4), (65, 4), bassmusicmaker_one],
1272         [(68, 4), (70, 4), bassmusicmaker_three],
1273         [(70, 4), (72, 4), bassmusicmaker_two],
1274         [(72, 4), (74, 4), bassmusicmaker_two],
1275         [(79, 4), (80, 4), bassmusicmaker_three],
1276         [(82, 4), (85, 4), bassmusicmaker_two],
1277         [(89, 4), (94, 4), bassmusicmaker_one],
1278         [(95, 4), (97, 4), bassmusicmaker_two],
1279         [(100, 4), (104, 4), bassmusicmaker_three],
1280         [(109, 4), (110, 4), bassmusicmaker_two],
1281         [(110, 4), (111, 4), bassmusicmaker_one],
1282         [(112, 4), (114, 4), bassmusicmaker_one],
1283         [(114, 4), (116, 4), bassmusicmaker_one],
1284         [(117, 4), (119, 4), bassmusicmaker_one],
1285         [(119, 4), (121, 4), bassmusicmaker_one],
1286         [(122, 4), (123, 4), bassmusicmaker_one],
1287         [(123, 4), (125, 4), bassmusicmaker_one],
1288         [(133, 4), (136, 4), bassmusicmaker_two],
1289         [(142, 4), (146, 4), bassmusicmaker_two],
1290         [(146, 4), (150, 4), bassmusicmaker_two],
1291         [(152, 4), (154, 4), bassmusicmaker_three],
1292         [(154, 4), (156, 4), bassmusicmaker_three],
1293         [(159, 4), (161, 4), bassmusicmaker_three],
1294         [(165, 4), (168, 4), bassmusicmaker_three],
1295         [(170, 4), (172, 4), bassmusicmaker_three],
1296         [(172, 4), (174, 4), bassmusicmaker_three],

```

```

1297     [(177, 4), (179, 4), bassmusicmaker_three],
1298     [(183, 4), (186, 4), bassmusicmaker_three],
1299     [(186, 4), (190, 4), bassmusicmaker_three],
1300 ]
1301 ])
1302
1303 all_timespan_lists = {
1304     'Voice 1': voice_1_timespan_list,
1305     'Voice 2': voice_2_timespan_list,
1306     'Voice 3': voice_3_timespan_list,
1307     'Voice 4': voice_4_timespan_list,
1308     'Voice 5': voice_5_timespan_list,
1309     'Voice 6': voice_6_timespan_list,
1310     'Voice 7': voice_7_timespan_list,
1311     'Voice 8': voice_8_timespan_list,
1312     'Voice 9': voice_9_timespan_list,
1313     'Voice 10': voice_10_timespan_list,
1314     'Voice 11': voice_11_timespan_list,
1315     'Voice 12': voice_12_timespan_list,
1316 }
1317
1318 global_timespan = abjad.Timespan(
1319     start_offset=0,
1320     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
1321 )
1322
1323 for voice_name, timespan_list in all_timespan_lists.items():
1324     silences = abjad.TimespanList([global_timespan])
1325     silences.extend(timespan_list)
1326     silences.sort()
1327     silences.compute_logical_xor()
1328     for silence_timespan in silences:
1329         timespan_list.append(
1330             abjad.AnnotatedTimespan(
1331                 start_offset=silence_timespan.start_offset,
1332                 stop_offset=silence_timespan.stop_offset,
1333                 annotation=MusicSpecifier(
1334                     music_maker=None,
1335                     voice_name=voice_name,
1336                 ),
1337             )
1338         )
1339     timespan_list.sort()
1340
1341 for voice_name, timespan_list in all_timespan_lists.items():
1342     shards = timespan_list.split_at_offsets(bounds)
1343     split_timespan_list = abjad.TimespanList()
1344     for shard in shards:
1345         split_timespan_list.extend(shard)
1346     split_timespan_list.sort()
1347     all_timespan_lists[voice_name] = timespan_list
1348
1349 score = abjad.Score([
1350     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 1')

```

```

1351 ,
1352     abjad.StaffGroup(
1353         [
1354             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
1355             lilypond_type='Staff',),
1356             abjad.Staff([abjad.Voice(name='Voice 2')], name='Staff 2',
1357             lilypond_type='Staff',),
1358             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
1359             lilypond_type='Staff',),
1360         ],
1361         name='Staff Group 1',
1362     ),
1363     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 2')
1364 ,
1365     abjad.StaffGroup(
1366         [
1367             abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
1368             lilypond_type='Staff',),
1369             abjad.Staff([abjad.Voice(name='Voice 5')], name='Staff 5',
1370             lilypond_type='Staff',),
1371             abjad.Staff([abjad.Voice(name='Voice 6')], name='Staff 6',
1372             lilypond_type='Staff',),
1373             abjad.Staff([abjad.Voice(name='Voice 7')], name='Staff 7',
1374             lilypond_type='Staff',),
1375         ],
1376         name='Staff Group 2',
1377     ),
1378     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 3')
1379 ,
1380     abjad.StaffGroup(
1381         [
1382             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1383             lilypond_type='Staff',),
1384             abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1385             lilypond_type='Staff',),
1386             abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1387             lilypond_type='Staff',),
1388             abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1389             lilypond_type='Staff',),
1390             abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1391             lilypond_type='Staff',),
1392         ],
1393         name='Staff Group 3',
1394     )
1395 ],
1396 )
1397
1398 for time_signature in time_signatures:
1399     skip = abjad.Skip(1, multiplier=(time_signature))
1400     abjad.attach(time_signature, skip)
1401     score['Global Context 1'].append(skip)
1402
1403 for time_signature in time_signatures:
1404     skip = abjad.Skip(1, multiplier=(time_signature))

```

```

1390     abjad.attach(time_signature, skip)
1391     score['Global Context 2'].append(skip)
1392
1393 for time_signature in time_signatures:
1394     skip = abjad.Skip(1, multiplier=(time_signature))
1395     abjad.attach(time_signature, skip)
1396     score['Global Context 3'].append(skip)
1397
1398 print('Making containers ...')
1399
1400 def make_container(music_maker, durations):
1401     selections = music_maker(durations)
1402     container = abjad.Container([])
1403     container.extend(selections)
1404     return container
1405
1406 def key_function(timespan):
1407     return timespan.annotation.music_maker or silence_maker
1408
1409 for voice_name, timespan_list in all_timespan_lists.items():
1410     for music_maker, grouper in itertools.groupby(
1411         timespan_list,
1412         key=key_function,
1413     ):
1414         durations = [timespan.duration for timespan in grouper]
1415         container = make_container(music_maker, durations)
1416         voice = score[voice_name]
1417         voice.append(container)
1418
1419 print('Splitting and rewriting ...')
1420
1421 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
1422     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1423         time_signature = time_signatures[i]
1424         abjad.mutate(shard).rewrite_meter(time_signature)
1425
1426 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
1427     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1428         time_signature = time_signatures[i]
1429         abjad.mutate(shard).rewrite_meter(time_signature)
1430
1431 for voice in abjad.iterate(score['Staff Group 3']).components(abjad.Voice):
1432     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1433         time_signature = time_signatures[i]
1434         abjad.mutate(shard).rewrite_meter(time_signature)
1435
1436 print('Beaming runs ...')
1437
1438 for voice in abjad.select(score).components(abjad.Voice):
1439     for run in abjad.select(voice).runs():
1440         if 1 < len(run):
1441             specifier = abjadext.rmakers.BeamSpecifier(
1442                 beam_each_division=False,
1443             )

```

```

1444     specifier(run)
1445     abjad.attach(abjad.StartBeam(), run[0])
1446     abjad.attach(abjad.StopBeam(), run[-1])
1447     for leaf in run:
1448         if abjad.Duration(1, 4) <= leaf.written_duration:
1449             continue
1450         previous_leaf = abjad.inspect(leaf).leaf(-1)
1451         next_leaf = abjad.inspect(leaf).leaf(1)
1452         if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
1453             abjad.Duration(1, 4) <= next_leaf.written_duration):
1454             left = previous_leaf.written_duration.flag_count
1455             right = leaf.written_duration.flag_count
1456             beam_count = abjad.BeamCount(
1457                 left=left,
1458                 right=right,
1459             )
1460             abjad.attach(beam_count, leaf)
1461             continue
1462         if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
1463             abjad.Duration(1, 4) <= previous_leaf.written_duration):
1464             left = leaf.written_duration.flag_count
1465             right = next_leaf.written_duration.flag_count
1466             beam_count = abjad.BeamCount(
1467                 left=left,
1468                 right=right,
1469             )
1470             abjad.attach(beam_count, leaf)
1471
1472 print('Beautifying score ...')
1473 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1474     for selection in abjad.select(staff).components(abjad.Rest).
1475         group_by_contiguity():
1476             start_command = abjad.LilyPondLiteral(
1477                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1478                 startStaff',
1479                 format_slot='before',
1480             )
1481             stop_command = abjad.LilyPondLiteral(
1482                 r'\stopStaff \startStaff',
1483                 format_slot='after',
1484             )
1485             abjad.attach(start_command, selection[0])
1486             abjad.attach(stop_command, selection[-1])
1487
1488 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1489     for selection in abjad.select(staff).components(abjad.Rest).
1490         group_by_contiguity():
1491             start_command = abjad.LilyPondLiteral(
1492                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1493                 startStaff',
1494                 format_slot='before',
1495             )
1496             stop_command = abjad.LilyPondLiteral(
1497                 r'\stopStaff \startStaff',

```

```

1494         format_slot='after',
1495     )
1496     abjad.attach(start_command, selection[0])
1497     abjad.attach(stop_command, selection[-1])
1498
1499 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1500     for selection in abjad.select(staff).components(abjad.Rest).
1501         group_by_contiguity():
1502             start_command = abjad.LilyPondLiteral(
1503                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1504                 startStaff',
1505                 format_slot='before',
1506             )
1507             stop_command = abjad.LilyPondLiteral(
1508                 r'\stopStaff \startStaff',
1509                 format_slot='after',
1510             )
1511             abjad.attach(start_command, selection[0])
1512             abjad.attach(stop_command, selection[-1])
1513
1514 print('Stopping Hairpins ...')
1515 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1516     for rest in abjad.iterate(staff).components(abjad.Rest):
1517         previous_leaf = abjad.inspect(rest).leaf(-1)
1518         if isinstance(previous_leaf, abjad.Note):
1519             abjad.attach(abjad.StopHairpin(), rest)
1520         elif isinstance(previous_leaf, abjad.Chord):
1521             abjad.attach(abjad.StopHairpin(), rest)
1522         elif isinstance(previous_leaf, abjad.Rest):
1523             pass
1524
1525 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1526     for rest in abjad.iterate(staff).components(abjad.Rest):
1527         previous_leaf = abjad.inspect(rest).leaf(-1)
1528         if isinstance(previous_leaf, abjad.Note):
1529             abjad.attach(abjad.StopHairpin(), rest)
1530         elif isinstance(previous_leaf, abjad.Chord):
1531             abjad.attach(abjad.StopHairpin(), rest)
1532         elif isinstance(previous_leaf, abjad.Rest):
1533             pass
1534
1535 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1536     for rest in abjad.iterate(staff).components(abjad.Rest):
1537         previous_leaf = abjad.inspect(rest).leaf(-1)
1538         if isinstance(previous_leaf, abjad.Note):
1539             abjad.attach(abjad.StopHairpin(), rest)
1540         elif isinstance(previous_leaf, abjad.Chord):
1541             abjad.attach(abjad.StopHairpin(), rest)
1542         elif isinstance(previous_leaf, abjad.Rest):
1543             pass
1544
1545 print('Adding pitch material ...')
def cyc(lst):
    count = 0

```

```

1546     while True:
1547         yield lst[count%len(lst)]
1548         count += 1
1549
1550     print('Adding attachments ...')
1551     bar_line = abjad.BarLine('|||')
1552     metro = abjad.MetronomeMark((1, 4), 60)
1553     markup1 = abjad.Markup(r'\bold { M }')
1554     markup2 = abjad.Markup(r'\bold { N }')
1555     markup3 = abjad.Markup(r'\bold { O }')
1556     markup4 = abjad.Markup(r'\bold { P }')
1557     markup5 = abjad.Markup(r'\bold { Q }')
1558     markup6 = abjad.Markup(r'\bold { R }')
1559     mark1 = abjad.RehearsalMark(markup=markup1)
1560     mark2 = abjad.RehearsalMark(markup=markup2)
1561     mark3 = abjad.RehearsalMark(markup=markup3)
1562     mark4 = abjad.RehearsalMark(markup=markup4)
1563     mark5 = abjad.RehearsalMark(markup=markup5)
1564     mark6 = abjad.RehearsalMark(markup=markup6)
1565
1566     instruments1 = cyc([
1567         abjad.Flute(),
1568         abjad.ClarinetInBFlat(),
1569         abjad.Bassoon(),
1570     ])
1571
1572     instruments2 = cyc([
1573         abjad.FrenchHorn(),
1574         abjad.Trumpet(),
1575         abjad.TenorTrombone(),
1576         abjad.Tuba(),
1577     ])
1578
1579     instruments3 = cyc([
1580         abjad.Violin(),
1581         abjad.Violin(),
1582         abjad.Viola(),
1583         abjad.Cello(),
1584         abjad.Contrabass(),
1585     ])
1586
1587     clefs1 = cyc([
1588         abjad.Clef('treble'),
1589         abjad.Clef('treble'),
1590         abjad.Clef('bass'),
1591     ])
1592
1593     clefs2 = cyc([
1594         abjad.Clef('treble'),
1595         abjad.Clef('treble'),
1596         abjad.Clef('bass'),
1597         abjad.Clef('bass'),
1598     ])
1599

```

```

1600 clefs3 = cyc([
1601     abjad.Clef('treble'),
1602     abjad.Clef('treble'),
1603     abjad.Clef('alto'),
1604     abjad.Clef('bass'),
1605     abjad.Clef('bass'),
1606 ])
1607
1608 abbreviations1 = cyc([
1609     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),
1610     abjad.MarginMarkup(markup=abjad.Markup('cl.'),),
1611     abjad.MarginMarkup(markup=abjad.Markup('bssn.'),),
1612 ])
1613
1614 abbreviations2 = cyc([
1615     abjad.MarginMarkup(markup=abjad.Markup('hr.'),),
1616     abjad.MarginMarkup(markup=abjad.Markup('trp.'),),
1617     abjad.MarginMarkup(markup=abjad.Markup('trmb.'),),
1618     abjad.MarginMarkup(markup=abjad.Markup('tb.'),),
1619 ])
1620
1621 abbreviations3 = cyc([
1622     abjad.MarginMarkup(markup=abjad.Markup('vln.I'),),
1623     abjad.MarginMarkup(markup=abjad.Markup('vln.II'),),
1624     abjad.MarginMarkup(markup=abjad.Markup('vla.'),),
1625     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
1626     abjad.MarginMarkup(markup=abjad.Markup('cb.'),),
1627 ])
1628
1629 names1 = cyc([
1630     abjad.StartMarkup(markup=abjad.Markup('Flute'),),
1631     abjad.StartMarkup(markup=abjad.Markup('Clarinet'),),
1632     abjad.StartMarkup(markup=abjad.Markup('Bassoon'),),
1633 ])
1634
1635 names2 = cyc([
1636     abjad.StartMarkup(markup=abjad.Markup('Horn'),),
1637     abjad.StartMarkup(markup=abjad.Markup('Trumpet'),),
1638     abjad.StartMarkup(markup=abjad.Markup('Trombone'),),
1639     abjad.StartMarkup(markup=abjad.Markup('Tuba'),),
1640 ])
1641
1642 names3 = cyc([
1643     abjad.StartMarkup(markup=abjad.Markup('Violin I'),),
1644     abjad.StartMarkup(markup=abjad.Markup('Violin II'),),
1645     abjad.StartMarkup(markup=abjad.Markup('Viola'),),
1646     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
1647     abjad.StartMarkup(markup=abjad.Markup('Contrabass'),),
1648 ])
1649 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1650     leaf1 = abjad.select(staff).leaves()[0]
1651     abjad.attach(next(instruments1), leaf1)
1652     abjad.attach(next(abbreviations1), leaf1)

```

```

1654 abjad.attach(next(names1), leaf1)
1655 abjad.attach(next(clefs1), leaf1)
1656
1657 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1658     leaf1 = abjad.select(staff).leaves()[0]
1659     abjad.attach(next(instruments2), leaf1)
1660     abjad.attach(next(abbreviations2), leaf1)
1661     abjad.attach(next(names2), leaf1)
1662     abjad.attach(next(clefs2), leaf1)
1663
1664 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1665     leaf1 = abjad.select(staff).leaves()[0]
1666     abjad.attach(next(instruments3), leaf1)
1667     abjad.attach(next(abbreviations3), leaf1)
1668     abjad.attach(next(names3), leaf1)
1669     abjad.attach(next(clefs3), leaf1)
1670
1671 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
1672     leaf1 = abjad.select(staff).leaves()[0]
1673     last_leaf = abjad.select(staff).leaves()[-1]
1674     abjad.attach(metro, leaf1)
1675     abjad.attach(bar_line, last_leaf)
1676
1677 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff)[0]:
1678     leaf1 = abjad.select(staff).leaves()[0]
1679     last_leaf = abjad.select(staff).leaves()[-1]
1680     abjad.attach(metro, leaf1)
1681     abjad.attach(bar_line, last_leaf)
1682
1683 for staff in abjad.select(score['Staff Group 3']).components(abjad.Staff)[0]:
1684     leaf1 = abjad.select(staff).leaves()[0]
1685     last_leaf = abjad.select(staff).leaves()[-1]
1686     abjad.attach(metro, leaf1)
1687     abjad.attach(bar_line, last_leaf)
1688
1689 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1690     leaf1 = abjad.select(staff).leaves()[7]
1691     abjad.attach(mark1, leaf1)
1692
1693 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1694     leaf1 = abjad.select(staff).leaves()[7]
1695     abjad.attach(mark1, leaf1)
1696
1697 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1698     leaf1 = abjad.select(staff).leaves()[7]
1699     abjad.attach(mark1, leaf1)
1700
1701 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1702     leaf2 = abjad.select(staff).leaves()[16]
1703     abjad.attach(mark2, leaf2)
1704
1705 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1706     leaf2 = abjad.select(staff).leaves()[16]
1707     abjad.attach(mark2, leaf2)

```

```

1708
1709 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1710     leaf2 = abjad.select(staff).leaves()[16]
1711     abjad.attach(mark2, leaf2)
1712
1713 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1714     leaf3 = abjad.select(staff).leaves()[22]
1715     abjad.attach(mark3, leaf3)
1716
1717 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1718     leaf3 = abjad.select(staff).leaves()[22]
1719     abjad.attach(mark3, leaf3)
1720
1721 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1722     leaf3 = abjad.select(staff).leaves()[22]
1723     abjad.attach(mark3, leaf3)
1724
1725 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1726     leaf4 = abjad.select(staff).leaves()[29]
1727     abjad.attach(mark4, leaf4)
1728
1729 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1730     leaf4 = abjad.select(staff).leaves()[29]
1731     abjad.attach(mark4, leaf4)
1732
1733 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1734     leaf4 = abjad.select(staff).leaves()[29]
1735     abjad.attach(mark4, leaf4)
1736
1737 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1738     leaf5 = abjad.select(staff).leaves()[34]
1739     abjad.attach(mark5, leaf5)
1740
1741 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1742     leaf5 = abjad.select(staff).leaves()[34]
1743     abjad.attach(mark5, leaf5)
1744
1745 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1746     leaf5 = abjad.select(staff).leaves()[34]
1747     abjad.attach(mark5, leaf5)
1748
1749 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1750     leaf6 = abjad.select(staff).leaves()[39]
1751     abjad.attach(mark6, leaf6)
1752
1753 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1754     leaf6 = abjad.select(staff).leaves()[39]
1755     abjad.attach(mark6, leaf6)
1756
1757 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1758     leaf6 = abjad.select(staff).leaves()[39]
1759     abjad.attach(mark6, leaf6)
1760
1761 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

```

```

1762     abjad.Instrument.transpose_from_sounding_pitch(staff)
1763
1764 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1765     abjad.Instrument.transpose_from_sounding_pitch(staff)
1766
1767 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1768     abjad.Instrument.transpose_from_sounding_pitch(staff)
1769
1770 score_file = abjad.LilyPondFile.new(
1771     score,
1772     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
1773     _stylesheets/abjad.ily'],
1774 )
1775
1776 abjad.SegmentMaker.comment_measure_numbers(score)
1777 ######
1778 directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/Segment_III'
1779 pdf_path = f'{directory}/Segment_III.pdf'
1780 path = pathlib.Path('Segment_III.pdf')
1781 if path.exists():
1782     print(f'Removing {pdf_path} ...')
1783     path.unlink()
1784 time_1 = time.time()
1785 print(f'Persisting {pdf_path} ...')
1786 result = abjad.persist(score_file).as_pdf(pdf_path)
1787 print(result[0])
1788 print(result[1])
1789 print(result[2])
1790 success = result[3]
1791 if success is False:
1792     print('LilyPond failed!')
1793 time_2 = time.time()
1794 total_time = time_2 - time_1
1795 print(f'Total time: {total_time} seconds')
1796 if path.exists():
1797     print(f'Opening {pdf_path} ...')
1798     os.system(f'open {pdf_path}')

```

**Code Example A.13:** Tianshu Segment\_III

#### A.3.1.4 SEGMENT\_IV

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed

```

```

11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (5, 4), (2, 4), (4, 4), (3, 4), (4, 4), (4, 4),
17         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
18         (4, 4), (4, 4), (5, 4), (5, 4), (3, 4), (3, 4),
19         (2, 4), (3, 4), (4, 4), (3, 4), (4, 4), (3, 4),
20         (5, 4), (3, 4), (3, 4), (4, 4), (3, 4), (3, 4),
21         (4, 4), (5, 4), (4, 4), (3, 4), (5, 4), (5, 4),
22         (5, 4), (5, 4), (4, 4), (4, 4), (5, 4), (5, 4),
23         (4, 4), (4, 4), (3, 4), (4, 4), (4, 4), (3, 4),
24         (9, 8),
25     ],
26 ]
27
28 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures])
29
30 def reduceMod3(rw):
31     return [(x % 4) for x in rw]
32
33 def reduceMod5(rw):
34     return [(x % 6) for x in rw]
35
36 def reduceMod7(rw):
37     return [(x % 8) for x in rw]
38
39 def reduceMod9(rw):
40     return [(x % 10) for x in rw]
41
42 def reduceMod13(rw):
43     return [(x % 14) for x in rw]
44
45 seed(1)
46 flute_random_walk_one = []
47 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
48 for i in range(1, 1000):
49     movement = -1 if random() < 0.5 else 1
50     value = flute_random_walk_one[i-1] + movement
51     flute_random_walk_one.append(value)
52 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
53 flute_chord_one = [2, 12, 18, 20, 25, 20, 18, 12, ]
54 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
55     flute_random_walk_one)]
56
56 seed(2)
57 clarinet_random_walk_one = []
58 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
59 for i in range(1, 1000):
60     movement = -1 if random() < 0.5 else 1
61     value = clarinet_random_walk_one[i-1] + movement
62     clarinet_random_walk_one.append(value)

```

```

63 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
64 clarinet_chord_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]
65 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod7(
66     clarinet_random_walk_one)]
67
68 seed(3)
69 bassoon_random_walk_one = []
70 bassoon_random_walk_one.append(-1 if random() < 0.5 else 1)
71 for i in range(1, 1000):
72     movement = -1 if random() < 0.5 else 1
73     value = bassoon_random_walk_one[i-1] + movement
74     bassoon_random_walk_one.append(value)
75 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
76 bassoon_chord_one = [-19, -8, -5, 2, 12, 2, -5, -8, ]
77 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod7(
78     bassoon_random_walk_one)]
79
80 seed(4)
81 horn_random_walk_one = []
82 horn_random_walk_one.append(-1 if random() < 0.5 else 1)
83 for i in range(1, 1000):
84     movement = -1 if random() < 0.5 else 1
85     value = horn_random_walk_one[i-1] + movement
86     horn_random_walk_one.append(value)
87 horn_random_walk_one = [abs(x) for x in horn_random_walk_one]
88 horn_chord_one = [-19, -8, -5, 2, 12, 18, 12, 2, -5, -8, ]
89 horn_notes_one = [horn_chord_one[x] for x in reduceMod9(horn_random_walk_one)]
90
91 seed(5)
92 trumpet_random_walk_one = []
93 trumpet_random_walk_one.append(-1 if random() < 0.5 else 1)
94 for i in range(1, 1000):
95     movement = -1 if random() < 0.5 else 1
96     value = trumpet_random_walk_one[i-1] + movement
97     trumpet_random_walk_one.append(value)
98 trumpet_random_walk_one = [abs(x) for x in trumpet_random_walk_one]
99 trumpet_chord_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]
100 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod7(
101     trumpet_random_walk_one)]
102
103 seed(6)
104 trombone_random_walk_one = []
105 trombone_random_walk_one.append(-1 if random() < 0.5 else 1)
106 for i in range(1, 1000):
107     movement = -1 if random() < 0.5 else 1
108     value = trombone_random_walk_one[i-1] + movement
109     trombone_random_walk_one.append(value)
110 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
111 trombone_chord_one = [-19, -8, -5, 2, -5, -8, ]
112 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod5(
113     trombone_random_walk_one)]
114
115 seed(7)
116 tuba_random_walk_one = []

```

```

113 tuba_random_walk_one.append(-1 if random() < 0.5 else 1)
114 for i in range(1, 1000):
115     movement = -1 if random() < 0.5 else 1
116     value = tuba_random_walk_one[i-1] + movement
117     tuba_random_walk_one.append(value)
118 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
119 tuba_chord_one = [-27, -19, -8, -5, 2, -5, -8, -19, ]
120 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod7(tuba_random_walk_one)]
121
122 seed(8)
123 violin1_random_walk_one = []
124 violin1_random_walk_one.append(-1 if random() < 0.5 else 1)
125 for i in range(1, 1000):
126     movement = -1 if random() < 0.5 else 1
127     value = violin1_random_walk_one[i-1] + movement
128     violin1_random_walk_one.append(value)
129 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
130 violin1_chord_one = [-5, 2, 12, 18, 20, 25, 34, 35, 34, 25, 20, 18, 12, 2, ]
131 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod13(
132     violin1_random_walk_one)]
133
134 seed(9)
135 violin2_random_walk_one = []
136 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
137 for i in range(1, 1000):
138     movement = -1 if random() < 0.5 else 1
139     value = violin2_random_walk_one[i-1] + movement
140     violin2_random_walk_one.append(value)
141 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
142 violin2_chord_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]
143 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod7(
144     violin2_random_walk_one)]
145
146 seed(10)
147 viola_random_walk_one = []
148 viola_random_walk_one.append(-1 if random() < 0.5 else 1)
149 for i in range(1, 1000):
150     movement = -1 if random() < 0.5 else 1
151     value = viola_random_walk_one[i-1] + movement
152     viola_random_walk_one.append(value)
153 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
154 viola_chord_one = [-8, -5, 2, 12, 18, 12, 2, -5, ]
155 viola_notes_one = [viola_chord_one[x] for x in reduceMod7(
156     viola_random_walk_one)]
157
158 seed(11)
159 cello_random_walk_one = []
160 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
161 for i in range(1, 1000):
162     movement = -1 if random() < 0.5 else 1
163     value = cello_random_walk_one[i-1] + movement
164     cello_random_walk_one.append(value)
165 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
166 cello_chord_one = [-19, -8, -5, 2, 12, 2, -5, -8]

```

```

164 cello_notes_one = [cello_chord_one[x] for x in reduceMod7(
165     cello_random_walk_one)]
166
166 seed(12)
167 bass_random_walk_one = []
168 bass_random_walk_one.append(-1 if random() < 0.5 else 1)
169 for i in range(1, 1000):
170     movement = -1 if random() < 0.5 else 1
171     value = bass_random_walk_one[i-1] + movement
172     bass_random_walk_one.append(value)
173 bass_random_walk_one = [abs(x) for x in bass_random_walk_one]
174 bass_chord_one = [-27, -19, -8, -5, 2, -5, -8, -19, ]
175 bass_notes_one = [bass_chord_one[x] for x in reduceMod7(bass_random_walk_one)]
176
177 flute_scale = [39, ]
178 clarinet_scale = [18, ]
179 bassoon_scale = [12, ]
180 horn_scale = [-5, ]
181 trumpet_scale = [12, ]
182 trombone_scale = [-5, ]
183 tuba_scale = [-27, ]
184 violin1_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25, 24.5,
185     24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5, 20, 20.5, 21,
186     21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26, 26.5, 27, 27.5, 28,
187     28.5, 29, 29.5, ]
188 violin2_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14, 13.5,
189     13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5, 10, 10.5,
190     11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16, 16.5, 17, 17.5, 18,
191     18.5, ]
192 viola_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1, 0.5,
193     0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5, 0, 0.5, 1,
194     1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]
195 cello_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8, -8.5, -9,
196     -9.5, -10, -10.5, -11, -11.5, -12, -12.5, -13, -13.5, -14, -13.5, -13,
197     -12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8, -7.5, -7, -6.5,
198     -6, -5.5, -5, -4.5, -4, -3.5, ]
199 bass_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18, -18.5, -19,
200     -19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5, -24, -24.5, -25,
201     -24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21, -20.5, -20, -19.5, -19,
202     -18.5, -18, -17.5, -17, -16.5, -16, -15.5, -15, -14.5, ]
203
203 seed(1)
204 flute_random_walk_two = []
205 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
206 for i in range(1, 1000):
207     movement = -1 if random() < 0.5 else 1
208     value = flute_random_walk_two[i-1] + movement
209     flute_random_walk_two.append(value)
210 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
211 flute_chord_two = [4, 12, 18, 22, 23, 22, 18, 12, ]
212 flute_notes_two = [flute_chord_two[x] for x in reduceMod7(
213     flute_random_walk_two)]
214
214 seed(2)

```

```

202 clarinet_random_walk_two = []
203 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
204 for i in range(1, 1000):
205     movement = -1 if random() < 0.5 else 1
206     value = clarinet_random_walk_two[i-1] + movement
207     clarinet_random_walk_two.append(value)
208 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
209 clarinet_chord_two = [-10, -7, 4, 12, 18, 22, 18, 12, 4, -7, ]
210 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod9(
211     clarinet_random_walk_two)]
212
213 seed(3)
214 bassoon_random_walk_two = []
215 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
216 for i in range(1, 1000):
217     movement = -1 if random() < 0.5 else 1
218     value = bassoon_random_walk_two[i-1] + movement
219     bassoon_random_walk_two.append(value)
220 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
221 bassoon_chord_two = [-17, -10, -7, 4, 12, 4, -7, -10, ]
222 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod7(
223     bassoon_random_walk_two)]
224
225 seed(4)
226 horn_random_walk_two = []
227 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
228 for i in range(1, 1000):
229     movement = -1 if random() < 0.5 else 1
230     value = horn_random_walk_two[i-1] + movement
231     horn_random_walk_two.append(value)
232 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
233 horn_chord_two = [-17, -10, -7, 4, -7, -10, ]
234 horn_notes_two = [horn_chord_two[x] for x in reduceMod5(horn_random_walk_two)]
235
236 seed(5)
237 trumpet_random_walk_two = []
238 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)
239 for i in range(1, 1000):
240     movement = -1 if random() < 0.5 else 1
241     value = trumpet_random_walk_two[i-1] + movement
242     trumpet_random_walk_two.append(value)
243 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]
244 trumpet_chord_two = [4, 12, 18, 12, ]
245 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod3(
246     trumpet_random_walk_two)]
247
248 seed(6)
249 trombone_random_walk_two = []
250 trombone_random_walk_two.append(-1 if random() < 0.5 else 1)
251 for i in range(1, 1000):
252     movement = -1 if random() < 0.5 else 1
253     value = trombone_random_walk_two[i-1] + movement
254     trombone_random_walk_two.append(value)
255 trombone_random_walk_two = [abs(x) for x in trombone_random_walk_two]
```

```

253 trombone_chord_two = [-17, -10, -7, 4, -7, -10, ]
254 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod5(
255     trombone_random_walk_two)]
256
257 seed(7)
258 tuba_random_walk_two = []
259 tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
260 for i in range(1, 1000):
261     movement = -1 if random() < 0.5 else 1
262     value = tuba_random_walk_two[i-1] + movement
263     tuba_random_walk_two.append(value)
264 tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
265 tuba_chord_two = [-27, -17, -10, -7, 4, -7, -10, -17, ]
266 tuba_notes_two = [tuba_chord_two[x] for x in reduceMod7(tuba_random_walk_two)]
267
268 seed(8)
269 violin1_random_walk_two = []
270 violin1_random_walk_two.append(-1 if random() < 0.5 else 1)
271 for i in range(1, 1000):
272     movement = -1 if random() < 0.5 else 1
273     value = violin1_random_walk_two[i-1] + movement
274     violin1_random_walk_two.append(value)
275 violin1_random_walk_two = [abs(x) for x in violin1_random_walk_two]
276 violin1_chord_two = [4, 12, 18, 22, 23, 32, 37, 39, 37, 32, 23, 22, 18, 12, ]
277 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod13(
278     violin1_random_walk_two)]
279
280 seed(9)
281 violin2_random_walk_two = []
282 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
283 for i in range(1, 1000):
284     movement = -1 if random() < 0.5 else 1
285     value = violin2_random_walk_two[i-1] + movement
286     violin2_random_walk_two.append(value)
287 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
288 violin2_chord_two = [4, 12, 18, 22, 23, 32, 23, 22, 18, 12, ]
289 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod9(
290     violin2_random_walk_two)]
291
292 seed(10)
293 viola_random_walk_two = []
294 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
295 for i in range(1, 1000):
296     movement = -1 if random() < 0.5 else 1
297     value = viola_random_walk_two[i-1] + movement
298     viola_random_walk_two.append(value)
299 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
300 viola_chord_two = [-10, -7, 4, 12, 18, 12, 4, -7, ]
301 viola_notes_two = [viola_chord_two[x] for x in reduceMod7(
302     viola_random_walk_two)]
303
304 seed(11)
305 cello_random_walk_two = []
306 cello_random_walk_two.append(-1 if random() < 0.5 else 1)

```

```

303 for i in range(1, 1000):
304     movement = -1 if random() < 0.5 else 1
305     value = cello_random_walk_two[i-1] + movement
306     cello_random_walk_two.append(value)
307 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
308 cello_chord_two = [-17, -10, -7, 4, 12, 4, -7, -10, ]
309 cello_notes_two = [cello_chord_two[x] for x in reduceMod7(
310     cello_random_walk_two)]
311
312 seed(12)
313 bass_random_walk_two = []
314 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
315 for i in range(1, 1000):
316     movement = -1 if random() < 0.5 else 1
317     value = bass_random_walk_two[i-1] + movement
318     bass_random_walk_two.append(value)
319 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
320 bass_chord_two = [-27, -17, -10, -7, 4, -7, -10, -17, ]
321 bass_notes_two = [bass_chord_two[x] for x in reduceMod7(bass_random_walk_two)]
322
323 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
324     talea=abjadext.rmakers.Talea(
325         counts=[2, 3, 2, 1, 4, 3, 1, 4, 5, 1],
326         denominator=8,
327         ),
328     beamSpecifier=abjadext.rmakers.BeamSpecifier(
329         beamDivisionsTogether=True,
330         beamRests=False,
331         ),
332     extraCountsPerDivision=[1, 1, 0, -1, 0],
333     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
334         leftClasses=[abjad.Note, abjad.Rest],
335         leftCounts=[0, 1, 1],
336         ),
337     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
338         trivialize=True,
339         extractTrivial=True,
340         rewriteRestFilled=True,
341         ),
342 )
343
344 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
345     talea=abjadext.rmakers.Talea(
346         counts=[2, 1, 3, 1, 4, 1, 1, 5],
347         denominator=16,
348         ),
349     beamSpecifier=abjadext.rmakers.BeamSpecifier(
350         beamDivisionsTogether=True,
351         beamRests=False,
352         ),
353     extraCountsPerDivision=[0, 1, 0, -1],
354     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
355         leftClasses=[abjad.Note, abjad.Rest],
356         leftCounts=[1, 1, 0, 0],
357         ),
358     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
359         trivialize=True,
360         extractTrivial=True,
361         rewriteRestFilled=True,
362         ),
363 )

```

```

356     right_classes=[abjad.Note, abjad.Rest],
357     right_counts=[1, 0, 0, 1],
358   ),
359   tupletSpecifier=abjadext.rmakers.TupletSpecifier(
360     trivialize=True,
361     extract_trivial=True,
362     rewrite_rest_filled=True,
363   ),
364 )
365
366 rmaker_three = abjadext.rmakers.TaleaRhythmMaker(
367   talea=abjadext.rmakers.Talea(
368     counts=[1, 2, 1, 3, 1, 4, 5, 1, 1],
369     denominator=16,
370   ),
371   beamSpecifier=abjadext.rmakers.BeamSpecifier(
372     beam_divisions_together=True,
373     beam_rests=False,
374   ),
375   extra_counts_per_division=[0, 1, 0, -1],
376   burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
377     left_classes=[abjad.Note, abjad.Rest],
378     left_counts=[1, 0, 1],
379   ),
380   tupletSpecifier=abjadext.rmakers.TupletSpecifier(
381     trivialize=True,
382     extract_trivial=True,
383     rewrite_rest_filled=True,
384   ),
385 )
386
387
388 attachment_handler_one = AttachmentHandler(
389   starting_dynamic='mf',
390   ending_dynamic='ff',
391   hairpin_indicator='<',
392   articulation='''',
393 )
394
395 attachment_handler_two = AttachmentHandler(
396   starting_dynamic='ff',
397   ending_dynamic='mf',
398   hairpin_indicator='>',
399   articulation='''',
400 )
401
402 attachment_handler_three = AttachmentHandler(
403   starting_dynamic='p',
404   ending_dynamic='pp',
405   hairpin_indicator='--',
406   articulation='tenuto',
407 )
408
409 #####oboe#####

```

```
410 flutemusicmaker_one = MusicMaker(  
411     rmaker=rmaker_one,  
412     pitches=flute_scale,  
413     continuous=True,  
414     attachment_handler=attachment_handler_one,  
415 )  
416 flutemusicmaker_two = MusicMaker(  
417     rmaker=rmaker_two,  
418     pitches=flute_notes_two,  
419     continuous=True,  
420     attachment_handler=attachment_handler_two,  
421 )  
422 flutemusicmaker_three = MusicMaker(  
423     rmaker=rmaker_three,  
424     pitches=flute_notes_one,  
425     continuous=True,  
426     attachment_handler=attachment_handler_three,  
427 )  
428 #####violin1#####  
429 violin1musicmaker_one = MusicMaker(  
430     rmaker=rmaker_one,  
431     pitches=violin1_scale,  
432     continuous=True,  
433     attachment_handler=attachment_handler_one,  
434 )  
435 violin1musicmaker_two = MusicMaker(  
436     rmaker=rmaker_two,  
437     pitches=violin1_notes_two,  
438     continuous=True,  
439     attachment_handler=attachment_handler_two,  
440 )  
441 violin1musicmaker_three = MusicMaker(  
442     rmaker=rmaker_three,  
443     pitches=violin1_notes_one,  
444     continuous=True,  
445     attachment_handler=attachment_handler_three,  
446 )  
447 #####trumpet#####  
448 trumpetmusicmaker_one = MusicMaker(  
449     rmaker=rmaker_one,  
450     pitches=trumpet_scale,  
451     continuous=True,  
452     attachment_handler=attachment_handler_one,  
453 )  
454 trumpetmusicmaker_two = MusicMaker(  
455     rmaker=rmaker_two,  
456     pitches=trumpet_notes_two,  
457     continuous=True,  
458     attachment_handler=attachment_handler_two,  
459 )  
460 trumpetmusicmaker_three = MusicMaker(  
461     rmaker=rmaker_three,  
462     pitches=trumpet_notes_one,  
463     continuous=True,
```

```

464     attachment_handler=attachment_handler_three,
465 )
466 #####clarinet#####
467 clarinetmusicmaker_one = MusicMaker(
468     rmaker=rmaker_one,
469     pitches=clarinet_scale,
470     continuous=True,
471     attachment_handler=attachment_handler_one,
472 )
473 clarinetmusicmaker_two = MusicMaker(
474     rmaker=rmaker_two,
475     pitches=clarinet_notes_two,
476     continuous=True,
477     attachment_handler=attachment_handler_two,
478 )
479 clarinetmusicmaker_three = MusicMaker(
480     rmaker=rmaker_three,
481     pitches=clarinet_notes_one,
482     continuous=True,
483     attachment_handler=attachment_handler_three,
484 )
485 #####violin2#####
486 violin2musicmaker_one = MusicMaker(
487     rmaker=rmaker_one,
488     pitches=violin2_scale,
489     continuous=True,
490     attachment_handler=attachment_handler_one,
491 )
492 violin2musicmaker_two = MusicMaker(
493     rmaker=rmaker_two,
494     pitches=violin2_notes_two,
495     continuous=True,
496     attachment_handler=attachment_handler_two,
497 )
498 violin2musicmaker_three = MusicMaker(
499     rmaker=rmaker_three,
500     pitches=violin2_notes_one,
501     continuous=True,
502     attachment_handler=attachment_handler_three,
503 )
504 #####viola#####
505 violamusicmaker_one = MusicMaker(
506     rmaker=rmaker_one,
507     pitches=viola_scale,
508     continuous=True,
509     attachment_handler=attachment_handler_one,
510 )
511 violamusicmaker_two = MusicMaker(
512     rmaker=rmaker_two,
513     pitches=viola_notes_two,
514     continuous=True,
515     attachment_handler=attachment_handler_two,
516 )
517 violamusicmaker_three = MusicMaker(

```

```
s18     rmaker=rmaker_three,
s19     pitches=viola_notes_one,
s20     continuous=True,
s21     attachment_handler=attachment_handler_three,
s22 )
s23 #####bassoon#####
s24 bassoonmusicmaker_one = MusicMaker(
s25     rmaker=rmaker_one,
s26     pitches=bassoon_scale,
s27     continuous=True,
s28     attachment_handler=attachment_handler_one,
s29 )
s30 bassoonmusicmaker_two = MusicMaker(
s31     rmaker=rmaker_two,
s32     pitches=bassoon_notes_two,
s33     continuous=True,
s34     attachment_handler=attachment_handler_two,
s35 )
s36 bassoonmusicmaker_three = MusicMaker(
s37     rmaker=rmaker_three,
s38     pitches=bassoon_notes_one,
s39     continuous=True,
s40     attachment_handler=attachment_handler_three,
s41 )
s42 #####trombone#####
s43 trombonemusicmaker_one = MusicMaker(
s44     rmaker=rmaker_one,
s45     pitches=trombone_scale,
s46     continuous=True,
s47     attachment_handler=attachment_handler_one,
s48 )
s49 trombonemusicmaker_two = MusicMaker(
s50     rmaker=rmaker_two,
s51     pitches=trombone_notes_two,
s52     continuous=True,
s53     attachment_handler=attachment_handler_two,
s54 )
s55 trombonemusicmaker_three = MusicMaker(
s56     rmaker=rmaker_three,
s57     pitches=trombone_notes_one,
s58     continuous=True,
s59     attachment_handler=attachment_handler_three,
s60 )
s61 #####cello#####
s62 cellomusicmaker_one = MusicMaker(
s63     rmaker=rmaker_one,
s64     pitches=cello_scale,
s65     continuous=True,
s66     attachment_handler=attachment_handler_one,
s67 )
s68 cellomusicmaker_two = MusicMaker(
s69     rmaker=rmaker_two,
s70     pitches=cello_notes_two,
s71     continuous=True,
```

```

572     attachment_handler=attachment_handler_two,
573 )
574 cellomusicmaker_three = MusicMaker(
575     rmaker=rmaker_three,
576     pitches=cello_notes_one,
577     continuous=True,
578     attachment_handler=attachment_handler_three,
579 )
580 #####horn#####
581 hornmusicmaker_one = MusicMaker(
582     rmaker=rmaker_one,
583     pitches=horn_scale,
584     continuous=True,
585     attachment_handler=attachment_handler_one,
586 )
587 hornmusicmaker_two = MusicMaker(
588     rmaker=rmaker_two,
589     pitches=horn_notes_two,
590     continuous=True,
591     attachment_handler=attachment_handler_two,
592 )
593 hornmusicmaker_three = MusicMaker(
594     rmaker=rmaker_three,
595     pitches=horn_notes_one,
596     continuous=True,
597     attachment_handler=attachment_handler_three,
598 )
599 #####tuba#####
600 tubamusicmaker_one = MusicMaker(
601     rmaker=rmaker_one,
602     pitches=tuba_scale,
603     continuous=True,
604     attachment_handler=attachment_handler_one,
605 )
606 tubamusicmaker_two = MusicMaker(
607     rmaker=rmaker_two,
608     pitches=tuba_notes_two,
609     continuous=True,
610     attachment_handler=attachment_handler_two,
611 )
612 tubamusicmaker_three = MusicMaker(
613     rmaker=rmaker_three,
614     pitches=tuba_notes_one,
615     continuous=True,
616     attachment_handler=attachment_handler_three,
617 )
618 #####bass#####
619 bassmusicmaker_one = MusicMaker(
620     rmaker=rmaker_one,
621     pitches=bass_scale,
622     continuous=True,
623     attachment_handler=attachment_handler_one,
624 )
625 bassmusicmaker_two = MusicMaker(

```

```

626     rmaker=rmaker_two,
627     pitches=bass_notes_two,
628     continuous=True,
629     attachment_handler=attachment_handler_two,
630 )
631 bassmusicmaker_three = MusicMaker(
632     rmaker=rmaker_three,
633     pitches=bass_notes_one,
634     continuous=True,
635     attachment_handler=attachment_handler_three,
636 )
637
638 silence_maker = abjadext.rmakers.NoteRhythmMaker(
639     division_masks=[
640         abjadext.rmakers.SilenceMask(
641             pattern=abjad.index([0], 1),
642             ),
643         ],
644     )
645
646
647 class MusicSpecifier:
648
649     def __init__(self, music_maker, voice_name):
650         self.music_maker = music_maker
651         self.voice_name = voice_name
652
653
654     print('Collecting timespans and rmakers ...')
655     ###group one###
656     voice_1_timespan_list = abjad.TimespanList([
657         abjad.AnnotatedTimespan(
658             start_offset=start_offset,
659             stop_offset=stop_offset,
660             annotation=MusicSpecifier(
661                 music_maker=music_maker,
662                 voice_name='Voice 1',
663             ),
664         )
665         for start_offset, stop_offset, music_maker in [
666             [(9, 4), (10, 4), flutemusicmaker_one],
667             [(15, 4), (18, 4), flutemusicmaker_two],
668             [(22, 4), (25, 4), flutemusicmaker_three],
669             [(27, 4), (30, 4), flutemusicmaker_one],
670             [(30, 4), (32, 4), flutemusicmaker_one],
671             [(35, 4), (39, 4), flutemusicmaker_two],
672             [(42, 4), (43, 4), flutemusicmaker_three],
673             [(43, 4), (44, 4), flutemusicmaker_three],
674             [(45, 4), (46, 4), flutemusicmaker_one],
675             [(46, 4), (50, 4), flutemusicmaker_one],
676             [(54, 4), (57, 4), flutemusicmaker_two],
677             [(59, 4), (60, 4), flutemusicmaker_three],
678             [(65, 4), (67, 4), flutemusicmaker_one],
679             [(67, 4), (69, 4), flutemusicmaker_one],

```

```

680     [(70, 4), (72, 4), flutemusicmaker_two],
681     [(72, 4), (75, 4), flutemusicmaker_two],
682     [(76, 4), (78, 4), flutemusicmaker_three],
683     [(81, 4), (82, 4), flutemusicmaker_one],
684     [(82, 4), (85, 4), flutemusicmaker_one],
685     [(90, 4), (91, 4), flutemusicmaker_two],
686     [(93, 4), (94, 4), flutemusicmaker_three],
687     [(94, 4), (96, 4), flutemusicmaker_three],
688     [(100, 4), (104, 4), flutemusicmaker_one],
689     [(104, 4), (105, 4), flutemusicmaker_one],
690     [(106, 4), (107, 4), flutemusicmaker_two],
691     [(107, 4), (108, 4), flutemusicmaker_two],
692     [(111, 4), (114, 4), flutemusicmaker_one],
693     [(114, 4), (115, 4), flutemusicmaker_one],
694     [(116, 4), (119, 4), flutemusicmaker_one],
695     [(119, 4), (120, 4), flutemusicmaker_one],
696     [(121, 4), (123, 4), flutemusicmaker_one],
697     [(123, 4), (125, 4), flutemusicmaker_one],
698     [(126, 4), (131, 4), flutemusicmaker_two],
699     [(131, 4), (133, 4), flutemusicmaker_two],
700     [(136, 4), (141, 4), flutemusicmaker_two],
701     [(148, 4), (150, 4), flutemusicmaker_two],
702     [(150, 4), (153, 4), flutemusicmaker_three],
703     [(155, 4), (159, 4), flutemusicmaker_three],
704     [(162, 4), (164, 4), flutemusicmaker_three],
705     [(168, 4), (171, 4), flutemusicmaker_three],
706     [(173, 4), (175, 4), flutemusicmaker_three],
707     [(175, 4), (177, 4), flutemusicmaker_three],
708     [(180, 4), (182, 4), flutemusicmaker_three],
709     [(186, 4), (190, 4), flutemusicmaker_three],
710     [(190, 4), (381, 8), silence_maker],
711 ]
712 ])
713
714 voice_5_timespan_list = abjad.TimespanList([
715     abjad.AnnotatedTimespan(
716         start_offset=start_offset,
717         stop_offset=stop_offset,
718         annotation=MusicSpecifier(
719             music_maker=music_maker,
720             voice_name='Voice 5',
721         ),
722     )
723     for start_offset, stop_offset, music_maker in [
724         [(9, 4), (10, 4), trumpetmusicmaker_one],
725         [(14, 4), (18, 4), trumpetmusicmaker_two],
726         [(23, 4), (25, 4), trumpetmusicmaker_three],
727         [(27, 4), (30, 4), trumpetmusicmaker_one],
728         [(30, 4), (32, 4), trumpetmusicmaker_one],
729         [(35, 4), (39, 4), trumpetmusicmaker_two],
730         [(42, 4), (43, 4), trumpetmusicmaker_three],
731         [(43, 4), (44, 4), trumpetmusicmaker_three],
732         [(45, 4), (46, 4), trumpetmusicmaker_one],
733         [(46, 4), (50, 4), trumpetmusicmaker_one],

```

```

734     [(54, 4), (57, 4), trumpetmusicmaker_two],
735     [(59, 4), (60, 4), trumpetmusicmaker_three],
736     [(65, 4), (67, 4), trumpetmusicmaker_one],
737     [(67, 4), (69, 4), trumpetmusicmaker_one],
738     [(70, 4), (72, 4), trumpetmusicmaker_two],
739     [(72, 4), (75, 4), trumpetmusicmaker_two],
740     [(76, 4), (78, 4), trumpetmusicmaker_three],
741     [(81, 4), (82, 4), trumpetmusicmaker_one],
742     [(82, 4), (85, 4), trumpetmusicmaker_one],
743     [(90, 4), (91, 4), trumpetmusicmaker_two],
744     [(93, 4), (94, 4), trumpetmusicmaker_three],
745     [(94, 4), (96, 4), trumpetmusicmaker_three],
746     [(100, 4), (104, 4), trumpetmusicmaker_one],
747     [(104, 4), (105, 4), trumpetmusicmaker_one],
748     [(106, 4), (107, 4), trumpetmusicmaker_two],
749     [(107, 4), (108, 4), trumpetmusicmaker_two],
750     [(111, 4), (114, 4), trumpetmusicmaker_one],
751     [(114, 4), (115, 4), trumpetmusicmaker_one],
752     [(116, 4), (119, 4), trumpetmusicmaker_one],
753     [(119, 4), (120, 4), trumpetmusicmaker_one],
754     [(121, 4), (123, 4), trumpetmusicmaker_one],
755     [(123, 4), (125, 4), trumpetmusicmaker_one],
756     [(126, 4), (131, 4), trumpetmusicmaker_two],
757     [(131, 4), (133, 4), trumpetmusicmaker_two],
758     [(136, 4), (141, 4), trumpetmusicmaker_two],
759     [(148, 4), (150, 4), trumpetmusicmaker_two],
760     [(150, 4), (154, 4), trumpetmusicmaker_three],
761     [(157, 4), (159, 4), trumpetmusicmaker_three],
762     [(163, 4), (164, 4), trumpetmusicmaker_three],
763     [(164, 4), (166, 4), trumpetmusicmaker_three],
764     [(168, 4), (172, 4), trumpetmusicmaker_three],
765     [(175, 4), (177, 4), trumpetmusicmaker_three],
766     [(181, 4), (183, 4), trumpetmusicmaker_three],
767     [(183, 4), (184, 4), trumpetmusicmaker_three],
768     [(186, 4), (190, 4), trumpetmusicmaker_three],
769     ],
770   ])
771
772 voice_8_timespan_list = abjad.TimespanList([
773     abjad.AnnotatedTimespan(
774         start_offset=start_offset,
775         stop_offset=stop_offset,
776         annotation=MusicSpecifier(
777             music_maker=music_maker,
778             voice_name='Voice 8',
779         ),
780     )
781     for start_offset, stop_offset, music_maker in [
782         [(9, 4), (10, 4), violin1musicmaker_one],
783         [(14, 4), (18, 4), violin1musicmaker_two],
784         [(22, 4), (25, 4), violin1musicmaker_three],
785         [(27, 4), (30, 4), violin1musicmaker_one],
786         [(35, 4), (39, 4), violin1musicmaker_two],
787         [(42, 4), (43, 4), violin1musicmaker_three],

```

```

788     [(43, 4), (44, 4), violin1musicmaker_three],
789     [(45, 4), (46, 4), violin1musicmaker_one],
790     [(46, 4), (50, 4), violin1musicmaker_one],
791     [(54, 4), (57, 4), violin1musicmaker_two],
792     [(59, 4), (60, 4), violin1musicmaker_three],
793     [(65, 4), (67, 4), violin1musicmaker_one],
794     [(67, 4), (69, 4), violin1musicmaker_one],
795     [(70, 4), (72, 4), violin1musicmaker_two],
796     [(72, 4), (75, 4), violin1musicmaker_two],
797     [(76, 4), (78, 4), violin1musicmaker_three],
798     [(81, 4), (82, 4), violin1musicmaker_one],
799     [(82, 4), (85, 4), violin1musicmaker_one],
800     [(90, 4), (91, 4), violin1musicmaker_two],
801     [(93, 4), (94, 4), violin1musicmaker_three],
802     [(94, 4), (96, 4), violin1musicmaker_three],
803     [(100, 4), (104, 4), violin1musicmaker_one],
804     [(104, 4), (105, 4), violin1musicmaker_one],
805     [(106, 4), (107, 4), violin1musicmaker_two],
806     [(107, 4), (108, 4), violin1musicmaker_two],
807     [(111, 4), (114, 4), violin1musicmaker_one],
808     [(114, 4), (115, 4), violin1musicmaker_one],
809     [(116, 4), (119, 4), violin1musicmaker_one],
810     [(119, 4), (120, 4), violin1musicmaker_one],
811     [(121, 4), (123, 4), violin1musicmaker_one],
812     [(123, 4), (125, 4), violin1musicmaker_one],
813     [(126, 4), (131, 4), violin1musicmaker_two],
814     [(131, 4), (133, 4), violin1musicmaker_two],
815     [(136, 4), (141, 4), violin1musicmaker_two],
816     [(148, 4), (150, 4), violin1musicmaker_two],
817     [(150, 4), (152, 4), violin1musicmaker_three],
818     [(156, 4), (159, 4), violin1musicmaker_three],
819     [(161, 4), (164, 4), violin1musicmaker_three],
820     [(164, 4), (165, 4), violin1musicmaker_three],
821     [(168, 4), (170, 4), violin1musicmaker_three],
822     [(174, 4), (175, 4), violin1musicmaker_three],
823     [(175, 4), (177, 4), violin1musicmaker_three],
824     [(179, 4), (183, 4), violin1musicmaker_three],
825     [(186, 4), (190, 4), violin1musicmaker_three],
826   ]
827 ])
828
829 ###group two###
830 voice_2_timespan_list = abjad.TimespanList([
831     abjad.AnnotatedTimespan(
832         start_offset=start_offset,
833         stop_offset=stop_offset,
834         annotation=MusicSpecifier(
835             music_maker=music_maker,
836             voice_name='Voice 2',
837         ),
838     )
839     for start_offset, stop_offset, music_maker in [
840         [(2, 4), (5, 4), clarinetmusicmaker_one],
841         [(10, 4), (11, 4), clarinetmusicmaker_two],

```

```

842     [(11, 4), (13, 4), clarinetmusicmaker_two],
843     [(16, 4), (18, 4), clarinetmusicmaker_three],
844     [(21, 4), (22, 4), clarinetmusicmaker_one],
845     [(22, 4), (25, 4), clarinetmusicmaker_one],
846     [(35, 4), (40, 4), clarinetmusicmaker_one],
847     [(44, 4), (46, 4), clarinetmusicmaker_two],
848     [(46, 4), (47, 4), clarinetmusicmaker_two],
849     [(49, 4), (50, 4), clarinetmusicmaker_three],
850     [(55, 4), (59, 4), clarinetmusicmaker_one],
851     [(62, 4), (64, 4), clarinetmusicmaker_two],
852     [(65, 4), (67, 4), clarinetmusicmaker_three],
853     [(67, 4), (70, 4), clarinetmusicmaker_three],
854     [(70, 4), (71, 4), clarinetmusicmaker_three],
855     [(73, 4), (75, 4), clarinetmusicmaker_two],
856     [(75, 4), (76, 4), clarinetmusicmaker_two],
857     [(80, 4), (82, 4), clarinetmusicmaker_one],
858     [(82, 4), (85, 4), clarinetmusicmaker_one],
859     [(86, 4), (88, 4), clarinetmusicmaker_two],
860     [(91, 4), (94, 4), clarinetmusicmaker_three],
861     [(94, 4), (95, 4), clarinetmusicmaker_three],
862     [(100, 4), (101, 4), clarinetmusicmaker_two],
863     [(103, 4), (104, 4), clarinetmusicmaker_one],
864     [(104, 4), (106, 4), clarinetmusicmaker_one],
865     [(110, 4), (114, 4), clarinetmusicmaker_one],
866     [(115, 4), (119, 4), clarinetmusicmaker_one],
867     [(120, 4), (123, 4), clarinetmusicmaker_one],
868     [(123, 4), (124, 4), clarinetmusicmaker_one],
869     [(125, 4), (126, 4), clarinetmusicmaker_two],
870     [(129, 4), (131, 4), clarinetmusicmaker_two],
871     [(131, 4), (134, 4), clarinetmusicmaker_two],
872     [(141, 4), (144, 4), clarinetmusicmaker_two],
873     [(149, 4), (150, 4), clarinetmusicmaker_two],
874     [(155, 4), (159, 4), clarinetmusicmaker_three],
875     [(162, 4), (164, 4), clarinetmusicmaker_three],
876     [(165, 4), (168, 4), clarinetmusicmaker_three],
877     [(168, 4), (170, 4), clarinetmusicmaker_three],
878     [(174, 4), (175, 4), clarinetmusicmaker_three],
879     [(175, 4), (177, 4), clarinetmusicmaker_three],
880     [(179, 4), (180, 4), clarinetmusicmaker_three],
881     [(185, 4), (186, 4), clarinetmusicmaker_three],
882     [(186, 4), (190, 4), clarinetmusicmaker_three],
883   ]
884 ])
885
886 voice_9_timespan_list = abjad.TimespanList([
887     abjad.AnnotatedTimespan(
888         start_offset=start_offset,
889         stop_offset=stop_offset,
890         annotation=MusicSpecifier(
891             music_maker=music_maker,
892             voice_name='Voice 9',
893         ),
894     ),
895     for start_offset, stop_offset, music_maker in [

```

```

896     [(2, 4), (5, 4), violin2musicmaker_one],
897     [(9, 4), (11, 4), violin2musicmaker_two],
898     [(11, 4), (13, 4), violin2musicmaker_two],
899     [(16, 4), (18, 4), violin2musicmaker_three],
900     [(21, 4), (22, 4), violin2musicmaker_one],
901     [(22, 4), (23, 4), violin2musicmaker_one],
902     [(35, 4), (40, 4), violin2musicmaker_one],
903     [(44, 4), (46, 4), violin2musicmaker_two],
904     [(46, 4), (47, 4), violin2musicmaker_two],
905     [(49, 4), (50, 4), violin2musicmaker_three],
906     [(55, 4), (59, 4), violin2musicmaker_one],
907     [(62, 4), (64, 4), violin2musicmaker_two],
908     [(65, 4), (67, 4), violin2musicmaker_three],
909     [(67, 4), (70, 4), violin2musicmaker_three],
910     [(70, 4), (71, 4), violin2musicmaker_three],
911     [(73, 4), (75, 4), violin2musicmaker_two],
912     [(75, 4), (76, 4), violin2musicmaker_two],
913     [(80, 4), (82, 4), violin2musicmaker_one],
914     [(82, 4), (85, 4), violin2musicmaker_one],
915     [(86, 4), (88, 4), violin2musicmaker_two],
916     [(91, 4), (94, 4), violin2musicmaker_three],
917     [(94, 4), (95, 4), violin2musicmaker_three],
918     [(100, 4), (101, 4), violin2musicmaker_two],
919     [(103, 4), (104, 4), violin2musicmaker_one],
920     [(104, 4), (106, 4), violin2musicmaker_one],
921     [(110, 4), (114, 4), violin2musicmaker_one],
922     [(115, 4), (119, 4), violin2musicmaker_one],
923     [(120, 4), (123, 4), violin2musicmaker_one],
924     [(123, 4), (124, 4), violin2musicmaker_one],
925     [(125, 4), (126, 4), violin2musicmaker_two],
926     [(129, 4), (131, 4), violin2musicmaker_two],
927     [(131, 4), (134, 4), violin2musicmaker_two],
928     [(141, 4), (144, 4), violin2musicmaker_two],
929     [(149, 4), (150, 4), violin2musicmaker_two],
930     [(154, 4), (157, 4), violin2musicmaker_three],
931     [(159, 4), (160, 4), violin2musicmaker_three],
932     [(165, 4), (168, 4), violin2musicmaker_three],
933     [(168, 4), (169, 4), violin2musicmaker_three],
934     [(172, 4), (174, 4), violin2musicmaker_three],
935     [(175, 4), (179, 4), violin2musicmaker_three],
936     [(179, 4), (180, 4), violin2musicmaker_three],
937     [(184, 4), (186, 4), violin2musicmaker_three],
938     [(186, 4), (190, 4), violin2musicmaker_three],
939   ]
940 ])
941
942 voice_10_timespan_list = abjad.TimespanList([
943     abjad.AnnotatedTimespan(
944         start_offset=start_offset,
945         stop_offset=stop_offset,
946         annotation=MusicSpecifier(
947             music_maker=music_maker,
948             voice_name='Voice 10',
949         ),

```

```

950 )
951 for start_offset, stop_offset, music_maker in [
952     [(2, 4), (5, 4), violamusicmaker_one],
953     [(9, 4), (11, 4), violamusicmaker_two],
954     [(11, 4), (13, 4), violamusicmaker_two],
955     [(17, 4), (18, 4), violamusicmaker_three],
956     [(21, 4), (22, 4), violamusicmaker_one],
957     [(22, 4), (25, 4), violamusicmaker_one],
958     [(29, 4), (30, 4), violamusicmaker_two],
959     [(30, 4), (32, 4), violamusicmaker_two],
960     [(35, 4), (40, 4), violamusicmaker_one],
961     [(44, 4), (46, 4), violamusicmaker_two],
962     [(46, 4), (47, 4), violamusicmaker_two],
963     [(49, 4), (50, 4), violamusicmaker_three],
964     [(55, 4), (59, 4), violamusicmaker_one],
965     [(62, 4), (64, 4), violamusicmaker_two],
966     [(65, 4), (67, 4), violamusicmaker_three],
967     [(67, 4), (70, 4), violamusicmaker_three],
968     [(70, 4), (71, 4), violamusicmaker_three],
969     [(73, 4), (75, 4), violamusicmaker_two],
970     [(75, 4), (76, 4), violamusicmaker_two],
971     [(80, 4), (82, 4), violamusicmaker_one],
972     [(82, 4), (85, 4), violamusicmaker_one],
973     [(86, 4), (88, 4), violamusicmaker_two],
974     [(91, 4), (94, 4), violamusicmaker_three],
975     [(94, 4), (95, 4), violamusicmaker_three],
976     [(100, 4), (101, 4), violamusicmaker_two],
977     [(103, 4), (104, 4), violamusicmaker_one],
978     [(104, 4), (106, 4), violamusicmaker_one],
979     [(110, 4), (114, 4), violamusicmaker_one],
980     [(115, 4), (119, 4), violamusicmaker_one],
981     [(120, 4), (123, 4), violamusicmaker_one],
982     [(123, 4), (124, 4), violamusicmaker_one],
983     [(125, 4), (126, 4), violamusicmaker_two],
984     [(129, 4), (131, 4), violamusicmaker_two],
985     [(131, 4), (134, 4), violamusicmaker_two],
986     [(141, 4), (144, 4), violamusicmaker_two],
987     [(149, 4), (150, 4), violamusicmaker_two],
988     [(153, 4), (154, 4), violamusicmaker_three],
989     [(154, 4), (155, 4), violamusicmaker_three],
990     [(156, 4), (159, 4), violamusicmaker_three],
991     [(159, 4), (161, 4), violamusicmaker_three],
992     [(165, 4), (168, 4), violamusicmaker_three],
993     [(170, 4), (171, 4), violamusicmaker_three],
994     [(176, 4), (179, 4), violamusicmaker_three],
995     [(179, 4), (180, 4), violamusicmaker_three],
996     [(183, 4), (185, 4), violamusicmaker_three],
997     [(186, 4), (190, 4), violamusicmaker_three],
998 ]
999 ])
1000
1001 ###group three###
1002 voice_3_timespan_list = abjad.TimespanList([
1003     abjad.AnnotatedTimespan(

```

```

1004     start_offset=start_offset,
1005     stop_offset=stop_offset,
1006     annotation=MusicSpecifier(
1007         music_maker=music_maker,
1008         voice_name='Voice 3',
1009     ),
1010 )
1011 for start_offset, stop_offset, music_maker in [
1012     [(7, 4), (11, 4), bassoonmusicmaker_one],
1013     [(15, 4), (16, 4), bassoonmusicmaker_two],
1014     [(19, 4), (22, 4), bassoonmusicmaker_three],
1015     [(22, 4), (23, 4), bassoonmusicmaker_three],
1016     [(27, 4), (30, 4), bassoonmusicmaker_one],
1017     [(32, 4), (35, 4), bassoonmusicmaker_two],
1018     [(35, 4), (36, 4), bassoonmusicmaker_three],
1019     [(37, 4), (40, 4), bassoonmusicmaker_two],
1020     [(40, 4), (42, 4), bassoonmusicmaker_two],
1021     [(46, 4), (49, 4), bassoonmusicmaker_one],
1022     [(51, 4), (52, 4), bassoonmusicmaker_three],
1023     [(57, 4), (59, 4), bassoonmusicmaker_two],
1024     [(59, 4), (61, 4), bassoonmusicmaker_two],
1025     [(64, 4), (66, 4), bassoonmusicmaker_one],
1026     [(67, 4), (70, 4), bassoonmusicmaker_three],
1027     [(70, 4), (72, 4), bassoonmusicmaker_one],
1028     [(72, 4), (73, 4), bassoonmusicmaker_one],
1029     [(77, 4), (79, 4), bassoonmusicmaker_two],
1030     [(79, 4), (82, 4), bassoonmusicmaker_two],
1031     [(83, 4), (85, 4), bassoonmusicmaker_three],
1032     [(88, 4), (89, 4), bassoonmusicmaker_two],
1033     [(89, 4), (92, 4), bassoonmusicmaker_two],
1034     [(97, 4), (98, 4), bassoonmusicmaker_one],
1035     [(100, 4), (103, 4), bassoonmusicmaker_two],
1036     [(107, 4), (110, 4), bassoonmusicmaker_three],
1037     [(110, 4), (112, 4), bassoonmusicmaker_one],
1038     [(113, 4), (114, 4), bassoonmusicmaker_one],
1039     [(114, 4), (117, 4), bassoonmusicmaker_one],
1040     [(118, 4), (119, 4), bassoonmusicmaker_one],
1041     [(119, 4), (122, 4), bassoonmusicmaker_one],
1042     [(123, 4), (125, 4), bassoonmusicmaker_one],
1043     [(126, 4), (131, 4), bassoonmusicmaker_two],
1044     [(138, 4), (141, 4), bassoonmusicmaker_two],
1045     [(146, 4), (150, 4), bassoonmusicmaker_two],
1046     [(150, 4), (154, 4), bassoonmusicmaker_three],
1047     [(154, 4), (155, 4), bassoonmusicmaker_three],
1048     [(159, 4), (162, 4), bassoonmusicmaker_three],
1049     [(164, 4), (165, 4), bassoonmusicmaker_three],
1050     [(170, 4), (172, 4), bassoonmusicmaker_three],
1051     [(172, 4), (174, 4), bassoonmusicmaker_three],
1052     [(177, 4), (179, 4), bassoonmusicmaker_three],
1053     [(180, 4), (183, 4), bassoonmusicmaker_three],
1054     [(183, 4), (185, 4), bassoonmusicmaker_three],
1055     [(186, 4), (190, 4), bassoonmusicmaker_three],
1056 ]
1057 ])

```

```

1058
1059 voice_6_timespan_list = abjad.TimespanList([
1060     abjad.AnnotatedTimespan(
1061         start_offset=start_offset,
1062         stop_offset=stop_offset,
1063         annotation=MusicSpecifier(
1064             music_maker=music_maker,
1065             voice_name='Voice 6',
1066         ),
1067     )
1068     for start_offset, stop_offset, music_maker in [
1069         [(7, 4), (11, 4), trombonemusicmaker_one],
1070         [(14, 4), (16, 4), trombonemusicmaker_two],
1071         [(19, 4), (22, 4), trombonemusicmaker_three],
1072         [(22, 4), (23, 4), trombonemusicmaker_three],
1073         [(27, 4), (29, 4), trombonemusicmaker_one],
1074         [(35, 4), (36, 4), trombonemusicmaker_three],
1075         [(37, 4), (40, 4), trombonemusicmaker_two],
1076         [(40, 4), (42, 4), trombonemusicmaker_two],
1077         [(46, 4), (49, 4), trombonemusicmaker_one],
1078         [(51, 4), (52, 4), trombonemusicmaker_three],
1079         [(57, 4), (59, 4), trombonemusicmaker_two],
1080         [(59, 4), (61, 4), trombonemusicmaker_two],
1081         [(64, 4), (66, 4), trombonemusicmaker_one],
1082         [(67, 4), (70, 4), trombonemusicmaker_three],
1083         [(70, 4), (72, 4), trombonemusicmaker_one],
1084         [(72, 4), (73, 4), trombonemusicmaker_one],
1085         [(77, 4), (79, 4), trombonemusicmaker_two],
1086         [(79, 4), (82, 4), trombonemusicmaker_two],
1087         [(83, 4), (85, 4), trombonemusicmaker_three],
1088         [(88, 4), (89, 4), trombonemusicmaker_two],
1089         [(89, 4), (92, 4), trombonemusicmaker_two],
1090         [(97, 4), (98, 4), trombonemusicmaker_one],
1091         [(100, 4), (103, 4), trombonemusicmaker_two],
1092         [(107, 4), (110, 4), trombonemusicmaker_three],
1093         [(110, 4), (112, 4), trombonemusicmaker_one],
1094         [(113, 4), (114, 4), trombonemusicmaker_one],
1095         [(114, 4), (117, 4), trombonemusicmaker_one],
1096         [(118, 4), (119, 4), trombonemusicmaker_one],
1097         [(119, 4), (122, 4), trombonemusicmaker_one],
1098         [(123, 4), (125, 4), trombonemusicmaker_one],
1099         [(126, 4), (131, 4), trombonemusicmaker_two],
1100         [(138, 4), (141, 4), trombonemusicmaker_two],
1101         [(146, 4), (150, 4), trombonemusicmaker_two],
1102         [(150, 4), (154, 4), trombonemusicmaker_three],
1103         [(157, 4), (159, 4), trombonemusicmaker_three],
1104         [(160, 4), (164, 4), trombonemusicmaker_three],
1105         [(164, 4), (165, 4), trombonemusicmaker_three],
1106         [(169, 4), (172, 4), trombonemusicmaker_three],
1107         [(174, 4), (175, 4), trombonemusicmaker_three],
1108         [(180, 4), (183, 4), trombonemusicmaker_three],
1109         [(183, 4), (184, 4), trombonemusicmaker_three],
1110         [(186, 4), (190, 4), trombonemusicmaker_three],
1111     ]

```

```

1112 ])
1113
1114 voice_11_timespan_list = abjad.TimespanList([
1115     abjad.AnnotatedTimespan(
1116         start_offset=start_offset,
1117         stop_offset=stop_offset,
1118         annotation=MusicSpecifier(
1119             music_maker=music_maker,
1120             voice_name='Voice 11',
1121         ),
1122     )
1123     for start_offset, stop_offset, music_maker in [
1124         [(7, 4), (11, 4), cellomusicmaker_one],
1125         [(14, 4), (16, 4), cellomusicmaker_two],
1126         [(21, 4), (22, 4), cellomusicmaker_three],
1127         [(22, 4), (23, 4), cellomusicmaker_three],
1128         [(27, 4), (30, 4), cellomusicmaker_one],
1129         [(35, 4), (36, 4), cellomusicmaker_three],
1130         [(37, 4), (40, 4), cellomusicmaker_two],
1131         [(40, 4), (42, 4), cellomusicmaker_two],
1132         [(46, 4), (49, 4), cellomusicmaker_one],
1133         [(51, 4), (52, 4), cellomusicmaker_three],
1134         [(57, 4), (59, 4), cellomusicmaker_two],
1135         [(59, 4), (61, 4), cellomusicmaker_two],
1136         [(64, 4), (66, 4), cellomusicmaker_one],
1137         [(67, 4), (70, 4), cellomusicmaker_three],
1138         [(70, 4), (72, 4), cellomusicmaker_one],
1139         [(72, 4), (73, 4), cellomusicmaker_one],
1140         [(77, 4), (79, 4), cellomusicmaker_two],
1141         [(79, 4), (82, 4), cellomusicmaker_two],
1142         [(83, 4), (85, 4), cellomusicmaker_three],
1143         [(88, 4), (89, 4), cellomusicmaker_two],
1144         [(89, 4), (92, 4), cellomusicmaker_two],
1145         [(97, 4), (98, 4), cellomusicmaker_one],
1146         [(100, 4), (103, 4), cellomusicmaker_two],
1147         [(107, 4), (110, 4), cellomusicmaker_three],
1148         [(110, 4), (112, 4), cellomusicmaker_one],
1149         [(113, 4), (114, 4), cellomusicmaker_one],
1150         [(114, 4), (117, 4), cellomusicmaker_one],
1151         [(118, 4), (119, 4), cellomusicmaker_one],
1152         [(119, 4), (122, 4), cellomusicmaker_one],
1153         [(123, 4), (125, 4), cellomusicmaker_one],
1154         [(126, 4), (131, 4), cellomusicmaker_two],
1155         [(138, 4), (141, 4), cellomusicmaker_two],
1156         [(146, 4), (150, 4), cellomusicmaker_two],
1157         [(150, 4), (153, 4), cellomusicmaker_three],
1158         [(155, 4), (156, 4), cellomusicmaker_three],
1159         [(161, 4), (164, 4), cellomusicmaker_three],
1160         [(164, 4), (165, 4), cellomusicmaker_three],
1161         [(168, 4), (170, 4), cellomusicmaker_three],
1162         [(171, 4), (172, 4), cellomusicmaker_three],
1163         [(172, 4), (175, 4), cellomusicmaker_three],
1164         [(175, 4), (176, 4), cellomusicmaker_three],
1165         [(180, 4), (183, 4), cellomusicmaker_three],

```

```

1166     [(185, 4), (186, 4), cellomusicmaker_three],
1167     [(186, 4), (190, 4), cellomusicmaker_three],
1168 ]
1169 ])
1170
1171 ###group four###
1172 voice_4_timespan_list = abjad.TimespanList([
1173     abjad.AnnotatedTimespan(
1174         start_offset=start_offset,
1175         stop_offset=stop_offset,
1176         annotation=MusicSpecifier(
1177             music_maker=music_maker,
1178             voice_name='Voice 4',
1179         ),
1180     )
1181     for start_offset, stop_offset, music_maker in [
1182         [(0, 4), (5, 4), hornmusicmaker_one],
1183         [(8, 4), (10, 4), hornmusicmaker_two],
1184         [(14, 4), (18, 4), hornmusicmaker_three],
1185         [(21, 4), (22, 4), hornmusicmaker_one],
1186         [(22, 4), (23, 4), hornmusicmaker_one],
1187         [(38, 4), (40, 4), hornmusicmaker_two],
1188         [(41, 4), (43, 4), hornmusicmaker_one],
1189         [(43, 4), (46, 4), hornmusicmaker_one],
1190         [(50, 4), (53, 4), hornmusicmaker_three],
1191         [(55, 4), (56, 4), hornmusicmaker_two],
1192         [(61, 4), (64, 4), hornmusicmaker_one],
1193         [(64, 4), (65, 4), hornmusicmaker_one],
1194         [(68, 4), (70, 4), hornmusicmaker_three],
1195         [(70, 4), (72, 4), hornmusicmaker_two],
1196         [(72, 4), (74, 4), hornmusicmaker_two],
1197         [(79, 4), (80, 4), hornmusicmaker_three],
1198         [(82, 4), (85, 4), hornmusicmaker_two],
1199         [(89, 4), (94, 4), hornmusicmaker_one],
1200         [(95, 4), (97, 4), hornmusicmaker_two],
1201         [(100, 4), (104, 4), hornmusicmaker_three],
1202         [(109, 4), (110, 4), hornmusicmaker_two],
1203         [(110, 4), (111, 4), hornmusicmaker_one],
1204         [(112, 4), (114, 4), hornmusicmaker_one],
1205         [(114, 4), (116, 4), hornmusicmaker_one],
1206         [(117, 4), (119, 4), hornmusicmaker_one],
1207         [(119, 4), (121, 4), hornmusicmaker_one],
1208         [(122, 4), (123, 4), hornmusicmaker_one],
1209         [(123, 4), (125, 4), hornmusicmaker_one],
1210         [(133, 4), (136, 4), hornmusicmaker_two],
1211         [(142, 4), (146, 4), hornmusicmaker_two],
1212         [(146, 4), (150, 4), hornmusicmaker_two],
1213         [(153, 4), (154, 4), hornmusicmaker_three],
1214         [(154, 4), (155, 4), hornmusicmaker_three],
1215         [(159, 4), (162, 4), hornmusicmaker_three],
1216         [(164, 4), (168, 4), hornmusicmaker_three],
1217         [(171, 4), (172, 4), hornmusicmaker_three],
1218         [(172, 4), (173, 4), hornmusicmaker_three],
1219         [(177, 4), (179, 4), hornmusicmaker_three],

```

```

1220     [(179, 4), (180, 4), hornmusicmaker_three],
1221     [(182, 4), (183, 4), hornmusicmaker_three],
1222     [(183, 4), (186, 4), hornmusicmaker_three],
1223     [(186, 4), (190, 4), hornmusicmaker_three],
1224   ]
1225 ])
1226
1227 voice_7_timespan_list = abjad.TimespanList([
1228   abjad.AnnotatedTimespan(
1229     start_offset=start_offset,
1230     stop_offset=stop_offset,
1231     annotation=MusicSpecifier(
1232       music_maker=music_maker,
1233       voice_name='Voice 7',
1234     ),
1235   )
1236   for start_offset, stop_offset, music_maker in [
1237     [(0, 4), (5, 4), tubamusicmaker_one],
1238     [(8, 4), (10, 4), tubamusicmaker_two],
1239     [(14, 4), (18, 4), tubamusicmaker_three],
1240     [(21, 4), (22, 4), tubamusicmaker_one],
1241     [(22, 4), (23, 4), tubamusicmaker_one],
1242     [(26, 4), (30, 4), tubamusicmaker_two],
1243     [(38, 4), (40, 4), tubamusicmaker_two],
1244     [(41, 4), (43, 4), tubamusicmaker_one],
1245     [(43, 4), (46, 4), tubamusicmaker_one],
1246     [(50, 4), (53, 4), tubamusicmaker_three],
1247     [(55, 4), (56, 4), tubamusicmaker_two],
1248     [(61, 4), (64, 4), tubamusicmaker_one],
1249     [(64, 4), (65, 4), tubamusicmaker_one],
1250     [(68, 4), (70, 4), tubamusicmaker_three],
1251     [(70, 4), (72, 4), tubamusicmaker_two],
1252     [(72, 4), (74, 4), tubamusicmaker_two],
1253     [(79, 4), (80, 4), tubamusicmaker_three],
1254     [(82, 4), (85, 4), tubamusicmaker_two],
1255     [(89, 4), (94, 4), tubamusicmaker_one],
1256     [(95, 4), (97, 4), tubamusicmaker_two],
1257     [(100, 4), (104, 4), tubamusicmaker_three],
1258     [(109, 4), (110, 4), tubamusicmaker_two],
1259     [(110, 4), (111, 4), tubamusicmaker_one],
1260     [(112, 4), (114, 4), tubamusicmaker_one],
1261     [(114, 4), (116, 4), tubamusicmaker_one],
1262     [(117, 4), (119, 4), tubamusicmaker_one],
1263     [(119, 4), (121, 4), tubamusicmaker_one],
1264     [(122, 4), (123, 4), tubamusicmaker_one],
1265     [(123, 4), (125, 4), tubamusicmaker_one],
1266     [(133, 4), (136, 4), tubamusicmaker_two],
1267     [(142, 4), (146, 4), tubamusicmaker_two],
1268     [(146, 4), (150, 4), tubamusicmaker_two],
1269     [(154, 4), (157, 4), tubamusicmaker_three],
1270     [(159, 4), (163, 4), tubamusicmaker_three],
1271     [(166, 4), (168, 4), tubamusicmaker_three],
1272     [(172, 4), (175, 4), tubamusicmaker_three],
1273     [(177, 4), (179, 4), tubamusicmaker_three],

```

```

1274     [(179, 4), (181, 4), tubamusicmaker_three],
1275     [(184, 4), (186, 4), tubamusicmaker_three],
1276     [(186, 4), (190, 4), tubamusicmaker_three],
1277   ]
1278 ])
1279
1280 voice_12_timespan_list = abjad.TimespanList([
1281     abjad.AnnotatedTimespan(
1282         start_offset=start_offset,
1283         stop_offset=stop_offset,
1284         annotation=MusicSpecifier(
1285             music_maker=music_maker,
1286             voice_name='Voice 12',
1287         ),
1288     )
1289     for start_offset, stop_offset, music_maker in [
1290         [(0, 4), (5, 4), bassmusicmaker_one],
1291         [(8, 4), (10, 4), bassmusicmaker_two],
1292         [(14, 4), (18, 4), bassmusicmaker_three],
1293         [(21, 4), (22, 4), bassmusicmaker_one],
1294         [(22, 4), (23, 4), bassmusicmaker_one],
1295         [(38, 4), (40, 4), bassmusicmaker_two],
1296         [(41, 4), (43, 4), bassmusicmaker_one],
1297         [(43, 4), (46, 4), bassmusicmaker_one],
1298         [(50, 4), (53, 4), bassmusicmaker_three],
1299         [(55, 4), (56, 4), bassmusicmaker_two],
1300         [(61, 4), (64, 4), bassmusicmaker_one],
1301         [(64, 4), (65, 4), bassmusicmaker_one],
1302         [(68, 4), (70, 4), bassmusicmaker_three],
1303         [(70, 4), (72, 4), bassmusicmaker_two],
1304         [(72, 4), (74, 4), bassmusicmaker_two],
1305         [(79, 4), (80, 4), bassmusicmaker_three],
1306         [(82, 4), (85, 4), bassmusicmaker_two],
1307         [(89, 4), (94, 4), bassmusicmaker_one],
1308         [(95, 4), (97, 4), bassmusicmaker_two],
1309         [(100, 4), (104, 4), bassmusicmaker_three],
1310         [(109, 4), (110, 4), bassmusicmaker_two],
1311         [(110, 4), (111, 4), bassmusicmaker_one],
1312         [(112, 4), (114, 4), bassmusicmaker_one],
1313         [(114, 4), (116, 4), bassmusicmaker_one],
1314         [(117, 4), (119, 4), bassmusicmaker_one],
1315         [(119, 4), (121, 4), bassmusicmaker_one],
1316         [(122, 4), (123, 4), bassmusicmaker_one],
1317         [(123, 4), (125, 4), bassmusicmaker_one],
1318         [(133, 4), (136, 4), bassmusicmaker_two],
1319         [(142, 4), (146, 4), bassmusicmaker_two],
1320         [(146, 4), (150, 4), bassmusicmaker_two],
1321         [(152, 4), (154, 4), bassmusicmaker_three],
1322         [(154, 4), (156, 4), bassmusicmaker_three],
1323         [(159, 4), (161, 4), bassmusicmaker_three],
1324         [(165, 4), (168, 4), bassmusicmaker_three],
1325         [(170, 4), (172, 4), bassmusicmaker_three],
1326         [(172, 4), (174, 4), bassmusicmaker_three],
1327         [(177, 4), (179, 4), bassmusicmaker_three],

```

```

1328     [(183, 4), (186, 4), bassmusicmaker_three],
1329     [(186, 4), (190, 4), bassmusicmaker_three],
1330 ]
1331 ])
1332
1333 all_timespan_lists = {
1334     'Voice 1': voice_1_timespan_list,
1335     'Voice 2': voice_2_timespan_list,
1336     'Voice 3': voice_3_timespan_list,
1337     'Voice 4': voice_4_timespan_list,
1338     'Voice 5': voice_5_timespan_list,
1339     'Voice 6': voice_6_timespan_list,
1340     'Voice 7': voice_7_timespan_list,
1341     'Voice 8': voice_8_timespan_list,
1342     'Voice 9': voice_9_timespan_list,
1343     'Voice 10': voice_10_timespan_list,
1344     'Voice 11': voice_11_timespan_list,
1345     'Voice 12': voice_12_timespan_list,
1346 }
1347
1348 global_timespan = abjad.Timespan(
1349     start_offset=0,
1350     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
1351 )
1352
1353
1354 for voice_name, timespan_list in all_timespan_lists.items():
1355     silences = abjad.TimespanList([global_timespan])
1356     silences.extend(timespan_list)
1357     silences.sort()
1358     silences.compute_logical_xor()
1359     for silence_timespan in silences:
1360         timespan_list.append(
1361             abjad.AnnotatedTimespan(
1362                 start_offset=silence_timespan.start_offset,
1363                 stop_offset=silence_timespan.stop_offset,
1364                 annotation=MusicSpecifier(
1365                     music_maker=None,
1366                     voice_name=voice_name,
1367                 ),
1368             )
1369         )
1370     timespan_list.sort()
1371
1372
1373 for voice_name, timespan_list in all_timespan_lists.items():
1374     shards = timespan_list.split_at_offsets(bounds)
1375     split_timespan_list = abjad.TimespanList()
1376     for shard in shards:
1377         split_timespan_list.extend(shard)
1378     split_timespan_list.sort()
1379     all_timespan_lists[voice_name] = timespan_list
1380
1381 score = abjad.Score([

```

```
1382     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 1')
1383     ,
1384     abjad.StaffGroup(
1385         [
1386             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
1387             lilypond_type='Staff'),
1388             abjad.Staff([abjad.Voice(name='Voice 2')], name='Staff 2',
1389             lilypond_type='Staff'),
1390             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
1391             lilypond_type='Staff'),
1392             ],
1393             name='Staff Group 1',
1394         ),
1395     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 2')
1396     ,
1397     abjad.StaffGroup(
1398         [
1399             abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
1400             lilypond_type='Staff'),
1401             abjad.Staff([abjad.Voice(name='Voice 5')], name='Staff 5',
1402             lilypond_type='Staff'),
1403             abjad.Staff([abjad.Voice(name='Voice 6')], name='Staff 6',
1404             lilypond_type='Staff'),
1405             abjad.Staff([abjad.Voice(name='Voice 7')], name='Staff 7',
1406             lilypond_type='Staff'),
1407             ],
1408             name='Staff Group 2',
1409         ),
1410     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context 3')
1411     ,
1412     abjad.StaffGroup(
1413         [
1414             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1415             lilypond_type='Staff'),
1416             abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1417             lilypond_type='Staff'),
1418             abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1419             lilypond_type='Staff'),
1420             abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1421             lilypond_type='Staff'),
1422             abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1423             lilypond_type='Staff'),
1424             ],
1425             name='Staff Group 3',
1426         )
1427     ],
1428 )
1429
1430
1431
1432
1433
1434
1435
1436 for time_signature in time_signatures:
1437     skip = abjad.Skip(1, multiplier=(time_signature))
1438     abjad.attach(time_signature, skip)
1439     score['Global Context 1'].append(skip)
```

```

1421 for time_signature in time_signatures:
1422     skip = abjad.Skip(1, multiplier=(time_signature))
1423     abjad.attach(time_signature, skip)
1424     score['Global Context 2'].append(skip)
1425
1426 for time_signature in time_signatures:
1427     skip = abjad.Skip(1, multiplier=(time_signature))
1428     abjad.attach(time_signature, skip)
1429     score['Global Context 3'].append(skip)
1430
1431 print('Making containers ...')
1432
1433 def make_container(music_maker, durations):
1434     selections = music_maker(durations)
1435     container = abjad.Container([])
1436     container.extend(selections)
1437     return container
1438
1439 def key_function(timespan):
1440     return timespan.annotation.music_maker or silence_maker
1441
1442 for voice_name, timespan_list in all_timespan_lists.items():
1443     for music_maker, grouper in itertools.groupby(
1444         timespan_list,
1445         key=key_function,
1446     ):
1447         durations = [timespan.duration for timespan in grouper]
1448         container = make_container(music_maker, durations)
1449         voice = score[voice_name]
1450         voice.append(container)
1451
1452 print('Splitting and rewriting ...')
1453
1454 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
1455     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1456         time_signature = time_signatures[i]
1457         abjad.mutate(shard).rewrite_meter(time_signature)
1458
1459 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
1460     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1461         time_signature = time_signatures[i]
1462         abjad.mutate(shard).rewrite_meter(time_signature)
1463
1464 for voice in abjad.iterate(score['Staff Group 3']).components(abjad.Voice):
1465     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
1466         time_signature = time_signatures[i]
1467         abjad.mutate(shard).rewrite_meter(time_signature)
1468
1469 print('Beaming runs ...')
1470
1471 for voice in abjad.select(score).components(abjad.Voice):
1472     for run in abjad.select(voice).runs():
1473         if 1 < len(run):
1474             specifier = abjadext.rmakers.BeamSpecifier()

```

```

1475         beam_each_division=False,
1476     )
1477 specifier(run)
1478 abjad.attach(abjad.StartBeam(), run[0])
1479 abjad.attach(abjad.StopBeam(), run[-1])
1480 for leaf in run:
1481     if abjad.Duration(1, 4) <= leaf.written_duration:
1482         continue
1483     previous_leaf = abjad.inspect(leaf).leaf(-1)
1484     next_leaf = abjad.inspect(leaf).leaf(1)
1485     if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
1486         abjad.Duration(1, 4) <= next_leaf.written_duration):
1487         left = previous_leaf.written_duration.flag_count
1488         right = leaf.written_duration.flag_count
1489         beam_count = abjad.BeamCount(
1490             left=left,
1491             right=right,
1492         )
1493         abjad.attach(beam_count, leaf)
1494         continue
1495     if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
1496         abjad.Duration(1, 4) <= previous_leaf.written_duration):
1497         left = leaf.written_duration.flag_count
1498         right = next_leaf.written_duration.flag_count
1499         beam_count = abjad.BeamCount(
1500             left=left,
1501             right=right,
1502         )
1503         abjad.attach(beam_count, leaf)
1504
1505 print('Beautifying score ...')
1506 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1507     for selection in abjad.select(staff).components(abjad.Rest).
1508         group_by_contiguity():
1509             start_command = abjad.LilyPondLiteral(
1510                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1511                 startStaff',
1512                 format_slot='before',
1513             )
1514             stop_command = abjad.LilyPondLiteral(
1515                 r'\stopStaff \startStaff',
1516                 format_slot='after',
1517             )
1518             abjad.attach(start_command, selection[0])
1519             abjad.attach(stop_command, selection[-1])
1520
1521 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1522     for selection in abjad.select(staff).components(abjad.Rest).
1523         group_by_contiguity():
1524             start_command = abjad.LilyPondLiteral(
1525                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1526                 startStaff',
1527                 format_slot='before',
1528             )

```

```

1525     stop_command = abjad.LilyPondLiteral(
1526         r'\stopStaff \startStaff',
1527         format_slot='after',
1528     )
1529     abjad.attach(start_command, selection[0])
1530     abjad.attach(stop_command, selection[-1])
1531
1532 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1533     for selection in abjad.select(staff).components(abjad.Rest).
1534         group_by_contiguity():
1535             start_command = abjad.LilyPondLiteral(
1536                 r'\stopStaff \once \override Staff.StaffSymbol.line-count = #1 \
1537                 startStaff',
1538                 format_slot='before',
1539             )
1540             stop_command = abjad.LilyPondLiteral(
1541                 r'\stopStaff \startStaff',
1542                 format_slot='after',
1543             )
1544             abjad.attach(start_command, selection[0])
1545             abjad.attach(stop_command, selection[-1])
1546
1547 print('Stopping Hairpins ...')
1548 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1549     for rest in abjad.iterate(staff).components(abjad.Rest):
1550         previous_leaf = abjad.inspect(rest).leaf(-1)
1551         if isinstance(previous_leaf, abjad.Note):
1552             abjad.attach(abjad.StopHairpin(), rest)
1553         elif isinstance(previous_leaf, abjad.Chord):
1554             abjad.attach(abjad.StopHairpin(), rest)
1555         elif isinstance(previous_leaf, abjad.Rest):
1556             pass
1557
1558 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1559     for rest in abjad.iterate(staff).components(abjad.Rest):
1560         previous_leaf = abjad.inspect(rest).leaf(-1)
1561         if isinstance(previous_leaf, abjad.Note):
1562             abjad.attach(abjad.StopHairpin(), rest)
1563         elif isinstance(previous_leaf, abjad.Chord):
1564             abjad.attach(abjad.StopHairpin(), rest)
1565         elif isinstance(previous_leaf, abjad.Rest):
1566             pass
1567
1568 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1569     for rest in abjad.iterate(staff).components(abjad.Rest):
1570         previous_leaf = abjad.inspect(rest).leaf(-1)
1571         if isinstance(previous_leaf, abjad.Note):
1572             abjad.attach(abjad.StopHairpin(), rest)
1573         elif isinstance(previous_leaf, abjad.Chord):
1574             abjad.attach(abjad.StopHairpin(), rest)
1575         elif isinstance(previous_leaf, abjad.Rest):
1576             pass
1577
1578 print('Adding pitch material ...')

```

```

1577 def cyc(lst):
1578     count = 0
1579     while True:
1580         yield lst[count%len(lst)]
1581         count += 1
1582
1583
1584 print('Adding attachments ...')
1585 bar_line = abjad.BarLine('|.')
1586 metro = abjad.MetronomeMark((1, 4), 120)
1587 markup1 = abjad.Markup(r'\bold { S }')
1588 markup2 = abjad.Markup(r'\bold { T }')
1589 markup3 = abjad.Markup(r'\bold { U }')
1590 markup4 = abjad.Markup(r'\bold { V }')
1591 markup5 = abjad.Markup(r'\bold { W }')
1592 markup6 = abjad.Markup(r'\bold { X }')
1593 mark1 = abjad.RehearsalMark(markup=markup1)
1594 mark2 = abjad.RehearsalMark(markup=markup2)
1595 mark3 = abjad.RehearsalMark(markup=markup3)
1596 mark4 = abjad.RehearsalMark(markup=markup4)
1597 mark5 = abjad.RehearsalMark(markup=markup5)
1598 mark6 = abjad.RehearsalMark(markup=markup6)
1599
1600 instruments1 = cyc([
1601     abjad.Flute(),
1602     abjad.ClarinetInBFlat(),
1603     abjad.Bassoon(),
1604 ])
1605
1606 instruments2 = cyc([
1607     abjad.FrenchHorn(),
1608     abjad.Trombone(),
1609     abjad.TenorTrombone(),
1610     abjad.Tuba(),
1611 ])
1612
1613 instruments3 = cyc([
1614     abjad.Violin(),
1615     abjad.Violin(),
1616     abjad.Viola(),
1617     abjad.Cello(),
1618     abjad.Contrabass(),
1619 ])
1620
1621 clefs1 = cyc([
1622     abjad.Clef('treble'),
1623     abjad.Clef('treble'),
1624     abjad.Clef('bass'),
1625 ])
1626
1627 clefs2 = cyc([
1628     abjad.Clef('treble'),
1629     abjad.Clef('treble'),
1630     abjad.Clef('bass'),

```

```

1631     abjad.Clef('bass'),
1632   ])
1633
1634 clefs3 = cyc([
1635     abjad.Clef('treble'),
1636     abjad.Clef('treble'),
1637     abjad.Clef('alto'),
1638     abjad.Clef('bass'),
1639     abjad.Clef('bass'),
1640   ])
1641
1642 abbreviations1 = cyc([
1643     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),
1644     abjad.MarginMarkup(markup=abjad.Markup('cl.'),),
1645     abjad.MarginMarkup(markup=abjad.Markup('bssn.'),),
1646   ])
1647
1648 abbreviations2 = cyc([
1649     abjad.MarginMarkup(markup=abjad.Markup('hr.'),),
1650     abjad.MarginMarkup(markup=abjad.Markup('trp.'),),
1651     abjad.MarginMarkup(markup=abjad.Markup('trmb.'),),
1652     abjad.MarginMarkup(markup=abjad.Markup('tb.'),),
1653   ])
1654
1655 abbreviations3 = cyc([
1656     abjad.MarginMarkup(markup=abjad.Markup('vln.I'),),
1657     abjad.MarginMarkup(markup=abjad.Markup('vln.II'),),
1658     abjad.MarginMarkup(markup=abjad.Markup('vla.'),),
1659     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
1660     abjad.MarginMarkup(markup=abjad.Markup('cb.'),),
1661   ])
1662
1663 names1 = cyc([
1664     abjad.StartMarkup(markup=abjad.Markup('Flute'),),
1665     abjad.StartMarkup(markup=abjad.Markup('Clarinet'),),
1666     abjad.StartMarkup(markup=abjad.Markup('Bassoon'),),
1667   ])
1668
1669 names2 = cyc([
1670     abjad.StartMarkup(markup=abjad.Markup('Horn'),),
1671     abjad.StartMarkup(markup=abjad.Markup('Trumpet'),),
1672     abjad.StartMarkup(markup=abjad.Markup('Trombone'),),
1673     abjad.StartMarkup(markup=abjad.Markup('Tuba'),),
1674   ])
1675
1676 names3 = cyc([
1677     abjad.StartMarkup(markup=abjad.Markup('Violin I'),),
1678     abjad.StartMarkup(markup=abjad.Markup('Violin II'),),
1679     abjad.StartMarkup(markup=abjad.Markup('Viola'),),
1680     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
1681     abjad.StartMarkup(markup=abjad.Markup('Contrabass'),),
1682   ])
1683
1684 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

```

```

1685 leaf1 = abjad.select(staff).leaves()[0]
1686 abjad.attach(next(instruments1), leaf1)
1687 abjad.attach(next(abbreviations1), leaf1)
1688 abjad.attach(next(names1), leaf1)
1689 abjad.attach(next(clefs1), leaf1)
1690
1691 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1692     leaf1 = abjad.select(staff).leaves()[0]
1693     abjad.attach(next(instruments2), leaf1)
1694     abjad.attach(next(abbreviations2), leaf1)
1695     abjad.attach(next(names2), leaf1)
1696     abjad.attach(next(clefs2), leaf1)
1697
1698 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1699     leaf1 = abjad.select(staff).leaves()[0]
1700     abjad.attach(next(instruments3), leaf1)
1701     abjad.attach(next(abbreviations3), leaf1)
1702     abjad.attach(next(names3), leaf1)
1703     abjad.attach(next(clefs3), leaf1)
1704
1705 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
1706     leaf1 = abjad.select(staff).leaves()[0]
1707     last_leaf = abjad.select(staff).leaves()[-1]
1708     abjad.attach(metro, leaf1)
1709     abjad.attach(bar_line, last_leaf)
1710
1711 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff)[0]:
1712     leaf1 = abjad.select(staff).leaves()[0]
1713     last_leaf = abjad.select(staff).leaves()[-1]
1714     abjad.attach(metro, leaf1)
1715     abjad.attach(bar_line, last_leaf)
1716
1717 for staff in abjad.select(score['Staff Group 3']).components(abjad.Staff)[0]:
1718     leaf1 = abjad.select(staff).leaves()[0]
1719     last_leaf = abjad.select(staff).leaves()[-1]
1720     abjad.attach(metro, leaf1)
1721     abjad.attach(bar_line, last_leaf)
1722
1723 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1724     leaf1 = abjad.select(staff).leaves()[7]
1725     abjad.attach(mark1, leaf1)
1726
1727 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1728     leaf1 = abjad.select(staff).leaves()[7]
1729     abjad.attach(mark1, leaf1)
1730
1731 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1732     leaf1 = abjad.select(staff).leaves()[7]
1733     abjad.attach(mark1, leaf1)
1734
1735 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1736     leaf2 = abjad.select(staff).leaves()[16]
1737     abjad.attach(mark2, leaf2)
1738

```

```

1739 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1740     leaf2 = abjad.select(staff).leaves()[16]
1741     abjad.attach(mark2, leaf2)
1742
1743 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1744     leaf2 = abjad.select(staff).leaves()[16]
1745     abjad.attach(mark2, leaf2)
1746
1747 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1748     leaf3 = abjad.select(staff).leaves()[22]
1749     abjad.attach(mark3, leaf3)
1750
1751 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1752     leaf3 = abjad.select(staff).leaves()[22]
1753     abjad.attach(mark3, leaf3)
1754
1755 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1756     leaf3 = abjad.select(staff).leaves()[22]
1757     abjad.attach(mark3, leaf3)
1758
1759 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1760     leaf4 = abjad.select(staff).leaves()[29]
1761     abjad.attach(mark4, leaf4)
1762
1763 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1764     leaf4 = abjad.select(staff).leaves()[29]
1765     abjad.attach(mark4, leaf4)
1766
1767 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1768     leaf4 = abjad.select(staff).leaves()[29]
1769     abjad.attach(mark4, leaf4)
1770
1771 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1772     leaf5 = abjad.select(staff).leaves()[34]
1773     abjad.attach(mark5, leaf5)
1774
1775 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1776     leaf5 = abjad.select(staff).leaves()[34]
1777     abjad.attach(mark5, leaf5)
1778
1779 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1780     leaf5 = abjad.select(staff).leaves()[34]
1781     abjad.attach(mark5, leaf5)
1782
1783 for staff in abjad.iterate(score['Global Context 1']).components(abjad.Staff):
1784     leaf6 = abjad.select(staff).leaves()[39]
1785     abjad.attach(mark6, leaf6)
1786
1787 for staff in abjad.iterate(score['Global Context 2']).components(abjad.Staff):
1788     leaf6 = abjad.select(staff).leaves()[39]
1789     abjad.attach(mark6, leaf6)
1790
1791 for staff in abjad.iterate(score['Global Context 3']).components(abjad.Staff):
1792     leaf6 = abjad.select(staff).leaves()[39]

```

```

1793 abjad.attach(mark6, leaf6)
1794
1795 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
1796     abjad.Instrument.transpose_from_sounding_pitch(staff)
1797
1798 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.Staff):
1799     abjad.Instrument.transpose_from_sounding_pitch(staff)
1800
1801 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.Staff):
1802     abjad.Instrument.transpose_from_sounding_pitch(staff)
1803
1804 score_file = abjad.LilyPondFile.new(
1805     score,
1806     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
1807     _stylesheets/abjad.ily'],
1808 )
1809
1810 abjad.SegmentMaker.comment_measure_numbers(score)
1811 ######
1812 directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/Segment_IV'
1813 pdf_path = f'{directory}/Segment_IV.pdf'
1814 path = pathlib.Path('Segment_IV.pdf')
1815 if path.exists():
1816     print(f'Removing {pdf_path} ...')
1817     path.unlink()
1818 time_1 = time.time()
1819 print(f'Persisting {pdf_path} ...')
1820 result = abjad.persist(score_file).as_pdf(pdf_path)
1821 print(result[0])
1822 print(result[1])
1823 print(result[2])
1824 success = result[3]
1825 if success is False:
1826     print('LilyPond failed!')
1827 time_2 = time.time()
1828 total_time = time_2 - time_1
1829 print(f'Total time: {total_time} seconds')
1830 if path.exists():
1831     print(f'Opening {pdf_path} ...')
1832     os.system(f'open {pdf_path}')

```

**Code Example A.14:** Tianshu Segment\_IV

### A.3.2 STYLESHEET

```

1 Tianshu Stylesheet.
2 % 2018-07-17 19:54
3
4 \version "2.19.82"

```

```

5 \language "english"
6 #(set-default-paper-size "11x17portrait")
7 #(set-global-staff-size 12)
8 \include "ekmel.ily"
9 \ekmelicStyle evans
10
11 \header {
12   tagline = ##f
13   breakbefore = ##t
14   dedication = \markup \override #'(
15     font-name . "Didot"
16   ) \fontsize #6 \italic {
17     "for Ensemble Ibis"
18   }
19   title = \markup { \epsfile #Y #30
20   #"/Users/evansdsg2/Scores//tianshu/tianshu/Segments/Segment_I/tianshu_title.
21   eps"
22 }
23   subtitle = \markup \override #'(
24     font-name . "Didot"
25   ) \fontsize #9 \bold \center-column {
26     "Tianshu"
27   }
28   subsubtitle = \markup \override #'(
29     font-name . "Didot"
30   ) \fontsize #5 \center-column {
31     "for twelve players"
32   }
33   arranger = \markup \override #'(
34     font-name . "Didot"
35   ) \fontsize #2.3 {
36     "Gregory Rowland Evans"
37   }

```

```

37 }
38
39 \layout {
40     \accidentalStyle forget
41     %\accidentalStyle modern
42     %\accidentalStyle modern-cautionary
43     %\accidentalStyle neo-modern
44     %\accidentalStyle dodecaphonic
45     indent = #5
46     %ragged-last = ##t
47     %ragged-right = ##t
48     %left-margin = #15
49 \context {
50     \name TimeSignatureContext
51     \type Engraver_group
52     \numericTimeSignature
53     \consists Axis_group_engraver
54     \consists Bar_number_engraver
55     \consists Time_signature_engraver
56     \consists Mark_engraver
57     \consists Metronome_mark_engraver
58     \override BarNumber.Y-extent = #'(0 . 0)
59     \override BarNumber.Y-offset = 0
60     \override BarNumber.extra-offset = #'(-4 . 0)
61     %\override BarNumber.font-name = "Didot"
62     \override BarNumber.stencil = #(
63         make-stencil-boxer 0.1 0.7 ly:text-interface::print
64     )
65     \override BarNumber.font-size = 1
66     \override BarNumber.padding = 4
67     \override MetronomeMark.X-extent = #'(0 . 0)
68     \override MetronomeMark.Y-extent = #'(0 . 0)
69     \override MetronomeMark.break-align-symbols = #'(left-edge)

```

```

70   \override MetronomeMark.extra-offset = #'(0 . 4)
71   \override MetronomeMark.font-size = 3
72   \override RehearsalMark.stencil = #(
73     make-stencil-circler 0.1 0.7 ly:text-interface::print
74   )
75   \override RehearsalMark.X-extent = #'(0 . 0)
76   \override RehearsalMark.X-offset = 6
77   \override RehearsalMark.Y-offset = -2.25
78   \override RehearsalMark.break-align-symbols = #'(time-signature)
79   \override RehearsalMark.break-visibility = #end-of-line-invisible
80   \override RehearsalMark.font-name = "Didot"
81   \override RehearsalMark.font-size = 8
82   \override RehearsalMark.outside-staff-priority = 500
83   \override RehearsalMark.self-alignment-X = #center
84     \override TimeSignature.X-extent = #'(0 . 0)
85     \override TimeSignature.X-offset = #ly:self-alignment-interface::x-
aligned-on-self
86     \override TimeSignature.Y-extent = #'(0 . 0)
87     \override TimeSignature.break-align-symbol = ##f
88     \override TimeSignature.break-visibility = #end-of-line-invisible
89     \override TimeSignature.font-size = #6
90     \override TimeSignature.self-alignment-X = #center
91   \override TimeSignature.whiteout = ##t
92     \override VerticalAxisGroup.default-staff-staff-spacing = #'(
93       (basic-distance . 0) (minimum-distance . 10) (padding . 6) (stretchability
94       . 0)
95     )
96   }
97   \context {
98     \Score
99     \remove Bar_number_engraver
100    \remove Mark_engraver
101    \accepts TimeSignatureContext

```

```

101   \override BarLine.bar-extent = #'(-2 . 2)
102   \override Beam.breakable = ##t
103   \override Beam.concaveness = #10000
104   \override Glissando.breakable = ##t
105   \override SpacingSpanner.strict-grace-spacing = ##t
106   \override SpacingSpanner.strict-note-spacing = ##t
107   \override SpacingSpanner.uniform-stretching = ##t
108   \override StaffGrouper.staff-staff-spacing = #'(
109     (basic-distance . 25) (minimum-distance . 25) (padding . 3)
110   )
111   \override TupletBracket.bracket-visibility = ##t
112   \override TupletBracket.minimum-length = #3
113   \override TupletBracket.padding = #2
114   \override TupletBracket.springs-and-rods = #ly:spanner::set-spacing-
115     rods
116   \override TupletNumber.text = #tuplet-number::calc-fraction-text
117   proportionalNotationDuration = #(ly:make-moment 1 42)
118   autoBeaming = ##f
119   tupletFullLength = ##t
120 }
121 \context {
122   \Voice
123   \remove Forbid_line_break_engraver
124 }
125 \context {
126   \Staff
127   \remove Time_signature_engraver
128 }
129 \context {
130   \RhythmicStaff
131   \remove Time_signature_engraver
132 }

```

```

133     \StaffGroup
134 }
135 }
136
137 \paper {
138
139     top-margin = 1.5\cm
140     bottom-margin = 1.5\cm
141
142     %top-margin = .90\in
143     oddHeaderMarkup = \markup ""
144     evenHeaderMarkup = \markup ""
145     oddFooterMarkup = \markup \fill-line {
146         " "
147         \concat {
148             " ~ "
149             \fontsize #2
150             \fromproperty #'page:page-number-string "~"
151         }
152         " "
153     }
154     evenFooterMarkup = \markup \fill-line {
155         " "
156         \concat { " ~ " \fontsize #2
157             \fromproperty #'page:page-number-string "~"
158         } " "
159     }
160 }
```

**Code Example A.15:** Tianshu Stylesheet

#### A.4 FOUR AGES OF SAND (FOR FLUTE, ALTO SAXOPHONE, AND VIOLONCELLO) SOURCE CODE

#### A.4.1 SEGMENTS

#### A.4.1.1 SEGMENT I

```
1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (9, 8), (6, 8), (5, 4), (4, 4), (6, 8),
17         (3, 8), (4, 4), (3, 4), (5, 8), (4, 4),
18         (11, 8),
19     ]
20 ]
21
22 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures])
23
24 def reduceMod3(rw):
25     return [(x % 4) for x in rw]
26
27 def reduceMod7(rw):
28     return [(x % 8) for x in rw]
29
30 def reduceMod9(rw):
31     return [(x % 10) for x in rw]
32
33 def cyc(lst):
34     count = 0
35     while True:
36         yield lst[count%len(lst)]
37         count += 1
38
39 def grouper(lst1, lst2):
40     def cyc(lst):
```

```

41     c = 0
42     while True:
43         yield lst[c%len(lst)]
44         c += 1
45     lst1 = cyc(lst1)
46     return [next(lst1) if i == 1 else [next(lst1) for _ in range(i)] for i in
47     lst2]
48
48 seed(1)
49 flute_random_walk_one = []
50 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
51 for i in range(1, 1000):
52     movement = -1 if random() < 0.5 else 1
53     value = flute_random_walk_one[i-1] + movement
54     flute_random_walk_one.append(value)
55 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
56 flute_chord_one = [1, 2, 9, 10, 18, 27, 18, 10, 9, 2, ]
57 flute_notes_one = [flute_chord_one[x] for x in reduceMod9(
58     flute_random_walk_one)]
59
59 seed(4)
60 saxophone_random_walk_one = []
61 saxophone_random_walk_one.append(-1 if random() < 0.5 else 1)
62 for i in range(1, 1000):
63     movement = -1 if random() < 0.5 else 1
64     value = saxophone_random_walk_one[i-1] + movement
65     saxophone_random_walk_one.append(value)
66 saxophone_random_walk_one = [abs(x) for x in saxophone_random_walk_one]
67 saxophone_chord_one = [-8, 1, 2, 9, 10, 9, 2, 1, ]
68 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod7(
69     saxophone_random_walk_one)]
70
70 seed(8)
71 cello_random_walk_one = []
72 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
73 for i in range(1, 1000):
74     movement = -1 if random() < 0.5 else 1
75     value = cello_random_walk_one[i-1] + movement
76     cello_random_walk_one.append(value)
77 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
78 cello_chord_one = [-24, -17, -16, -8, 1, 2, 1, -8, -16, -17, ]
79 cello_notes_one = [cello_chord_one[x] for x in reduceMod9(
80     cello_random_walk_one)]
81
81 seed(1)
82 flute_random_walk_two = []
83 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
84 for i in range(1, 1000):
85     movement = -1 if random() < 0.5 else 1
86     value = flute_random_walk_two[i-1] + movement
87     flute_random_walk_two.append(value)
88 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
89 flute_chord_two = [1, 9, 18, 9, ]
90 flute_notes_two = [flute_chord_two[x] for x in reduceMod3(

```

```

        flute_random_walk_two)]

91 seed(4)
92 saxophone_random_walk_two = []
93 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
94 for i in range(1, 1000):
95     movement = -1 if random() < 0.5 else 1
96     value = saxophone_random_walk_two[i-1] + movement
97     saxophone_random_walk_two.append(value)
98 saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
99 saxophone_chord_two = [-8, 2, 10, 2, ]
100 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod3(
101     saxophone_random_walk_two)]

102 seed(8)
103 cello_random_walk_two = []
104 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
105 for i in range(1, 1000):
106     movement = -1 if random() < 0.5 else 1
107     value = cello_random_walk_two[i-1] + movement
108     cello_random_walk_two.append(value)
109 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
110 cello_chord_two = [-24, -16, 1, -16, ]
111 cello_notes_two = [cello_chord_two[x] for x in reduceMod3(
112     cello_random_walk_two)]

113 seed(1)
114 flute_random_walk_three = []
115 flute_random_walk_three.append(-1 if random() < 0.5 else 1)
116 for i in range(1, 1000):
117     movement = -1 if random() < 0.5 else 1
118     value = flute_random_walk_three[i-1] + movement
119     flute_random_walk_three.append(value)
120 flute_random_walk_three = [abs(x) for x in flute_random_walk_three]
121 flute_chord_three = [2, 10, 27, 10, ]
122 flute_notes_three = [flute_chord_three[x] for x in reduceMod3(
123     flute_random_walk_three)]

124 seed(4)
125 saxophone_random_walk_three = []
126 saxophone_random_walk_three.append(-1 if random() < 0.5 else 1)
127 for i in range(1, 1000):
128     movement = -1 if random() < 0.5 else 1
129     value = saxophone_random_walk_three[i-1] + movement
130     saxophone_random_walk_three.append(value)
131 saxophone_random_walk_three = [abs(x) for x in saxophone_random_walk_three]
132 saxophone_chord_three = [1, 2, 10, 2, ]
133 saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod3(
134     saxophone_random_walk_three)]

135 seed(8)
136 cello_random_walk_three = []
137 cello_random_walk_three.append(-1 if random() < 0.5 else 1)
138 for i in range(1, 1000):

```

```

140     movement = -1 if random() < 0.5 else 1
141     value = cello_random_walk_three[i-1] + movement
142     cello_random_walk_three.append(value)
143 cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
144 cello_chord_three = [-17, -8, 2, -8, ]
145 cello_notes_three = [cello_chord_three[x] for x in reduceMod3(
146     cello_random_walk_three)]
147
148 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
149     talea=abjadext.rmakers.Talea(
150         counts=[2, 7, 2, 3, 2, 4, 7, 2, 5, 6],
151         denominator=32,
152         ),
153     beamSpecifier=abjadext.rmakers.BeamSpecifier(
154         beamDivisionsTogether=True,
155         beamRests=False,
156         ),
157     extraCountsPerDivision=[0, 1, 0, -1, 1, ],
158     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
159         leftClasses=[abjad.Note, abjad.Rest],
160         leftCounts=[1, 0, 1],
161         ),
162     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
163         trivialize=True,
164         extractTrivial=True,
165         rewriteRestFilled=True,
166         ),
167     )
168
169 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
170     talea=abjadext.rmakers.Talea(
171         counts=[1, 2, 2, 3, 3, 3, 2, 2, 1, ],
172         denominator=16,
173         ),
174     beamSpecifier=abjadext.rmakers.BeamSpecifier(
175         beamDivisionsTogether=True,
176         beamRests=False,
177         ),
178     extraCountsPerDivision=[1, 0, -1, 0, 1],
179     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
180         leftClasses=[abjad.Note, abjad.Rest],
181         leftCounts=[1, 0, 1],
182         ),
183     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
184         trivialize=True,
185         extractTrivial=True,
186         rewriteRestFilled=True,
187         ),
188     )
189
190 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
191     denominators=[16, 16, 8, 16, 16, 8],
192     extraCountsPerDivision=[1, 0, 0, -1, 0, 1, -1, 0],

```

```

193     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
194         left_classes=[abjad.Rest],
195         left_counts=[1],
196         right_classes=[abjad.Rest],
197         right_counts=[1],
198         outer_divisions_only=True,
199         ),
200     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
201         trivialize=True,
202         extract_trivial=True,
203         rewrite_rest_filled=True,
204         ),
205     )
206
207
208 attachmentHandlerOne = AttachmentHandler(
209     starting_dynamic='mf',
210     ending_dynamic='p',
211     hairpin_indicator='--',
212     articulation='accent',
213     )
214
215 attachmentHandlerTwo = AttachmentHandler(
216     starting_dynamic='mf',
217     ending_dynamic='ff',
218     hairpin_indicator='<',
219     articulation='tenuto',
220     )
221
222 attachmentHandlerThree = AttachmentHandler(
223     starting_dynamic='ff',
224     ending_dynamic='mp',
225     hairpin_indicator='|>',
226     articulation='|',
227     )
228
229 #####oboe#####
230 fluteMusicMakerOne = MusicMaker(
231     rmaker=rmakerOne,
232     pitches=fluteNotesOne,
233     continuous=True,
234     attachmentHandler=attachmentHandlerOne,
235     )
236 fluteMusicMakerTwo = MusicMaker(
237     rmaker=rmakerTwo,
238     pitches=fluteNotesTwo,
239     continuous=True,
240     attachmentHandler=attachmentHandlerTwo,
241     )
242 fluteMusicMakerThree = MusicMaker(
243     rmaker=rmakerThree,
244     pitches=fluteNotesThree,
245     continuous=True,
246     attachmentHandler=attachmentHandlerThree,

```

```

247 )
248 #####saxophone#####
249 saxophonemusicmaker_one = MusicMaker(
250     rmaker=rmaker_one,
251     pitches=saxophone_notes_one,
252     continuous=True,
253     attachment_handler=attachment_handler_one,
254 )
255 saxophonemusicmaker_two = MusicMaker(
256     rmaker=rmaker_two,
257     pitches=saxophone_notes_two,
258     continuous=True,
259     attachment_handler=attachment_handler_two,
260 )
261 saxophonemusicmaker_three = MusicMaker(
262     rmaker=rmaker_three,
263     pitches=saxophone_notes_three,
264     continuous=True,
265     attachment_handler=attachment_handler_three,
266 )
267 #####cello#####
268 cellomusicmaker_one = MusicMaker(
269     rmaker=rmaker_one,
270     pitches=cello_notes_one,
271     continuous=True,
272     attachment_handler=attachment_handler_one,
273 )
274 cellomusicmaker_two = MusicMaker(
275     rmaker=rmaker_two,
276     pitches=cello_notes_two,
277     continuous=True,
278     attachment_handler=attachment_handler_two,
279 )
280 cellomusicmaker_three = MusicMaker(
281     rmaker=rmaker_three,
282     pitches=cello_notes_three,
283     continuous=True,
284     attachment_handler=attachment_handler_three,
285 )
286
287 silence_maker = abjadext.rmakers.NoteRhythmMaker(
288     division_masks=[
289         abjadext.rmakers.SilenceMask(
290             pattern=abjad.index([0], 1),
291         ),
292     ],
293 )
294
295 class MusicSpecifier:
296
297     def __init__(self, music_maker, voice_name):
298         self.music_maker = music_maker
299         self.voice_name = voice_name
300

```

```

301 print('Collecting timespans and rmakers ...')
302 #####group one#####
303 voice_1_timespan_list = abjad.TimespanList([
304     abjad.AnnotatedTimespan(
305         start_offset=start_offset,
306         stop_offset=stop_offset,
307         annotation=MusicSpecifier(
308             music_maker=music_maker,
309             voice_name='Voice 1',
310         ),
311     ),
312     for start_offset, stop_offset, music_maker in [
313         [(0, 8), (3, 8), flutemusicmaker_one],
314         [(4, 8), (8, 8), flutemusicmaker_two],
315         [(10, 8), (12, 8), flutemusicmaker_three],
316         [(12, 8), (15, 8), flutemusicmaker_one],
317         [(18, 8), (24, 8), flutemusicmaker_two],
318         [(28, 8), (33, 8), flutemusicmaker_three],
319         [(33, 8), (35, 8), flutemusicmaker_one],
320         [(40, 8), (42, 8), flutemusicmaker_two],
321         [(42, 8), (44, 8), flutemusicmaker_three],
322         [(44, 8), (48, 8), flutemusicmaker_one],
323         [(54, 8), (55, 8), flutemusicmaker_two],
324         [(62, 8), (64, 8), flutemusicmaker_three],
325         [(72, 8), (75, 8), flutemusicmaker_one],
326         [(76, 8), (79, 8), flutemusicmaker_two],
327         [(79, 8), (80, 8), silence_maker],
328     ],
329 ],
330 ])
331
332 voice_3_timespan_list = abjad.TimespanList([
333     abjad.AnnotatedTimespan(
334         start_offset=start_offset,
335         stop_offset=stop_offset,
336         annotation=MusicSpecifier(
337             music_maker=music_maker,
338             voice_name='Voice 3',
339         ),
340     ),
341     for start_offset, stop_offset, music_maker in [
342         [(9, 8), (12, 8), saxophonemusicmaker_one],
343         [(20, 8), (24, 8), saxophonemusicmaker_two],
344         [(31, 8), (33, 8), saxophonemusicmaker_three],
345         [(33, 8), (36, 8), saxophonemusicmaker_one],
346         [(42, 8), (48, 8), saxophonemusicmaker_two],
347         [(53, 8), (56, 8), saxophonemusicmaker_three],
348         [(56, 8), (60, 8), saxophonemusicmaker_one],
349         [(64, 8), (69, 8), saxophonemusicmaker_two],
350         [(69, 8), (72, 8), saxophonemusicmaker_three],
351         [(75, 8), (79, 8), saxophonemusicmaker_one],
352     ],
353 ],
354 ])

```

```

355 voice_8_timespan_list = abjad.TimespanList([
356     abjad.AnnotatedTimespan(
357         start_offset=start_offset,
358         stop_offset=stop_offset,
359         annotation=MusicSpecifier(
360             music_maker=music_maker,
361             voice_name='Voice 8',
362         ),
363     )
364     for start_offset, stop_offset, music_maker in [
365         [(15, 8), (18, 8), celломузыкаmaker_one],
366         [(18, 8), (22, 8), celломузыкаmaker_two],
367         [(25, 8), (29, 8), celломузыкаmaker_three],
368         [(29, 8), (32, 8), celломузыкаmaker_one],
369         [(35, 8), (39, 8), celломузыкаmaker_two],
370         [(39, 8), (42, 8), celломузыкаmaker_three],
371         [(45, 8), (50, 8), celломузыкаmaker_one],
372         [(50, 8), (52, 8), celломузыкаmaker_two],
373         [(55, 8), (56, 8), celломузыкаmaker_three],
374         [(56, 8), (61, 8), celломузыкаmaker_one],
375         [(61, 8), (62, 8), celломузыкаmaker_two],
376         [(65, 8), (69, 8), celломузыкаmaker_three],
377         [(69, 8), (72, 8), celломузыкаmaker_one],
378         [(75, 8), (79, 8), celломузыкаmaker_two],
379     ]
380   ])
381
382 all_timespan_lists = {
383     'Voice 1': voice_1_timespan_list,
384     'Voice 3': voice_3_timespan_list,
385     'Voice 8': voice_8_timespan_list,
386   }
387
388 global_timespan = abjad.Timespan(
389     start_offset=0,
390     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
391   )
392
393
394 for voice_name, timespan_list in all_timespan_lists.items():
395     silences = abjad.TimespanList([global_timespan])
396     silences.extend(timespan_list)
397     silences.sort()
398     silences.compute_logical_xor()
399     for silence_timespan in silences:
400         timespan_list.append(
401             abjad.AnnotatedTimespan(
402                 start_offset=silence_timespan.start_offset,
403                 stop_offset=silence_timespan.stop_offset,
404                 annotation=MusicSpecifier(
405                     music_maker=None,
406                     voice_name=voice_name,
407                 ),
408             )

```

```

409     )
410 timespan_list.sort()
411
412 for voice_name, timespan_list in all_timespan_lists.items():
413     shards = timespan_list.split_at_offsets(bounds)
414     split_timespan_list = abjad.TimespanList()
415     for shard in shards:
416         split_timespan_list.extend(shard)
417     split_timespan_list.sort()
418     all_timespan_lists[voice_name] = timespan_list
419
420 score = abjad.Score([
421     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context'),
422     abjad.StaffGroup(
423         [
424             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
425                         lilypond_type='Staff'),
426             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
427                         lilypond_type='Staff'),
428             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
429                         lilypond_type='Staff'),
430             ],
431         ),
432     ],
433 )
434
435 for time_signature in time_signatures:
436     skip = abjad.Skip(1, multiplier=(time_signature))
437     abjad.attach(time_signature, skip)
438     score['Global Context'].append(skip)
439
440 print('Making containers ...')
441
442 def make_container(music_maker, durations):
443     selections = music_maker(durations)
444     container = abjad.Container([])
445     container.extend(selections)
446     return container
447
448 def key_function(timespan):
449     return timespan.annotation.music_maker or silence_maker
450
451 for voice_name, timespan_list in all_timespan_lists.items():
452     for music_maker, grouper in itertools.groupby(
453         timespan_list,
454         key=key_function,
455     ):
456         durations = [timespan.duration for timespan in grouper]
457         container = make_container(music_maker, durations)
458         voice = score[voice_name]
459         voice.append(container)

```

```

460 print('Splitting and rewriting ...')
461
462 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
463     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
464         time_signature = time_signatures[i]
465         abjad.mutate(shard).rewrite_meter(time_signature)
466
467 print('Beaming runs ...')
468
469 for voice in abjad.select(score).components(abjad.Voice):
470     for run in abjad.select(voice).runs():
471         if 1 < len(run):
472             specifier = abjadext.rmakers.BeamSpecifier(
473                 beam_each_division=False,
474                 )
475             specifier(run)
476             abjad.attach(abjad.StartBeam(), run[0])
477             abjad.attach(abjad.StopBeam(), run[-1])
478             for leaf in run:
479                 if abjad.Duration(1, 4) <= leaf.written_duration:
480                     continue
481                 previous_leaf = abjad.inspect(leaf).leaf(-1)
482                 next_leaf = abjad.inspect(leaf).leaf(1)
483                 if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
484                     abjad.Duration(1, 4) <= next_leaf.written_duration):
485                     left = previous_leaf.written_duration.flag_count
486                     right = leaf.written_duration.flag_count
487                     beam_count = abjad.BeamCount(
488                         left=left,
489                         right=right,
490                         )
491                     abjad.attach(beam_count, leaf)
492                     continue
493                 if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
494                     abjad.Duration(1, 4) <= previous_leaf.written_duration):
495                     left = leaf.written_duration.flag_count
496                     right = next_leaf.written_duration.flag_count
497                     beam_count = abjad.BeamCount(
498                         left=left,
499                         right=right,
500                         )
501                     abjad.attach(beam_count, leaf)
502
503 print('Stopping Hairpins ...')
504 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
505     for rest in abjad.iterate(staff).components(abjad.Rest):
506         previous_leaf = abjad.inspect(rest).leaf(-1)
507         if isinstance(previous_leaf, abjad.Note):
508             abjad.attach(abjad.StopHairpin(), rest)
509         elif isinstance(previous_leaf, abjad.Chord):
510             abjad.attach(abjad.StopHairpin(), rest)
511         elif isinstance(previous_leaf, abjad.Rest):
512             pass
513

```

```

$14 print('Adding attachments ...')
$15 bar_line = abjad.BarLine('|||')
$16 metro = abjad.MetronomeMark((1, 4), 60)
$17 markup1 = abjad.Markup(r'\bold { A }')
$18 markup2 = abjad.Markup(r'\bold { B }')
$19 markup3 = abjad.Markup(r'\bold { C }')
$20 markup4 = abjad.Markup(r'\bold { D }')
$21 markup5 = abjad.Markup(r'\bold { E }')
$22 markup6 = abjad.Markup(r'\bold { F }')
$23 mark1 = abjad.RehearsalMark(markup=markup1)
$24 mark2 = abjad.RehearsalMark(markup=markup2)
$25 mark3 = abjad.RehearsalMark(markup=markup3)
$26 mark4 = abjad.RehearsalMark(markup=markup4)
$27 mark5 = abjad.RehearsalMark(markup=markup5)
$28 mark6 = abjad.RehearsalMark(markup=markup6)
$29
$30 instruments1 = cyc([
$31     abjad.Flute(),
$32     abjad.AltoSaxophone(),
$33     abjad.Cello(),
$34 ])
$35
$36 clefs1 = cyc([
$37     abjad.Clef('treble'),
$38     abjad.Clef('treble'),
$39     abjad.Clef('bass'),
$40 ])
$41
$42 abbreviations1 = cyc([
$43     abjad.MarginMarkup(markup=abjad.Markup('fl.')),
$44     abjad.MarginMarkup(markup=abjad.Markup('sx.')),
$45     abjad.MarginMarkup(markup=abjad.Markup('vc.')),
$46 ])
$47
$48 names1 = cyc([
$49     abjad.StartMarkup(markup=abjad.Markup('Flute')),
$50     abjad.StartMarkup(markup=abjad.Markup('Saxophone')),
$51     abjad.StartMarkup(markup=abjad.Markup('Violoncello')),
$52 ])
$53
$54 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
$55     leaf1 = abjad.select(staff).leaves()[0]
$56     abjad.attach(next(instruments1), leaf1)
$57     abjad.attach(next(abbreviations1), leaf1)
$58     abjad.attach(next(names1), leaf1)
$59     abjad.attach(next(clefs1), leaf1)
$60
$61 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
$62     leaf1 = abjad.select(staff).leaves()[0]
$63     last_leaf = abjad.select(staff).leaves()[-1]
$64     abjad.attach(metro, leaf1)
$65     abjad.attach(bar_line, last_leaf)
$66
$67 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):

```

```

568     abjad.Instrument.transpose_from_sounding_pitch(staff)

569 score_file = abjad.LilyPondFile.new(
570     score,
571     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
572     _stylesheets/abjad.ily'],
573 )
574
575 abjad.SegmentMaker.comment_measure_numbers(score)
576 ######
577
578 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand/
579     Segments/Segment_I'
580 pdf_path = f'{directory}/Segment_I.pdf'
581 path = pathlib.Path('Segment_I.pdf')
582 if path.exists():
583     print(f'Removing {pdf_path} ...')
584     path.unlink()
585 time_1 = time.time()
586 print(f'Persisting {pdf_path} ...')
587 result = abjad.persist(score_file).as_pdf(pdf_path)
588 print(result[0])
589 print(result[1])
590 print(result[2])
591 success = result[3]
592 if success is False:
593     print('LilyPond failed!')
594 time_2 = time.time()
595 total_time = time_2 - time_1
596 print(f'Total time: {total_time} seconds')
597 if path.exists():
598     print(f'Opening {pdf_path} ...')
      os.system(f'open {pdf_path}')

```

**Code Example A.16:** Four Ages of Sand Segment\_I

#### A.4.1.2 SEGMENT\_II

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [

```

```

16         (9, 8), (6, 8), (5, 4), (4, 4), (6, 8),
17         (3, 8), (4, 4), (3, 4), (5, 8), (4, 4),
18         (11, 8),
19     ]
20 ]
21
22 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures
23 ])
24
25 def reduceMod3(rw):
26     return [(x % 4) for x in rw]
27
28 def reduceMod9(rw):
29     return [(x % 10) for x in rw]
30
31 def cyc(lst):
32     count = 0
33     while True:
34         yield lst[count%len(lst)]
35         count += 1
36
37 def grouper(lst1, lst2):
38     def cyc(lst):
39         c = 0
40         while True:
41             yield lst[c%len(lst)]
42             c += 1
43     lst1 = cyc(lst1)
44     return [next(lst1) if i == 1 else [next(lst1) for _ in range(i)] for i in
45     lst2]
46
47 seed(1)
48 flute_random_walk_one = []
49 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
50 for i in range(1, 1000):
51     movement = -1 if random() < 0.5 else 1
52     value = flute_random_walk_one[i-1] + movement
53     flute_random_walk_one.append(value)
54 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
55 flute_chord_one = [0, 11, 14, 17, 18, 22, 18, 17, 14, 11, ]
56 flute_notes_one = [flute_chord_one[x] for x in reduceMod9(
57     flute_random_walk_one)]
58
59 seed(4)
60 saxophone_random_walk_one = []
61 saxophone_random_walk_one.append(-1 if random() < 0.5 else 1)
62 for i in range(1, 1000):
63     movement = -1 if random() < 0.5 else 1
64     value = saxophone_random_walk_one[i-1] + movement
65     saxophone_random_walk_one.append(value)
66 saxophone_random_walk_one = [abs(x) for x in saxophone_random_walk_one]
67 saxophone_chord_one = [-8, -5, -4, 0, 11, 14, 11, 0, -4, -5, ]
68 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod9(
69     saxophone_random_walk_one)]

```

```

66
67 seed(8)
68 cello_random_walk_one = []
69 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
70 for i in range(1, 1000):
71     movement = -1 if random() < 0.5 else 1
72     value = cello_random_walk_one[i-1] + movement
73     cello_random_walk_one.append(value)
74 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
75 cello_chord_one = [-22, -11, -8, -5, -4, 0, -4, -5, -8, -11, ]
76 cello_notes_one = [cello_chord_one[x] for x in reduceMod9(
77     cello_random_walk_one)]
78
79 seed(1)
80 flute_random_walk_two = []
81 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
82 for i in range(1, 1000):
83     movement = -1 if random() < 0.5 else 1
84     value = flute_random_walk_two[i-1] + movement
85     flute_random_walk_two.append(value)
86 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
87 flute_chord_two = [0, 14, 18, 14, ]
88 flute_notes_two = [flute_chord_two[x] for x in reduceMod3(
89     flute_random_walk_two)]
90
91 seed(4)
92 saxophone_random_walk_two = []
93 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
94 for i in range(1, 1000):
95     movement = -1 if random() < 0.5 else 1
96     value = saxophone_random_walk_two[i-1] + movement
97     saxophone_random_walk_two.append(value)
98 saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
99 saxophone_chord_two = [-8, -4, 11, -4, ]
100 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod3(
101     saxophone_random_walk_two)]
102
103 seed(8)
104 cello_random_walk_two = []
105 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
106 for i in range(1, 1000):
107     movement = -1 if random() < 0.5 else 1
108     value = cello_random_walk_two[i-1] + movement
109     cello_random_walk_two.append(value)
110 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
111 cello_chord_two = [-22, -8, -4, -8, ]
112 cello_notes_two = [cello_chord_two[x] for x in reduceMod3(
113     cello_random_walk_two)]
114
115 seed(1)
116 flute_random_walk_three = []
117 flute_random_walk_three.append(-1 if random() < 0.5 else 1)
118 for i in range(1, 1000):
119     movement = -1 if random() < 0.5 else 1
120
121
122
123
124
125

```

```

116     value = flute_random_walk_three[i-1] + movement
117     flute_random_walk_three.append(value)
118 flute_random_walk_three = [abs(x) for x in flute_random_walk_three]
119 flute_chord_three = [11, 17, 22, 17, ]
120 flute_notes_three = [flute_chord_three[x] for x in reduceMod3(
121     flute_random_walk_three)]
122
123 seed(4)
124 saxophone_random_walk_three = []
125 saxophone_random_walk_three.append(-1 if random() < 0.5 else 1)
126 for i in range(1, 1000):
127     movement = -1 if random() < 0.5 else 1
128     value = saxophone_random_walk_three[i-1] + movement
129     saxophone_random_walk_three.append(value)
130 saxophone_random_walk_three = [abs(x) for x in saxophone_random_walk_three]
131 saxophone_chord_three = [-5, -4, 11, -4, ]
132 saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod3(
133     saxophone_random_walk_three)]
134
135 seed(8)
136 cello_random_walk_three = []
137 cello_random_walk_three.append(-1 if random() < 0.5 else 1)
138 for i in range(1, 1000):
139     movement = -1 if random() < 0.5 else 1
140     value = cello_random_walk_three[i-1] + movement
141     cello_random_walk_three.append(value)
142 cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
143 cello_chord_three = [-11, -5, 0, -5, ]
144 cello_notes_three = [cello_chord_three[x] for x in reduceMod3(
145     cello_random_walk_three)]
146
147 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
148     talea=abjadext.rmakers.Talea(
149         counts=[3, 1, 3, 1, 3, 1, 3, 1, 3, 1],
150         denominator=16,
151         ),
152     beamSpecifier=abjadext.rmakers.BeamSpecifier(
153         beamDivisionsTogether=True,
154         beamRests=False,
155         ),
156     extraCountsPerDivision=[1, 0, -1, 0],
157     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
158         leftClasses=[abjad.Note, abjad.Rest],
159         leftCounts=[1, 0, 1],
160         ),
161     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
162         trivialize=True,
163         extractTrivial=True,
164         rewriteRestFilled=True,
165         ),
166     )
167
168 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
169     talea=abjadext.rmakers.Talea(

```

```

167     counts=[1, 2, 2, 3, 3, 3, 2, 2, 1, ],
168     denominator=16,
169   ),
170   beamSpecifier=abjadext.rmakers.BeamSpecifier(
171     beamDivisionsTogether=True,
172     beamRests=False,
173   ),
174   extraCountsPerDivision=[0, -1, 0, 1],
175   burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
176     leftClasses=[abjad.Note, abjad.Rest],
177     leftCounts=[1, 0, 1],
178   ),
179   tupletSpecifier=abjadext.rmakers.TupletSpecifier(
180     trivialize=True,
181     extractTrivial=True,
182     rewriteRestFilled=True,
183   ),
184 )
185
186 rmakerThree = abjadext.rmakers.EvenDivisionRhythmMaker(
187   denominators=[8, 16, 8, 16, 8, 16],
188   extraCountsPerDivision=[-1, 0, 1, -1, 0, 1, 0, 0],
189   burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
190     leftClasses=[abjad.Rest],
191     leftCounts=[1],
192     rightClasses=[abjad.Rest],
193     rightCounts=[1],
194     outerDivisionsOnly=True,
195   ),
196   tupletSpecifier=abjadext.rmakers.TupletSpecifier(
197     trivialize=True,
198     extractTrivial=True,
199     rewriteRestFilled=True,
200   ),
201 )
202
203 attachmentHandlerOne = AttachmentHandler(
204   startingDynamic='f',
205   endingDynamic='mp',
206   hairpinIndicator='--',
207   articulation='accent',
208 )
209
210 attachmentHandlerTwo = AttachmentHandler(
211   startingDynamic='fff',
212   endingDynamic='mf',
213   hairpinIndicator='|>',
214   articulation='tenuto',
215 )
216
217 attachmentHandlerThree = AttachmentHandler(
218   startingDynamic='pp',
219   endingDynamic='mf',
220   hairpinIndicator='<',


```

```

221     articulation=' ' ,
222 )
223
224 ##### oboe#####
225 flutemusicmaker_one = MusicMaker(
226     rmaker=rmaker_one ,
227     pitches=flute_notes_one ,
228     continuous=True ,
229     attachment_handler=attachment_handler_one ,
230 )
231 flutemusicmaker_two = MusicMaker(
232     rmaker=rmaker_two ,
233     pitches=flute_notes_two ,
234     continuous=True ,
235     attachment_handler=attachment_handler_two ,
236 )
237 flutemusicmaker_three = MusicMaker(
238     rmaker=rmaker_three ,
239     pitches=flute_notes_three ,
240     continuous=True ,
241     attachment_handler=attachment_handler_three ,
242 )
243 ##### saxophone#####
244 saxophonemusicmaker_one = MusicMaker(
245     rmaker=rmaker_one ,
246     pitches=saxophone_notes_one ,
247     continuous=True ,
248     attachment_handler=attachment_handler_one ,
249 )
250 saxophonemusicmaker_two = MusicMaker(
251     rmaker=rmaker_two ,
252     pitches=saxophone_notes_two ,
253     continuous=True ,
254     attachment_handler=attachment_handler_two ,
255 )
256 saxophonemusicmaker_three = MusicMaker(
257     rmaker=rmaker_three ,
258     pitches=saxophone_notes_three ,
259     continuous=True ,
260     attachment_handler=attachment_handler_three ,
261 )
262 ##### cello#####
263 cellomusicmaker_one = MusicMaker(
264     rmaker=rmaker_one ,
265     pitches=cello_notes_one ,
266     continuous=True ,
267     attachment_handler=attachment_handler_one ,
268 )
269 cellomusicmaker_two = MusicMaker(
270     rmaker=rmaker_two ,
271     pitches=cello_notes_two ,
272     continuous=True ,
273     attachment_handler=attachment_handler_two ,
274 )

```

```

275 cellomusicmaker_three = MusicMaker(
276     rmaker=rmaker_three,
277     pitches=cello_notes_three,
278     continuous=True,
279     attachment_handler=attachment_handler_three,
280 )
281
282 silence_maker = abjadext.rmakers.NoteRhythmMaker(
283     division_masks=[
284         abjadext.rmakers.SilenceMask(
285             pattern=abjad.index([0], 1),
286         ),
287     ],
288 )
289
290
291 class MusicSpecifier:
292
293     def __init__(self, music_maker, voice_name):
294         self.music_maker = music_maker
295         self.voice_name = voice_name
296
297
298 print('Collecting timespans and rmakers ...')
299 #####group one#####
300 voice_1_timespan_list = abjad.TimespanList([
301     abjad.AnnotatedTimespan(
302         start_offset=start_offset,
303         stop_offset=stop_offset,
304         annotation=MusicSpecifier(
305             music_maker=music_maker,
306             voice_name='Voice 1',
307         ),
308     )
309     for start_offset, stop_offset, music_maker in [
310         [(15, 8), (18, 8), flutemusicmaker_one],
311         [(18, 8), (22, 8), flutemusicmaker_two],
312         [(25, 8), (29, 8), flutemusicmaker_three],
313         [(29, 8), (32, 8), flutemusicmaker_one],
314         [(35, 8), (39, 8), flutemusicmaker_two],
315         [(39, 8), (42, 8), flutemusicmaker_three],
316         [(45, 8), (50, 8), flutemusicmaker_one],
317         [(50, 8), (52, 8), flutemusicmaker_two],
318         [(55, 8), (56, 8), flutemusicmaker_three],
319         [(56, 8), (61, 8), flutemusicmaker_one],
320         [(61, 8), (62, 8), flutemusicmaker_two],
321         [(65, 8), (69, 8), flutemusicmaker_three],
322         [(69, 8), (72, 8), flutemusicmaker_one],
323         [(75, 8), (79, 8), flutemusicmaker_two],
324     ]
325 ])
326
327 voice_3_timespan_list = abjad.TimespanList([
328     abjad.AnnotatedTimespan(

```

```

329     start_offset=start_offset,
330     stop_offset=stop_offset,
331     annotation=MusicSpecifier(
332         music_maker=music_maker,
333         voice_name='Voice 3',
334     ),
335 )
336 for start_offset, stop_offset, music_maker in [
337     [(0, 8), (3, 8), saxophonemusicmaker_one],
338     [(4, 8), (8, 8), saxophonemusicmaker_two],
339     [(10, 8), (12, 8), saxophonemusicmaker_three],
340     [(12, 8), (15, 8), saxophonemusicmaker_one],
341     [(18, 8), (24, 8), saxophonemusicmaker_two],
342     [(28, 8), (33, 8), saxophonemusicmaker_three],
343     [(33, 8), (35, 8), saxophonemusicmaker_one],
344     [(40, 8), (42, 8), saxophonemusicmaker_two],
345     [(42, 8), (44, 8), saxophonemusicmaker_three],
346     [(44, 8), (48, 8), saxophonemusicmaker_one],
347     [(54, 8), (55, 8), saxophonemusicmaker_two],
348     [(62, 8), (64, 8), saxophonemusicmaker_three],
349     [(72, 8), (75, 8), saxophonemusicmaker_one],
350     [(76, 8), (79, 8), saxophonemusicmaker_two],
351     [(79, 8), (80, 8), silence_maker],
352 ]
353 ]
354
355 voice_8_timespan_list = abjad.TimespanList([
356     abjad.AnnotatedTimespan(
357         start_offset=start_offset,
358         stop_offset=stop_offset,
359         annotation=MusicSpecifier(
360             music_maker=music_maker,
361             voice_name='Voice 8',
362         ),
363     )
364     for start_offset, stop_offset, music_maker in [
365         [(9, 8), (12, 8), cellomusicmaker_one],
366         [(20, 8), (24, 8), cellomusicmaker_two],
367         [(31, 8), (33, 8), cellomusicmaker_three],
368         [(33, 8), (36, 8), cellomusicmaker_one],
369         [(42, 8), (48, 8), cellomusicmaker_two],
370         [(53, 8), (56, 8), cellomusicmaker_three],
371         [(56, 8), (60, 8), cellomusicmaker_one],
372         [(64, 8), (69, 8), cellomusicmaker_two],
373         [(69, 8), (72, 8), cellomusicmaker_three],
374         [(75, 8), (79, 8), cellomusicmaker_one],
375     ]
376 ]
377
378 all_timespan_lists = {
379     'Voice 1': voice_1_timespan_list,
380     'Voice 3': voice_3_timespan_list,
381     'Voice 8': voice_8_timespan_list,
382 }

```

```

383
384
385 global_timespan = abjad.Timespan(
386     start_offset=0,
387     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
388 )
389
390 for voice_name, timespan_list in all_timespan_lists.items():
391     silences = abjad.TimespanList([global_timespan])
392     silences.extend(timespan_list)
393     silences.sort()
394     silences.compute_logical_xor()
395     for silence_timespan in silences:
396         timespan_list.append(
397             abjad.AnnotatedTimespan(
398                 start_offset=silence_timespan.start_offset,
399                 stop_offset=silence_timespan.stop_offset,
400                 annotation=MusicSpecifier(
401                     music_maker=None,
402                     voice_name=voice_name,
403                 ),
404             )
405         )
406     timespan_list.sort()
407
408 for voice_name, timespan_list in all_timespan_lists.items():
409     shards = timespan_list.split_at_offsets(bounds)
410     split_timespan_list = abjad.TimespanList()
411     for shard in shards:
412         split_timespan_list.extend(shard)
413     split_timespan_list.sort()
414     all_timespan_lists[voice_name] = timespan_list
415
416 score = abjad.Score([
417     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context'),
418     abjad.StaffGroup(
419         [
420             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
421             lilypond_type='Staff'),
422             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
423             lilypond_type='Staff'),
424             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
425             lilypond_type='Staff'),
426         ],
427     )
428
429 for time_signature in time_signatures:
430     skip = abjad.Skip(1, multiplier=(time_signature))
431     abjad.attach(time_signature, skip)
432     score['Global Context'].append(skip)
433

```

```

434 print('Making containers ...')
435
436 def make_container(music_maker, durations):
437     selections = music_maker(durations)
438     container = abjad.Container([])
439     container.extend(selections)
440     return container
441
442 def key_function(timespan):
443     return timespan.annotation.music_maker or silence_maker
444
445 for voice_name, timespan_list in all_timespan_lists.items():
446     for music_maker, grouper in itertools.groupby(
447         timespan_list,
448         key=key_function,
449     ):
450         durations = [timespan.duration for timespan in grouper]
451         container = make_container(music_maker, durations)
452         voice = score[voice_name]
453         voice.append(container)
454
455 print('Splitting and rewriting ...')
456
457 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
458     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
459         time_signature = time_signatures[i]
460         abjad.mutate(shard).rewrite_meter(time_signature)
461
462 print('Beaming runs ...')
463
464 for voice in abjad.select(score).components(abjad.Voice):
465     for run in abjad.select(voice).runs():
466         if 1 < len(run):
467             specifier = abjadext.rmakers.BeamSpecifier(
468                 beam_each_division=False,
469                 )
470             specifier(run)
471             abjad.attach(abjad.StartBeam(), run[0])
472             abjad.attach(abjad.StopBeam(), run[-1])
473             for leaf in run:
474                 if abjad.Duration(1, 4) <= leaf.written_duration:
475                     continue
476                 previous_leaf = abjad.inspect(leaf).leaf(-1)
477                 next_leaf = abjad.inspect(leaf).leaf(1)
478                 if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
479                     abjad.Duration(1, 4) <= next_leaf.written_duration):
480                     left = previous_leaf.written_duration.flag_count
481                     right = leaf.written_duration.flag_count
482                     beam_count = abjad.BeamCount(
483                         left=left,
484                         right=right,
485                         )
486                     abjad.attach(beam_count, leaf)
487                     continue

```

```

488         if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
489             abjad.Duration(1, 4) <= previous_leaf.written_duration):
490             left = leaf.written_duration.flag_count
491             right = next_leaf.written_duration.flag_count
492             beam_count = abjad.BeamCount(
493                 left=left,
494                 right=right,
495             )
496             abjad.attach(beam_count, leaf)
497
498 print('Stopping Hairpins ...')
499 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
500     for rest in abjad.iterate(staff).components(abjad.Rest):
501         previous_leaf = abjad.inspect(rest).leaf(-1)
502         if isinstance(previous_leaf, abjad.Note):
503             abjad.attach(abjad.StopHairpin(), rest)
504         elif isinstance(previous_leaf, abjad.Chord):
505             abjad.attach(abjad.StopHairpin(), rest)
506         elif isinstance(previous_leaf, abjad.Rest):
507             pass
508
509 print('Adding attachments ...')
510 bar_line = abjad.BarLine('|||')
511 metro = abjad.MetronomeMark((1, 4), 60)
512 markup1 = abjad.Markup(r'\bold { A }')
513 markup2 = abjad.Markup(r'\bold { B }')
514 markup3 = abjad.Markup(r'\bold { C }')
515 markup4 = abjad.Markup(r'\bold { D }')
516 markup5 = abjad.Markup(r'\bold { E }')
517 markup6 = abjad.Markup(r'\bold { F }')
518 mark1 = abjad.RehearsalMark(markup=markup1)
519 mark2 = abjad.RehearsalMark(markup=markup2)
520 mark3 = abjad.RehearsalMark(markup=markup3)
521 mark4 = abjad.RehearsalMark(markup=markup4)
522 mark5 = abjad.RehearsalMark(markup=markup5)
523 mark6 = abjad.RehearsalMark(markup=markup6)
524
525 def _apply_numerators_and_tech(staff, nums, tech):
526     numerators = cyc(nums)
527     techs = cyc(tech)
528     for logical_tie in abjad.select(staff).logical_ties(pitched=True):
529         tech = next(techs)
530         numerator = next(numerators)
531         bcp = abjad.BowContactPoint((numerator, 5))
532         technis = abjad.BowMotionTechnique(tech)
533         for note in logical_tie:
534             abjad.attach(bcp, note)
535             abjad.attach(technis, note)
536     for run in abjad.select(staff).runs():
537         abjad.bow_contact_spanner(run, omit_bow_changes=False)
538
539 instruments1 = cyc([
540     abjad.Flute(),
541     abjad.AltoSaxophone(),

```

```

542     abjad.Cello(),
543 ])
544
545 clefs1 = cyc([
546     abjad.Clef('treble'),
547     abjad.Clef('treble'),
548     abjad.Clef('bass'),
549 ])
550
551 abbreviations1 = cyc([
552     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),
553     abjad.MarginMarkup(markup=abjad.Markup('sx.'),),
554     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
555 ])
556
557 names1 = cyc([
558     abjad.StartMarkup(markup=abjad.Markup('Flute'),),
559     abjad.StartMarkup(markup=abjad.Markup('Saxophone'),),
560     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
561 ])
562
563 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
564     leaf1 = abjad.select(staff).leaves()[0]
565     abjad.attach(next(instruments1), leaf1)
566     abjad.attach(next(abbreviations1), leaf1)
567     abjad.attach(next(names1), leaf1)
568     abjad.attach(next(clefs1), leaf1)
569
570 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
571     leaf1 = abjad.select(staff).leaves()[0]
572     last_leaf = abjad.select(staff).leaves()[-1]
573     #abjad.attach(metro, leaf1)
574     abjad.attach(bar_line, last_leaf)
575
576 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
577     abjad.Instrument.transpose_from_sounding_pitch(staff)
578
579 score_file = abjad.LilyPondFile.new(
580     score,
581     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
582     _stylesheets/abjad.ily'],
583 )
584
585 abjad.SegmentMaker.comment_measure_numbers(score)
586 ######
587
588 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand/
589     Segments/Segment_II'
590 pdf_path = f'{directory}/Segment_II.pdf'
591 path = pathlib.Path('Segment_II.pdf')
592 if path.exists():
593     print(f'Removing {pdf_path} ...')
594     path.unlink()
595 time_1 = time.time()

```

```

594 print(f'Persisting {pdf_path} ...')
595 result = abjad.persist(score_file).as_pdf(pdf_path)
596 print(result[0])
597 print(result[1])
598 print(result[2])
599 success = result[3]
600 if success is False:
601     print('LilyPond failed!')
602 time_2 = time.time()
603 total_time = time_2 - time_1
604 print(f'Total time: {total_time} seconds')
605 if path.exists():
606     print(f'Opening {pdf_path} ...')
607     os.system(f'open {pdf_path}')

```

**Code Example A.17:** Four Ages of Sand Segment\_II

#### A.4.1.3 SEGMENT\_III

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (9, 8), (6, 8), (5, 4), (4, 4), (6, 8),
17         (3, 8), (4, 4), (3, 4), (5, 8), (4, 4),
18         (11, 8),
19     ]
20 ]
21
22 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures
23 ])
24
25 def reduceMod1(rw):
26     return [(x % 2) for x in rw]
27
28 def reduceMod3(rw):
29     return [(x % 4) for x in rw]
30
31 def reduceMod5(rw):
32     return [(x % 6) for x in rw]
33 def reduceMod7(rw):

```

```

34     return [(x % 8) for x in rw]
35
36 def cyc(lst):
37     count = 0
38     while True:
39         yield lst[count%len(lst)]
40         count += 1
41
42 def grouper(lst1, lst2):
43     def cyc(lst):
44         c = 0
45         while True:
46             yield lst[c%len(lst)]
47             c += 1
48     lst1 = cyc(lst1)
49     return [next(lst1) if i == 1 else [next(lst1) for _ in range(i)] for i in
50     lst2]
51
52 seed(1)
53 flute_random_walk_one = []
54 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
55 for i in range(1, 1000):
56     movement = -1 if random() < 0.5 else 1
57     value = flute_random_walk_one[i-1] + movement
58     flute_random_walk_one.append(value)
59 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
60 flute_chord_one = [2, 11, 12, 20, 31, 20, 12, 11, ]
61 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
62     flute_random_walk_one)]
63
64 seed(4)
65 saxophone_random_walk_one = []
66 saxophone_random_walk_one.append(-1 if random() < 0.5 else 1)
67 for i in range(1, 1000):
68     movement = -1 if random() < 0.5 else 1
69     value = saxophone_random_walk_one[i-1] + movement
70     saxophone_random_walk_one.append(value)
71 saxophone_random_walk_one = [abs(x) for x in saxophone_random_walk_one]
72 saxophone_chord_one = [-10, 2, 11, 12, 1, 2, ]
73 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod5(
74     saxophone_random_walk_one)]
75
76 seed(8)
77 cello_random_walk_one = []
78 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
79 for i in range(1, 1000):
80     movement = -1 if random() < 0.5 else 1
81     value = cello_random_walk_one[i-1] + movement
82     cello_random_walk_one.append(value)
83 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
84 cello_chord_one = [-18, -10, 2, -10, ]
85 cello_notes_one = [cello_chord_one[x] for x in reduceMod3(
86     cello_random_walk_one)]

```

```

84 seed(1)
85 flute_random_walk_two = []
86 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
87 for i in range(1, 1000):
88     movement = -1 if random() < 0.5 else 1
89     value = flute_random_walk_two[i-1] + movement
90     flute_random_walk_two.append(value)
91 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
92 flute_chord_two = [2, 12, 31, 12, ]
93 flute_notes_two = [flute_chord_two[x] for x in reduceMod3(
94     flute_random_walk_two)]
95
96 seed(4)
97 saxophone_random_walk_two = []
98 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
99 for i in range(1, 1000):
100    movement = -1 if random() < 0.5 else 1
101    value = saxophone_random_walk_two[i-1] + movement
102    saxophone_random_walk_two.append(value)
103 saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
104 saxophone_chord_two = [11, 20, ]
105 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod1(
106     saxophone_random_walk_two)]
107
108 seed(8)
109 cello_random_walk_two = []
110 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
111 for i in range(1, 1000):
112     movement = -1 if random() < 0.5 else 1
113     value = cello_random_walk_two[i-1] + movement
114     cello_random_walk_two.append(value)
115 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
116 cello_chord_two = [-18, 2, ]
117 cello_notes_two = [cello_chord_two[x] for x in reduceMod1(
118     cello_random_walk_two)]
119
120 seed(1)
121 flute_random_walk_three = []
122 flute_random_walk_three.append(-1 if random() < 0.5 else 1)
123 for i in range(1, 1000):
124     movement = -1 if random() < 0.5 else 1
125     value = flute_random_walk_three[i-1] + movement
126     flute_random_walk_three.append(value)
127 flute_random_walk_three = [abs(x) for x in flute_random_walk_three]
128 flute_chord_three = [11, 20, ]
129 flute_notes_three = [flute_chord_three[x] for x in reduceMod1(
130     flute_random_walk_three)]
131
132 seed(4)
133 saxophone_random_walk_three = []
134 saxophone_random_walk_three.append(-1 if random() < 0.5 else 1)
135 for i in range(1, 1000):
136     movement = -1 if random() < 0.5 else 1
137     value = saxophone_random_walk_three[i-1] + movement

```

```

134     saxophone_random_walk_three.append(value)
135 saxophone_random_walk_three = [abs(x) for x in saxophone_random_walk_three]
136 saxophone_chord_three = [2, 12, ]
137 saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod1(
138     saxophone_random_walk_three)]
139
139 seed(8)
140 cello_random_walk_three = []
141 cello_random_walk_three.append(-1 if random() < 0.5 else 1)
142 for i in range(1, 1000):
143     movement = -1 if random() < 0.5 else 1
144     value = cello_random_walk_three[i-1] + movement
145     cello_random_walk_three.append(value)
146 cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
147 cello_notes_three = [-10, ]
148
149 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
150     talea=abjadext.rmakers.Talea(
151         counts=[2, 1, 3, 2, 2, 3, 1, ],
152         denominator=16,
153         ),
154     beamSpecifier=abjadext.rmakers.BeamSpecifier(
155         beamDivisionsTogether=True,
156         beamRests=False,
157         ),
158     extraCountsPerDivision=[0, 1, 0, -1],
159     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
160         leftClasses=[abjad.Note, abjad.Rest],
161         leftCounts=[1, 0, 1],
162         ),
163     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
164         trivialize=True,
165         extractTrivial=True,
166         rewriteRestFilled=True,
167         ),
168 )
169
170 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
171     talea=abjadext.rmakers.Talea(
172         counts=[1, 2, 2, 3, 3, 3, 2, 2, 1, ],
173         denominator=16,
174         ),
175     beamSpecifier=abjadext.rmakers.BeamSpecifier(
176         beamDivisionsTogether=True,
177         beamRests=False,
178         ),
179     extraCountsPerDivision=[1, 0, -1, 0],
180     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
181         leftClasses=[abjad.Note, abjad.Rest],
182         leftCounts=[1, 0, 1],
183         ),
184     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
185         trivialize=True,
186         extractTrivial=True,

```

```

187         rewrite_rest_filled=True,
188     ),
189 )
190
191 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
192     denominators=[16, 16, 8, 16, 16, 8],
193     extra_counts_per_division=[1, 0, 0, -1, 0, 1, -1, 0],
194     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
195         left_classes=[abjad.Rest],
196         left_counts=[1],
197         right_classes=[abjad.Rest],
198         right_counts=[1],
199         outer_divisions_only=True,
200     ),
201     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
202         trivialize=True,
203         extract_trivial=True,
204         rewrite_rest_filled=True,
205     ),
206 )
207
208 attachment_handler_one = AttachmentHandler(
209     starting_dynamic='p',
210     ending_dynamic='mp',
211     hairpin_indicator='--',
212     articulation='accent',
213 )
214
215 attachment_handler_two = AttachmentHandler(
216     starting_dynamic='fff',
217     ending_dynamic='mf',
218     hairpin_indicator='>',
219     articulation='tenuto',
220 )
221
222 attachment_handler_three = AttachmentHandler(
223     starting_dynamic='mp',
224     ending_dynamic='ff',
225     hairpin_indicator='<|',
226     articulation='',
227 )
228
229 ##### oboe#####
230 flutemusicmaker_one = MusicMaker(
231     rmaker=rmaker_one,
232     pitches=flute_notes_one,
233     continuous=True,
234     attachment_handler=attachment_handler_one,
235 )
236 flutemusicmaker_two = MusicMaker(
237     rmaker=rmaker_two,
238     pitches=flute_notes_two,
239     continuous=True,
240     attachment_handler=attachment_handler_two,

```

```

241 )
242 flutemusicmaker_three = MusicMaker(
243     rmaker=rmaker_three,
244     pitches=flute_notes_three,
245     continuous=True,
246     attachment_handler=attachment_handler_three,
247 )
248 #####saxophone#####
249 saxophonemusicmaker_one = MusicMaker(
250     rmaker=rmaker_one,
251     pitches=saxophone_notes_one,
252     continuous=True,
253     attachment_handler=attachment_handler_one,
254 )
255 saxophonemusicmaker_two = MusicMaker(
256     rmaker=rmaker_two,
257     pitches=saxophone_notes_two,
258     continuous=True,
259     attachment_handler=attachment_handler_two,
260 )
261 saxophonemusicmaker_three = MusicMaker(
262     rmaker=rmaker_three,
263     pitches=saxophone_notes_three,
264     continuous=True,
265     attachment_handler=attachment_handler_three,
266 )
267 #####cello#####
268 cellomusicmaker_one = MusicMaker(
269     rmaker=rmaker_one,
270     pitches=cello_notes_one,
271     continuous=True,
272     attachment_handler=attachment_handler_one,
273 )
274 cellomusicmaker_two = MusicMaker(
275     rmaker=rmaker_two,
276     pitches=cello_notes_two,
277     continuous=True,
278     attachment_handler=attachment_handler_two,
279 )
280 cellomusicmaker_three = MusicMaker(
281     rmaker=rmaker_three,
282     pitches=cello_notes_three,
283     continuous=True,
284     attachment_handler=attachment_handler_three,
285 )
286
287 silence_maker = abjadext.rmakers.NoteRhythmMaker(
288     division_masks=[
289         abjadext.rmakers.SilenceMask(
290             pattern=abjad.index([0], 1),
291             ),
292         ],
293     )
294

```

```

295
296 class MusicSpecifier:
297
298     def __init__(self, music_maker, voice_name):
299         self.music_maker = music_maker
300         self.voice_name = voice_name
301
302     print('Collecting timespans and rmakers ...')
303     ###group one###
304     voice_1_timespan_list = abjad.TimespanList([
305         abjad.AnnotatedTimespan(
306             start_offset=start_offset,
307             stop_offset=stop_offset,
308             annotation=MusicSpecifier(
309                 music_maker=music_maker,
310                 voice_name='Voice 1',
311             ),
312         ),
313         for start_offset, stop_offset, music_maker in [
314             [(9, 8), (12, 8), flutemusicmaker_one],
315             [(20, 8), (24, 8), flutemusicmaker_two],
316             [(31, 8), (33, 8), flutemusicmaker_three],
317             [(33, 8), (36, 8), flutemusicmaker_one],
318             [(42, 8), (48, 8), flutemusicmaker_two],
319             [(53, 8), (56, 8), flutemusicmaker_three],
320             [(56, 8), (60, 8), flutemusicmaker_one],
321             [(64, 8), (69, 8), flutemusicmaker_two],
322             [(69, 8), (72, 8), flutemusicmaker_three],
323             [(75, 8), (79, 8), flutemusicmaker_one],
324         ]
325     ])
326
327     voice_3_timespan_list = abjad.TimespanList([
328         abjad.AnnotatedTimespan(
329             start_offset=start_offset,
330             stop_offset=stop_offset,
331             annotation=MusicSpecifier(
332                 music_maker=music_maker,
333                 voice_name='Voice 3',
334             ),
335         ),
336         for start_offset, stop_offset, music_maker in [
337             [(15, 8), (18, 8), saxophonemusicmaker_one],
338             [(18, 8), (22, 8), saxophonemusicmaker_two],
339             [(25, 8), (29, 8), saxophonemusicmaker_three],
340             [(29, 8), (32, 8), saxophonemusicmaker_one],
341             [(35, 8), (39, 8), saxophonemusicmaker_two],
342             [(39, 8), (42, 8), saxophonemusicmaker_three],
343             [(45, 8), (50, 8), saxophonemusicmaker_one],
344             [(50, 8), (52, 8), saxophonemusicmaker_two],
345             [(55, 8), (56, 8), saxophonemusicmaker_three],
346             [(56, 8), (61, 8), saxophonemusicmaker_one],
347             [(61, 8), (62, 8), saxophonemusicmaker_two],
348             [(65, 8), (69, 8), saxophonemusicmaker_three],

```

```

349     [(69, 8), (72, 8), saxophonemusicmaker_one],
350     [(75, 8), (79, 8), saxophonemusicmaker_two],
351 ]
352 ])
353
354 voice_8_timespan_list = abjad.TimespanList([
355     abjad.AnnotatedTimespan(
356         start_offset=start_offset,
357         stop_offset=stop_offset,
358         annotation=MusicSpecifier(
359             music_maker=music_maker,
360             voice_name='Voice 8',
361         ),
362     )
363     for start_offset, stop_offset, music_maker in [
364         [(0, 8), (3, 8), cellomusicmaker_one],
365         [(4, 8), (8, 8), cellomusicmaker_two],
366         [(10, 8), (12, 8), cellomusicmaker_three],
367         [(12, 8), (15, 8), cellomusicmaker_one],
368         [(18, 8), (24, 8), cellomusicmaker_two],
369         [(28, 8), (33, 8), cellomusicmaker_three],
370         [(33, 8), (35, 8), cellomusicmaker_one],
371         [(40, 8), (42, 8), cellomusicmaker_two],
372         [(42, 8), (44, 8), cellomusicmaker_three],
373         [(44, 8), (48, 8), cellomusicmaker_one],
374         [(54, 8), (55, 8), cellomusicmaker_two],
375         [(62, 8), (64, 8), cellomusicmaker_three],
376         [(72, 8), (75, 8), cellomusicmaker_one],
377         [(76, 8), (79, 8), cellomusicmaker_two],
378         [(79, 8), (80, 8), silence_maker],
379     ]
380 ])
381
382 all_timespan_lists = {
383     'Voice 1': voice_1_timespan_list,
384     'Voice 3': voice_3_timespan_list,
385     'Voice 8': voice_8_timespan_list,
386 }
387
388 global_timespan = abjad.Timespan(
389     start_offset=0,
390     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
391 )
392
393 for voice_name, timespan_list in all_timespan_lists.items():
394     silences = abjad.TimespanList([global_timespan])
395     silences.extend(timespan_list)
396     silences.sort()
397     silences.compute_logical_xor()
398     for silence_timespan in silences:
399         timespan_list.append(
400             abjad.AnnotatedTimespan(
401                 start_offset=silence_timespan.start_offset,
402                 stop_offset=silence_timespan.stop_offset,

```

```

403         annotation=MusicSpecifier(
404             music_maker=None,
405             voice_name=voice_name,
406         ),
407     )
408
409     timespan_list.sort()
410
411 for voice_name, timespan_list in all_timespan_lists.items():
412     shards = timespan_list.split_at_offsets(bounds)
413     split_timespan_list = abjad.TimespanList()
414     for shard in shards:
415         split_timespan_list.extend(shard)
416     split_timespan_list.sort()
417     all_timespan_lists[voice_name] = timespan_list
418
419 score = abjad.Score([
420     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context'),
421     abjad.StaffGroup(
422         [
423             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
424             lilypond_type='Staff'),
425             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
426             lilypond_type='Staff'),
427             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
428             lilypond_type='Staff'),
429         ],
430         name='Staff Group 1',
431     ),
432 ],
433 )
434
435 for time_signature in time_signatures:
436     skip = abjad.Skip(1, multiplier=(time_signature))
437     abjad.attach(time_signature, skip)
438     score['Global Context'].append(skip)
439
440 print('Making containers ...')
441
442 def make_container(music_maker, durations):
443     selections = music_maker(durations)
444     container = abjad.Container([])
445     container.extend(selections)
446     return container
447
448 def key_function(timespan):
449     return timespan.annotation.music_maker or silence_maker
450
451 for voice_name, timespan_list in all_timespan_lists.items():
452     for music_maker, grouper in itertools.groupby(
453         timespan_list,
454         key=key_function,
455     ):
456         durations = [timespan.duration for timespan in grouper]

```

```

454     container = make_container(music_maker, durations)
455     voice = score[voice_name]
456     voice.append(container)
457
458 print('Splitting and rewriting ...')
459
460 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
461     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
462         time_signature = time_signatures[i]
463         abjad.mutate(shard).rewrite_meter(time_signature)
464
465 print('Beaming runs ...')
466
467 for voice in abjad.select(score).components(abjad.Voice):
468     for run in abjad.select(voice).runs():
469         if 1 < len(run):
470             specifier = abjadext.rmakers.BeamSpecifier(
471                 beam_each_division=False,
472                 )
473             specifier(run)
474             abjad.attach(abjad.StartBeam(), run[0])
475             abjad.attach(abjad.StopBeam(), run[-1])
476             for leaf in run:
477                 if abjad.Duration(1, 4) <= leaf.written_duration:
478                     continue
479                 previous_leaf = abjad.inspect(leaf).leaf(-1)
480                 next_leaf = abjad.inspect(leaf).leaf(1)
481                 if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
482                     abjad.Duration(1, 4) <= next_leaf.written_duration):
483                     left = previous_leaf.written_duration.flag_count
484                     right = leaf.written_duration.flag_count
485                     beam_count = abjad.BeamCount(
486                         left=left,
487                         right=right,
488                         )
489                     abjad.attach(beam_count, leaf)
490                     continue
491                 if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
492                     abjad.Duration(1, 4) <= previous_leaf.written_duration):
493                     left = leaf.written_duration.flag_count
494                     right = next_leaf.written_duration.flag_count
495                     beam_count = abjad.BeamCount(
496                         left=left,
497                         right=right,
498                         )
499                     abjad.attach(beam_count, leaf)
500
501 print('Stopping Hairpins ...')
502 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
503     for rest in abjad.iterate(staff).components(abjad.Rest):
504         previous_leaf = abjad.inspect(rest).leaf(-1)
505         if isinstance(previous_leaf, abjad.Note):
506             abjad.attach(abjad.StopHairpin(), rest)
507         elif isinstance(previous_leaf, abjad.Chord):

```

```

508         abjad.attach(abjad.StopHairpin(), rest)
509     elif isinstance(previous_leaf, abjad.Rest):
510         pass
511
512     print('Adding attachments ...')
513     bar_line = abjad.BarLine('|||')
514     metro = abjad.MetronomeMark((1, 4), 60)
515     markup1 = abjad.Markup(r'\bold { A }')
516     markup2 = abjad.Markup(r'\bold { B }')
517     markup3 = abjad.Markup(r'\bold { C }')
518     markup4 = abjad.Markup(r'\bold { D }')
519     markup5 = abjad.Markup(r'\bold { E }')
520     markup6 = abjad.Markup(r'\bold { F }')
521     mark1 = abjad.RehearsalMark(markup=markup1)
522     mark2 = abjad.RehearsalMark(markup=markup2)
523     mark3 = abjad.RehearsalMark(markup=markup3)
524     mark4 = abjad.RehearsalMark(markup=markup4)
525     mark5 = abjad.RehearsalMark(markup=markup5)
526     mark6 = abjad.RehearsalMark(markup=markup6)
527
528 def _apply_numerators_and_tech(staff, nums, tech):
529     numerators = cyc(nums)
530     techs = cyc(tech)
531     for logical_tie in abjad.select(staff).logical_ties(pitched=True):
532         tech = next(techs)
533         numerator = next(numerators)
534         bcp = abjad.BowContactPoint((numerator, 5))
535         technis = abjad.BowMotionTechnique(tech)
536         for note in logical_tie:
537             abjad.attach(bcp, note)
538             abjad.attach(technis, note)
539     for run in abjad.select(staff).runs():
540         abjad.bow_contact_spanner(run, omit_bow_changes=False)
541
542 instruments1 = cyc([
543     abjad.Flute(),
544     abjad.AltoSaxophone(),
545     abjad.Cello(),
546 ])
547
548 clefs1 = cyc([
549     abjad.Clef('treble'),
550     abjad.Clef('treble'),
551     abjad.Clef('bass'),
552 ])
553
554 abbreviations1 = cyc([
555     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),
556     abjad.MarginMarkup(markup=abjad.Markup('sx.'),),
557     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
558 ])
559
560 names1 = cyc([
561     abjad.StartMarkup(markup=abjad.Markup('Flute'),),

```

```

562 abjad.StartMarkup(markup=abjad.Markup('Saxophone'),),
563 abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
564 ])
565
566 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
567     leaf1 = abjad.select(staff).leaves()[0]
568     abjad.attach(next(instruments1), leaf1)
569     abjad.attach(next(abbreviations1), leaf1)
570     abjad.attach(next(names1), leaf1)
571     abjad.attach(next(clefs1), leaf1)
572
573 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
574     leaf1 = abjad.select(staff).leaves()[0]
575     last_leaf = abjad.select(staff).leaves()[-1]
576     #abjad.attach(metro, leaf1)
577     abjad.attach(bar_line, last_leaf)
578
579 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
580     abjad.Instrument.transpose_from_sounding_pitch(staff)
581
582 score_file = abjad.LilyPondFile.new(
583     score,
584     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
585     _stylesheets/abjad.ily'],
586 )
587
588 abjad.SegmentMaker.comment_measure_numbers(score)
589 ######
590 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand/
591     Segments/Segment_III'
592 pdf_path = f'{directory}/Segment_III.pdf'
593 path = pathlib.Path('Segment_III.pdf')
594 if path.exists():
595     print(f'Removing {pdf_path} ...')
596     path.unlink()
597 time_1 = time.time()
598 print(f'Persisting {pdf_path} ...')
599 result = abjad.persist(score_file).as_pdf(pdf_path)
600 print(result[0])
601 print(result[1])
602 print(result[2])
603 success = result[3]
604 if success is False:
605     print('LilyPond failed!')
606 time_2 = time.time()
607 total_time = time_2 - time_1
608 print(f'Total time: {total_time} seconds')
609 if path.exists():
610     print(f'Opening {pdf_path} ...')
611     os.system(f'open {pdf_path}')

```

**Code Example A.18:** Four Ages of Sand Segment\_III

#### A.4.1.4 SEGMENT\_IV

```

1 import abjad
2 import itertools
3 import os
4 import pathlib
5 import time
6 import abjadext.rmakers
7 from MusicMaker import MusicMaker
8 from AttachmentHandler import AttachmentHandler
9 from random import random
10 from random import seed
11
12 print('Interpreting file ...')
13
14 time_signatures = [
15     abjad.TimeSignature(pair) for pair in [
16         (9, 8), (6, 8), (5, 4), (4, 4), (6, 8),
17         (3, 8), (4, 4), (3, 4), (5, 8), (4, 4),
18         (11, 8),
19     ]
20 ]
21
22 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in time_signatures
23     ])
24
25 def reduceMod1(rw):
26     return [(x % 2) for x in rw]
27
28 def reduceMod3(rw):
29     return [(x % 4) for x in rw]
30
31 def reduceMod5(rw):
32     return [(x % 6) for x in rw]
33
34 def reduceMod7(rw):
35     return [(x % 8) for x in rw]
36
37 def cyc(lst):
38     count = 0
39     while True:
40         yield lst[count%len(lst)]
41         count += 1
42
43 def grouper(lst1, lst2):
44     def cyc(lst):
45         c = 0
46         while True:
47             yield lst[c%len(lst)]
48             c += 1
49     lst1 = cyc(lst1)
50     return [next(lst1) if i == 1 else [next(lst1) for _ in range(i)] for i in

```

```

1st2]

50
51 seed(1)
52 flute_random_walk_one = []
53 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
54 for i in range(1, 1000):
55     movement = -1 if random() < 0.5 else 1
56     value = flute_random_walk_one[i-1] + movement
57     flute_random_walk_one.append(value)
58 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
59 flute_chord_one = [2, 13, 16, 20, 31, 20, 16, 13, ]
60 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
61     flute_random_walk_one)]

62 seed(4)
63 saxophone_random_walk_one = []
64 saxophone_random_walk_one.append(-1 if random() < 0.5 else 1)
65 for i in range(1, 1000):
66     movement = -1 if random() < 0.5 else 1
67     value = saxophone_random_walk_one[i-1] + movement
68     saxophone_random_walk_one.append(value)
69 saxophone_random_walk_one = [abs(x) for x in saxophone_random_walk_one]
70 saxophone_chord_one = [-3, 2, 13, 16, 13, 2, ]
71 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod5(
72     saxophone_random_walk_one)]

73 seed(8)
74 cello_random_walk_one = []
75 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
76 for i in range(1, 1000):
77     movement = -1 if random() < 0.5 else 1
78     value = cello_random_walk_one[i-1] + movement
79     cello_random_walk_one.append(value)
80 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
81 cello_chord_one = [-21, -18, -14, -3, 2, -3, -14, -18, ]
82 cello_notes_one = [cello_chord_one[x] for x in reduceMod7(
83     cello_random_walk_one)]

84 seed(1)
85 flute_random_walk_two = []
86 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
87 for i in range(1, 1000):
88     movement = -1 if random() < 0.5 else 1
89     value = flute_random_walk_two[i-1] + movement
90     flute_random_walk_two.append(value)
91 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
92 flute_chord_two = [2, 16, 31, 16, ]
93 flute_notes_two = [flute_chord_two[x] for x in reduceMod3(
94     flute_random_walk_two)]

95 seed(4)
96 saxophone_random_walk_two = []
97 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
98 for i in range(1, 1000):

```

```

99     movement = -1 if random() < 0.5 else 1
100    value = saxophone_random_walk_two[i-1] + movement
101    saxophone_random_walk_two.append(value)
102    saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
103    saxophone_chord_two = [-3, 13, ]
104    saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod1(
105        saxophone_random_walk_two)]
106
106 seed(8)
107 cello_random_walk_two = []
108 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
109 for i in range(1, 1000):
110     movement = -1 if random() < 0.5 else 1
111     value = cello_random_walk_two[i-1] + movement
112     cello_random_walk_two.append(value)
113    cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
114    cello_chord_two = [-21, -14, 2, -14, ]
115    cello_notes_two = [cello_chord_two[x] for x in reduceMod3(
116        cello_random_walk_two)]
116
117 seed(1)
118 flute_random_walk_three = []
119 flute_random_walk_three.append(-1 if random() < 0.5 else 1)
120 for i in range(1, 1000):
121     movement = -1 if random() < 0.5 else 1
122     value = flute_random_walk_three[i-1] + movement
123     flute_random_walk_three.append(value)
124    flute_random_walk_three = [abs(x) for x in flute_random_walk_three]
125    flute_chord_three = [13, 20, ]
126    flute_notes_three = [flute_chord_three[x] for x in reduceMod1(
127        flute_random_walk_three)]
127
128 seed(4)
129 saxophone_random_walk_three = []
130 saxophone_random_walk_three.append(-1 if random() < 0.5 else 1)
131 for i in range(1, 1000):
132     movement = -1 if random() < 0.5 else 1
133     value = saxophone_random_walk_three[i-1] + movement
134     saxophone_random_walk_three.append(value)
135    saxophone_random_walk_three = [abs(x) for x in saxophone_random_walk_three]
136    saxophone_chord_three = [2, 16, ]
137    saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod1(
138        saxophone_random_walk_three)]
138
139 seed(8)
140 cello_random_walk_three = []
141 cello_random_walk_three.append(-1 if random() < 0.5 else 1)
142 for i in range(1, 1000):
143     movement = -1 if random() < 0.5 else 1
144     value = cello_random_walk_three[i-1] + movement
145     cello_random_walk_three.append(value)
146    cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
147    cello_chord_three = [-18, -3, ]
148    cello_notes_three = [cello_chord_three[x] for x in reduceMod1(

```

```

    cello_random_walk_three)]

149 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
150     talea=abjadext.rmakers.Talea(
151         counts=[3, 6, 1, 4, 5, 1, 7, 1, 2, 1, ],
152         denominator=32,
153         ),
154         ),
155 beamSpecifier=abjadext.rmakers.BeamSpecifier(
156     beamDivisionsTogether=True,
157     beamRests=False,
158     ),
159 extraCountsPerDivision=[0, 1, 0, -1],
160 burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
161     leftClasses=[abjad.Note, abjad.Rest],
162     leftCounts=[1, 0, 1],
163     ),
164 tupletSpecifier=abjadext.rmakers.TupletSpecifier(
165     trivialize=True,
166     extractTrivial=True,
167     rewriteRestFilled=True,
168     ),
169 )
170
171 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
172     talea=abjadext.rmakers.Talea(
173         counts=[1, 1, 1, 3, 2, 1, 2, 1, 3, ],
174         denominator=16,
175         ),
176 beamSpecifier=abjadext.rmakers.BeamSpecifier(
177     beamDivisionsTogether=True,
178     beamRests=False,
179     ),
180 extraCountsPerDivision=[1, 0, -1, 0],
181 burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
182     leftClasses=[abjad.Note, abjad.Rest],
183     leftCounts=[1, 0, 1],
184     ),
185 tupletSpecifier=abjadext.rmakers.TupletSpecifier(
186     trivialize=True,
187     extractTrivial=True,
188     rewriteRestFilled=True,
189     ),
190 )
191
192 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
193     denominators=[16, 16, 8, 16, 16, 8],
194     extraCountsPerDivision=[1, 0, 0, -1, 0, 1, -1, 0],
195     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
196         leftClasses=[abjad.Rest],
197         leftCounts=[1],
198         rightClasses=[abjad.Rest],
199         rightCounts=[1],
200         outerDivisionsOnly=True,
201         ),

```

```

202     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
203         trivialize=True,
204         extract_trivial=True,
205         rewrite_rest_filled=True,
206         ),
207     )
208
209 attachmentHandlerOne = AttachmentHandler(
210     starting_dynamic='p',
211     ending_dynamic='mp',
212     hairpin_indicator='--',
213     articulation='accent',
214 )
215
216 attachmentHandlerTwo = AttachmentHandler(
217     starting_dynamic='fff',
218     ending_dynamic='mf',
219     hairpin_indicator='>',
220     articulation='tenuto',
221 )
222
223 attachmentHandlerThree = AttachmentHandler(
224     starting_dynamic='mp',
225     ending_dynamic='ff',
226     hairpin_indicator='<|',
227     articulation='',
228 )
229
230 ##### oboe #####
231 fluteMusicMakerOne = MusicMaker(
232     rmaker=rmakerOne,
233     pitches=fluteNotesOne,
234     continuous=True,
235     attachmentHandler=attachmentHandlerOne,
236 )
237 fluteMusicMakerTwo = MusicMaker(
238     rmaker=rmakerTwo,
239     pitches=fluteNotesTwo,
240     continuous=True,
241     attachmentHandler=attachmentHandlerTwo,
242 )
243 fluteMusicMakerThree = MusicMaker(
244     rmaker=rmakerThree,
245     pitches=fluteNotesThree,
246     continuous=True,
247     attachmentHandler=attachmentHandlerThree,
248 )
249 ##### saxophone #####
250 saxophoneMusicMakerOne = MusicMaker(
251     rmaker=rmakerOne,
252     pitches=saxophoneNotesOne,
253     continuous=True,
254     attachmentHandler=attachmentHandlerOne,
255 )

```

```

256 saxophonemusicmaker_two = MusicMaker(
257     rmaker=rmaker_two,
258     pitches=saxophone_notes_two,
259     continuous=True,
260     attachment_handler=attachment_handler_two,
261 )
262 saxophonemusicmaker_three = MusicMaker(
263     rmaker=rmaker_three,
264     pitches=saxophone_notes_three,
265     continuous=True,
266     attachment_handler=attachment_handler_three,
267 )
268 #####cello#####
269 cellomusicmaker_one = MusicMaker(
270     rmaker=rmaker_one,
271     pitches=cello_notes_one,
272     continuous=True,
273     attachment_handler=attachment_handler_one,
274 )
275 cellomusicmaker_two = MusicMaker(
276     rmaker=rmaker_two,
277     pitches=cello_notes_two,
278     continuous=True,
279     attachment_handler=attachment_handler_two,
280 )
281 cellomusicmaker_three = MusicMaker(
282     rmaker=rmaker_three,
283     pitches=cello_notes_three,
284     continuous=True,
285     attachment_handler=attachment_handler_three,
286 )
287
288 silence_maker = abjadext.rmakers.NoteRhythmMaker(
289     division_masks=[
290         abjadext.rmakers.SilenceMask(
291             pattern=abjad.index([0], 1),
292         ),
293     ],
294 )
295
296 class MusicSpecifier:
297
298     def __init__(self, music_maker, voice_name):
299         self.music_maker = music_maker
300         self.voice_name = voice_name
301
302     print('Collecting timespans and rmakers ...')
303 #####group one#####
304     voice_1_timespan_list = abjad.TimespanList([
305         abjad.AnnotatedTimespan(
306             start_offset=start_offset,
307             stop_offset=stop_offset,
308             annotation=MusicSpecifier(
309                 music_maker=music_maker,

```

```

310         voice_name='Voice 1',
311     ),
312 )
313 for start_offset, stop_offset, music_maker in [
314     [(0, 8), (3, 8), flutemusicmaker_one],
315     [(4, 8), (8, 8), flutemusicmaker_two],
316     [(10, 8), (12, 8), flutemusicmaker_three],
317     [(12, 8), (15, 8), flutemusicmaker_one],
318     [(18, 8), (24, 8), flutemusicmaker_two],
319     [(28, 8), (33, 8), flutemusicmaker_three],
320     [(33, 8), (35, 8), flutemusicmaker_one],
321     [(40, 8), (42, 8), flutemusicmaker_two],
322     [(42, 8), (44, 8), flutemusicmaker_three],
323     [(44, 8), (48, 8), flutemusicmaker_one],
324     [(54, 8), (55, 8), flutemusicmaker_two],
325     [(62, 8), (64, 8), flutemusicmaker_three],
326     [(72, 8), (75, 8), flutemusicmaker_one],
327     [(76, 8), (79, 8), flutemusicmaker_two],
328     [(79, 8), (80, 8), silence_maker],
329 ]
330 ])
331
332 voice_3_timespan_list = abjad.TimespanList([
333     abjad.AnnotatedTimespan(
334         start_offset=start_offset,
335         stop_offset=stop_offset,
336         annotation=MusicSpecifier(
337             music_maker=music_maker,
338             voice_name='Voice 3',
339         ),
340     )
341     for start_offset, stop_offset, music_maker in [
342         [(9, 8), (12, 8), saxophonemusicmaker_one],
343         [(20, 8), (24, 8), saxophonemusicmaker_two],
344         [(31, 8), (33, 8), saxophonemusicmaker_three],
345         [(33, 8), (36, 8), saxophonemusicmaker_one],
346         [(42, 8), (48, 8), saxophonemusicmaker_two],
347         [(53, 8), (56, 8), saxophonemusicmaker_three],
348         [(56, 8), (60, 8), saxophonemusicmaker_one],
349         [(64, 8), (69, 8), saxophonemusicmaker_two],
350         [(69, 8), (72, 8), saxophonemusicmaker_three],
351         [(75, 8), (79, 8), saxophonemusicmaker_one],
352     ]
353 ])
354
355 voice_8_timespan_list = abjad.TimespanList([
356     abjad.AnnotatedTimespan(
357         start_offset=start_offset,
358         stop_offset=stop_offset,
359         annotation=MusicSpecifier(
360             music_maker=music_maker,
361             voice_name='Voice 8',
362         ),
363     )

```

```

364     for start_offset, stop_offset, music_maker in [
365         [(15, 8), (18, 8), celломузыкар_один],
366         [(18, 8), (22, 8), celломузыкар_два],
367         [(25, 8), (29, 8), celломузыкар_три],
368         [(29, 8), (32, 8), celломузыкар_один],
369         [(35, 8), (39, 8), celломузыкар_два],
370         [(39, 8), (42, 8), celломузыкар_три],
371         [(45, 8), (50, 8), celломузыкар_один],
372         [(50, 8), (52, 8), celломузыкар_два],
373         [(55, 8), (56, 8), celломузыкар_три],
374         [(56, 8), (61, 8), celломузыкар_один],
375         [(61, 8), (62, 8), celломузыкар_два],
376         [(65, 8), (69, 8), celломузыкар_три],
377         [(69, 8), (72, 8), celломузыкар_один],
378         [(75, 8), (79, 8), celломузыкар_два],
379     ]
380   ])
381
382 all_timespan_lists = {
383     'Voice 1': voice_1_timespan_list,
384     'Voice 3': voice_3_timespan_list,
385     'Voice 8': voice_8_timespan_list,
386 }
387
388 global_timespan = abjad.Timespan(
389     start_offset=0,
390     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values())
391 )
392
393 for voice_name, timespan_list in all_timespan_lists.items():
394     silences = abjad.TimespanList([global_timespan])
395     silences.extend(timespan_list)
396     silences.sort()
397     silences.compute_logical_xor()
398     for silence_timespan in silences:
399         timespan_list.append(
400             abjad.AnnotatedTimespan(
401                 start_offset=silence_timespan.start_offset,
402                 stop_offset=silence_timespan.stop_offset,
403                 annotation=MusicSpecifier(
404                     music_maker=None,
405                     voice_name=voice_name,
406                 ),
407             )
408         )
409     timespan_list.sort()
410
411 for voice_name, timespan_list in all_timespan_lists.items():
412     shards = timespan_list.split_at_offsets(bounds)
413     split_timespan_list = abjad.TimespanList()
414     for shard in shards:
415         split_timespan_list.extend(shard)
416     split_timespan_list.sort()
417     all_timespan_lists[voice_name] = timespan_list

```

```

418 score = abjad.Score([
419     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global Context'),
420     abjad.StaffGroup(
421         [
422             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
423             lilypond_type='Staff',),
424             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
425             lilypond_type='Staff',),
426             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
427             lilypond_type='Staff',),
428         ],
429         name='Staff Group 1',
430     ),
431 ],
432 )
433
434 for time_signature in time_signatures:
435     skip = abjad.Skip(1, multiplier=(time_signature))
436     abjad.attach(time_signature, skip)
437     score['Global Context'].append(skip)
438
439 print('Making containers ...')
440
441 def make_container(music_maker, durations):
442     selections = music_maker(durations)
443     container = abjad.Container([])
444     container.extend(selections)
445     return container
446
447 def key_function(timespan):
448     return timespan.annotation.music_maker or silence_maker
449
450 for voice_name, timespan_list in all_timespan_lists.items():
451     for music_maker, grouper in itertools.groupby(
452         timespan_list,
453         key=key_function,
454     ):
455         durations = [timespan.duration for timespan in grouper]
456         container = make_container(music_maker, durations)
457         voice = score[voice_name]
458         voice.append(container)
459
460 print('Splitting and rewriting ...')
461
462 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.Voice):
463     for i, shard in enumerate(abjad.mutate(voice[:]).split(time_signatures)):
464         time_signature = time_signatures[i]
465         abjad.mutate(shard).rewrite_meter(time_signature)
466
467 print('Beaming runs ...')
468
469 for voice in abjad.select(score).components(abjad.Voice):
470     for run in abjad.select(voice).runs():

```

```

469     if 1 < len(run):
470         specifier = abjadext.rmakers.BeamSpecifier(
471             beam_each_division=False,
472             )
473         specifier(run)
474         abjad.attach(abjad.StartBeam(), run[0])
475         abjad.attach(abjad.StopBeam(), run[-1])
476         for leaf in run:
477             if abjad.Duration(1, 4) <= leaf.written_duration:
478                 continue
479             previous_leaf = abjad.inspect(leaf).leaf(-1)
480             next_leaf = abjad.inspect(leaf).leaf(1)
481             if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
482                 abjad.Duration(1, 4) <= next_leaf.written_duration):
483                 left = previous_leaf.written_duration.flag_count
484                 right = leaf.written_duration.flag_count
485                 beam_count = abjad.BeamCount(
486                     left=left,
487                     right=right,
488                     )
489                 abjad.attach(beam_count, leaf)
490                 continue
491             if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
492                 abjad.Duration(1, 4) <= previous_leaf.written_duration):
493                 left = leaf.written_duration.flag_count
494                 right = next_leaf.written_duration.flag_count
495                 beam_count = abjad.BeamCount(
496                     left=left,
497                     right=right,
498                     )
499                 abjad.attach(beam_count, leaf)
500
501 print('Stopping Hairpins ...')
502 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
503     for rest in abjad.iterate(staff).components(abjad.Rest):
504         previous_leaf = abjad.inspect(rest).leaf(-1)
505         if isinstance(previous_leaf, abjad.Note):
506             abjad.attach(abjad.StopHairpin(), rest)
507         elif isinstance(previous_leaf, abjad.Chord):
508             abjad.attach(abjad.StopHairpin(), rest)
509         elif isinstance(previous_leaf, abjad.Rest):
510             pass
511
512 print('Adding attachments ...')
513 bar_line = abjad.BarLine('||.')
514 metro = abjad.MetronomeMark((1, 4), 60)
515 markup1 = abjad.Markup(r'\bold { A }')
516 markup2 = abjad.Markup(r'\bold { B }')
517 markup3 = abjad.Markup(r'\bold { C }')
518 markup4 = abjad.Markup(r'\bold { D }')
519 markup5 = abjad.Markup(r'\bold { E }')
520 markup6 = abjad.Markup(r'\bold { F }')
521 mark1 = abjad.RehearsalMark(markup=markup1)
522 mark2 = abjad.RehearsalMark(markup=markup2)

```

```

523 mark3 = abjad.RehearsalMark(markup=markup3)
524 mark4 = abjad.RehearsalMark(markup=markup4)
525 mark5 = abjad.RehearsalMark(markup=markup5)
526 mark6 = abjad.RehearsalMark(markup=markup6)
527
528 def _apply_numerators_and_tech(staff, nums, tech):
529     numerators = cyc(nums)
530     techs = cyc(tech)
531     for logical_tie in abjad.select(staff).logical_ties(pitched=True):
532         tech = next(techs)
533         numerator = next(numerators)
534         bcp = abjad.BowContactPoint((numerator, 5))
535         technis = abjad.BowMotionTechnique(tech)
536         for note in logical_tie:
537             abjad.attach(bcp, note)
538             abjad.attach(technis, note)
539     for run in abjad.select(staff).runs():
540         abjad.bow_contact_spanner(run, omit_bow_changes=False)
541
542 instruments1 = cyc([
543     abjad.Flute(),
544     abjad.AltoSaxophone(),
545     abjad.Cello(),
546 ])
547
548 clefs1 = cyc([
549     abjad.Clef('treble'),
550     abjad.Clef('treble'),
551     abjad.Clef('bass'),
552 ])
553
554 abbreviations1 = cyc([
555     abjad.MarginMarkup(markup=abjad.Markup('fl.'),),
556     abjad.MarginMarkup(markup=abjad.Markup('sx.'),),
557     abjad.MarginMarkup(markup=abjad.Markup('vc.'),),
558 ])
559
560 names1 = cyc([
561     abjad.StartMarkup(markup=abjad.Markup('Flute'),),
562     abjad.StartMarkup(markup=abjad.Markup('Saxophone'),),
563     abjad.StartMarkup(markup=abjad.Markup('Violoncello'),),
564 ])
565
566 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
567     leaf1 = abjad.select(staff).leaves()[0]
568     abjad.attach(next(instruments1), leaf1)
569     abjad.attach(next(abbreviations1), leaf1)
570     abjad.attach(next(names1), leaf1)
571     abjad.attach(next(clefs1), leaf1)
572
573 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff)[0]:
574     leaf1 = abjad.select(staff).leaves()[0]
575     last_leaf = abjad.select(staff).leaves()[-1]
576     #abjad.attach(metro, leaf1)

```

```

577 abjad.attach(bar_line, last_leaf)
578
579 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
580     abjad.Instrument.transpose_from_sounding_pitch(staff)
581
582 score_file = abjad.LilyPondFile.new(
583     score,
584     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/source/
585     _stylesheets/abjad.ily'],
586 )
587
588 abjad.SegmentMaker.comment_measure_numbers(score)
589 ######
590 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand/
591     Segments/Segment_IV'
592 pdf_path = f'{directory}/Segment_IV.pdf'
593 path = pathlib.Path('Segment_IV.pdf')
594 if path.exists():
595     print(f'Removing {pdf_path} ...')
596     path.unlink()
597 time_1 = time.time()
598 print(f'Persisting {pdf_path} ...')
599 result = abjad.persist(score_file).as_pdf(pdf_path)
600 print(result[0])
601 print(result[1])
602 print(result[2])
603 success = result[3]
604 if success is False:
605     print('LilyPond failed!')
606 time_2 = time.time()
607 total_time = time_2 - time_1
608 print(f'Total time: {total_time} seconds')
609 if path.exists():
610     print(f'Opening {pdf_path} ...')
611     os.system(f'open {pdf_path}')

```

**Code Example A.19:** Four Ages of Sand Segment\_IV

#### A.4.2 STYLESHEET

```

1 Four Ages of Sand Stylesheet.
2 % 2018-07-17 19:54
3
4 \version "2.19.82"
5 \language "english"
6 #(set-default-paper-size "letter")
7 #(set-global-staff-size 13)

```

```

8 \include "ekmelily"
9 \ekmelicStyle evans
10
11 \header {
12   tagline = ##f
13   breakbefore = ##t
14   title = \markup \override #'(
15     font-name . "Didot"
16   ) \fontsize #15 \bold \center-column {
17     "Four Ages of Sand"
18   }
19   subtitle = \markup \override #'(
20     font-name . "Didot"
21   ) \fontsize #4 \center-column {
22     "for flute, saxophone, and violoncello"
23   }
24   arranger = \markup \override #'(
25     font-name . "Didot"
26   ) \fontsize #2.5 {
27     "Gregory Rowland Evans"
28   }
29 }
30
31 \layout {
32   \accidentalStyle forget
33   \%accidentalStyle modern
34   \%accidentalStyle modern-cautionary
35   \%accidentalStyle neo-modern
36   \%accidentalStyle dodecaphonic
37   indent = #5
38   %ragged-last = ##t
39   %ragged-right = ##t
40   %left-margin = #15

```

```

41 \context {
42     \name TimeSignatureContext
43     \type Engraver_group
44     \numericTimeSignature
45     \consists Axis_group_engraver
46     \consists Bar_number_engraver
47         \consists Time_signature_engraver
48     \consists Mark_engraver
49     \consists Metronome_mark_engraver
50     \override BarNumber.Y-extent = #'(0 . 0)
51     \override BarNumber.Y-offset = 0
52     \override BarNumber.extra-offset = #'(-4 . 0)
53     \%override BarNumber.font-name = "Didot"
54     \override BarNumber.stencil = #(
55         make-stencil-boxer 0.1 0.7 ly:text-interface::print
56     )
57     \override BarNumber.font-size = 1
58     \override BarNumber.padding = 4
59     \override MetronomeMark.X-extent = #'(0 . 0)
60     \override MetronomeMark.Y-extent = #'(0 . 0)
61     \override MetronomeMark.break-align-symbols = #'(left-edge)
62     \override MetronomeMark.extra-offset = #'(0 . 4)
63     \override MetronomeMark.font-size = 3
64     \override RehearsalMark.stencil = #(
65         make-stencil-circler 0.1 0.7 ly:text-interface::print
66     )
67     \override RehearsalMark.X-extent = #'(0 . 0)
68     \override RehearsalMark.X-offset = 6
69     \override RehearsalMark.Y-offset = -2.25
70     \override RehearsalMark.break-align-symbols = #'(time-signature)
71     \override RehearsalMark.break-visibility = #end-of-line-invisible
72     \override RehearsalMark.font-name = "Didot"
73     \override RehearsalMark.font-size = 8

```

```

74   \override RehearsalMark.outside-staff-priority = 500
75   \override RehearsalMark.self-alignment-X = #center
76     \override TimeSignature.X-extent = #'(0 . 0)
77     \override TimeSignature.X-offset = #ly:self-alignment-interface::x-
aligned-on-self
78     \override TimeSignature.Y-extent = #'(0 . 0)
79   \override TimeSignature.Y-offset = -5
80     \override TimeSignature.break-align-symbol = ##f
81     \override TimeSignature.break-visibility = #end-of-line-invisible
82     \override TimeSignature.font-size = #4
83   \override TimeSignature.self-alignment-X = #center
84   \override VerticalAxisGroup.default-staff-staff-spacing = #'(
85     (basic-distance . 0) (minimum-distance . 10) (padding . 6) (stretchability
86     . 0)
87   )
88 }
89
90 \context {
91   \Score
92     \remove Bar_number_engraver
93     \remove Mark_engraver
94     \accepts TimeSignatureContext
95   \override BarLine.bar-extent = #'(-2 . 2)
96     \override Beam.breakable = ##t
97   \override Beam.concaveness = #10000
98     \override Beam.beam-thickness = #0.8
99     \override Beam.length-fraction = #1.5
100   \override DynamicText.font-size = #-2
101   \override Glissando.breakable = ##t
102     \override SpacingSpanner.strict-grace-spacing = ##t
103     \override SpacingSpanner.strict-note-spacing = ##t
104     \override SpacingSpanner.uniform-stretching = ##t
105     \override StaffGrouper.staff-staff-spacing = #'(
106       (basic-distance . 14) (minimum-distance . 14) (padding . 1)

```

```

105     s)
106
107     \override Stem.thickness = #0.75
108
109     \override TupletBracket.bracket-visibility = ##t
110     \override TupletBracket.minimum-length = #3
111     \override TupletBracket.padding = #2
112     \override TupletBracket.springs-and-rods = #ly:spanner::set-spacing-
113     rods
114
115     \override TupletNumber.text = #tuplet-number::calc-fraction-text
116     proportionalNotationDuration = #(ly:make-moment 1 40)
117
118     autoBeaming = ##f
119
120     tupletFullLength = ##t
121
122     }
123
124     \context {
125
126         \Voice
127
128         \remove Forbid_line_break_engraver
129
130     }
131
132     \context {
133
134         \Staff
135
136         \remove Time_signature_engraver
137
138         \hide BarLine
139
140     }
141
142     \context {
143
144         \RhythmicStaff
145
146         \remove Time_signature_engraver
147
148     }
149
150     \context {
151
152         \StaffGroup
153
154     }
155
156     }
157
158
159     \paper {
160
161
162     top-margin = 1.5\cm

```

```

137 bottom-margin = 1.5\cm
138
139 %top-margin = .90\in
140 oddHeaderMarkup = \markup ""
141 evenHeaderMarkup = \markup ""
142 oddFooterMarkup = \markup \fill-line {
143   " "
144   \concat {
145     "Four Ages of Sand ~"
146     \fontsize #2
147     \fromproperty #'page:page-number-string "~ Evans"
148   }
149   " "
150 }
151 evenFooterMarkup = \markup \fill-line {
152   " "
153   \concat { "Four Ages of Sand ~" \fontsize #2
154     \fromproperty #'page:page-number-string "~ Evans"
155   } ""
156 }
157 }
```

**Code Example A.20:** Four Ages of Sand Stylesheet

*Resistance is not, then, in some limited sense a glorious banner of the past, but rather an unremitting struggle and a new consciousness in continuous development through subjective action, its aim being the objective process that leads to those ideals for which so many fell and continue even now to be murdered.*

*The musician too takes part in this fight.*

(2018, *Music and Revolution* pp.273–274)

Luigi Nono

# B

## Appendix of Scores

THE SCORES IN THIS APPENDIX were each composed in 2018. Work on the compositions was begun and completed in a fairly rapid succession. As can be seen in the source code of appendix A, these pieces feature many organizational similarities. The compositions should not be considered as a cycle or series, but individual works that happen to share certain consistent principles.

### B.1 SCORES

#### B.1.1 CTHAR (FOR TWO CELLOS) SCORE

# Cthar

for two cellos

2018

Gregory Rowland Evans

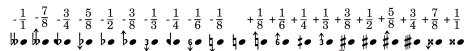
## FOREWORD

Cthar is an Aramaic word, pronounced "seth-ar" meaning "to hide" or "to disassemble."

(G.R.E.)

## PERFORMANCE NOTES

### Microtones:



Accidentals apply only to the pitch which they immediately precede.

**Bow Position Staff**: The upper staff for each instrument notates the horizontal contact point at which the bow touches the string. These positions are written as fractions where  $\frac{0}{1}$  represents *au talon* and  $\frac{1}{1}$  represents *punta d'arco*.

**String Positions**: "ord." stands for ordinario, "st." stands for sul tasto, "sp." stands for sul ponticello, and "msp." stands for molto sul ponticello. An attempt should be made to smoothly transition from one to the next as indicated by the dashed line connecting the two positions.

**Dynamics**: The dynamics indicated should be considered "effort dynamics." As such, the combination of bow speed and effort will often make the cello produce both "flautando" and "scratch" tones. These are the effects desired.

c.6'20"

# Cthar

for two cellos

Gregory Rowland Evans

**4** ♩ = 60

Bow Hand

Violoncello I

Left Hand

Bow Hand

Violoncello II

Left Hand

**5**

st. .... 17.16 ..... ord.

17.16

mf p mp

sp. .... 18.15 ..... 3/4 msp.

7.8

p > > > mp f mf p mp > > > p > > mp

**3** 4

B.H.

vc.I

L.H.

B.H.

vc.II

L.H.

13.12 sp. .... 11.12

mf

13.12 st.

25.24

mp ff p mp

Cthar ~1~ Evans

**5** **4**

B.H.  
vc.I  
vc.II  
L.H.

17.16  
11.12  
imp.

ord.  
sp.

11.12  
mf

**7** **4**

B.H.  
vc.I  
vc.II  
L.H.

18  
11.12  
st.

11.12  
mf

msp.  
7.8  
ord.

15.16  
mp

Cchar -2- Evans

9  $\frac{5}{4}$  A

B.H.

vc.I

L.H. *mp*

B.H.

vc.II

L.H. *fff* *mf* *p* *mp*

10  $\frac{4}{4}$

B.H.

vc.I

L.H. *msp* *sd* *17.16*

B.H.

vc.II

L.H. *fff* *mf*

Cchar ~3~ Evans

Musical score for orchestra and piano, page 15, measures 15-16. The score includes parts for B.H., vc.I, L.H., B.H., vc.II, and L.H. The piano part features a bass line with specific fingerings (e.g., 4, 3, 5, 4, 5) and dynamic markings like *ord.*, *sp.*, *msp.*, *fff*, and *p*. The strings play sustained notes with grace marks. Measure 16 begins with a forte dynamic (*ff*) and ends with a piano dynamic (*p*).

Cthar -4- Evans

17 (B)

B.H.

vc.II

L.H.

B.H.

ve.II

L.H.

ord. 17.15

9.8

*mp*

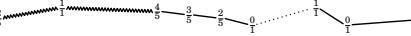
*sp.*

*ff*

*ff*

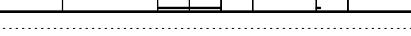
19

**4**

B.H. 

ve.II

**5**

L.H. 

7.6

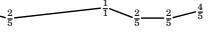
B.H. 

13.12

ve.II

L.H. 

mpf - mp

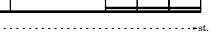
B.H. 

fff - fff

mf

ve.II

15.16

L.H. 

mp

9.8

mf

map... 15.16

fff

15.16

mf - p

23.24

mp

fff

Cthar -5- Evans

Musical score for orchestra and piano showing measures 21-22. The score includes parts for B.H., ve.I, L.H., B.H., ve.II, L.H., and piano. The piano part features a complex rhythmic pattern with grace notes and sustained notes. Measure 21 ends with a forte dynamic (ff) and measure 22 begins with a piano dynamic (p). Various dynamics and performance instructions like "ord.", "st.", and "mp" are included.

[25]  $\frac{5}{4}$  (C)

B.H.  
vc.I  
L.H.  
B.H.  
vc.II  
L.H.

ord.  
sp.  
ff  
mf  
mp  
ff  
mf  
ff  
mf  
p  
mp

[27]  $\frac{3}{4}$

B.H.  
vc.I  
L.H.  
B.H.  
vc.II  
L.H.

msp.  
ord.  
mp  
ff  
ff  
ff  
ff  
ff  
ff  
ff  
ff  
ff  
ff

Cchar ~7~ Evans

[29] **4**

B.H.  
vc.I  
vc.II  
L.H.

ff  
ord. .... 3.4 -> st.  
ord. .... 3.4 -> 21.00 -> sp.  
msp. .... ord.  
ff  
ff  
ff  
ff

**5**

[31] **3**

B.H.  
vc.I  
vc.II  
L.H.

ff  
ord. .... 17.00 -> ff  
ff  
ff  
ff

**4**

ff  
ff  
ff  
ff

**5**

ff  
ff  
ff  
ff

Cchar ~8~ Evans

Musical score for orchestra and piano, page 33, measures 5 and 6. The score includes parts for B.H., vc.I, L.H., B.H., vc.II, and L.H. Measure 5 starts with a forte dynamic. Measure 6 begins with a piano dynamic. Various rhythmic patterns involving eighth and sixteenth notes are shown across the staves.

[37]

[38]

41

**B.H.**

**ve.I** ..... ord.

**L.H.**

**B.H.**

**ve.II** ..... ord.

**L.H.**

42

**B.H.**

**ve.I** ..... sp.

**L.H.**

**B.H.**

**ve.II** ..... sp.

**L.H.**

Musical score for orchestra and piano, page 43, measures 43-44. The score includes parts for Bassoon (B.H.), Trombone (vc.I), Trombone (L.H.), Bassoon (B.H.), Trombone (vc.II), Trombone (L.H.), and Piano (ord.). Measure 43 starts with a dynamic *mp*. Measure 44 begins with a dynamic *sf*.

**[45] 4**

B.H. V  
vc.I 2d st.

L.H. p 9.8

B.H. V  
vc.II ord. 15.16  
L.H. p mp fff mf

**[46] 5**

B.H. V  
vc.I 3 5 1 5  
L.H. 17.16  
B.H. V  
vc.II 17.16  
L.H. sp. 17.16

**[47] 5**

B.H. 3 5 1 5  
vc.I 5d sp.  
L.H. 32  
B.H. V  
vc.II msp. 17.16 ord.  
L.H. 17.16 mf



B.1.2 TIANSHU (FOR 12 PLAYERS) SCORE

天書

Tianshu

for 12 players

2018

Gregory Rowland Evans

## FOREWORD

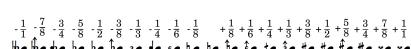
Tiānshū is the name of an art installation in the form of a book by artist Xu Bing filled with meaningless glyphs in the style of traditional Chinese characters. The term tiān shū, which can be translated to mean "divine writing," originally referred to religious texts but is now used to mean "gibberish." A possible alternative title could be "Nonsense Writing."

(G.R.E.)

## PERFORMANCE NOTES

Score is transposed.

**Microtones:**



Accidentals apply only to the pitch which they immediately precede.

## INSTRUMENTATION

Flute

Clarinet in Bb

Bassoon

Horn in F

Trumpet in Bb

Trombone

Tuba

2 Violins

Viola

Violoncello

Contrabass

c.8'

*Tianshu* is dedicated in admiration and friendship to Trevor Bača, Josiah Wolf Oberholtzer, and Jeffrey Treviño from whom I have learned so much.

for Ensemble Ibis



Tianshu  
for twelve players

Gregory Rowland Evans

4 3 4 4

fl. fff mf

cl. mp ff p

bassn. fff mp ff 11:12

4 3 4 4

hr. mp ff p

trp. fff mp ff

trmb. fff mp ff

th. mp ff p

4 3 4 4

vln.I fff mp

vln.II mp ff p

vla. mp ff ff p

vc. mp ff p

cb. mp ff p

<img alt="Musical score for orchestra and choir. The score is divided into three systems by vertical bar lines. Each system consists of two measures. The instruments listed on the left are flute (fl.), clarinet (cl.), bassoon (bassn.), horn (hr.), trumpet (trp.), trombone (trmb.), tuba (th.), violin I (vln.I), violin II (vln.II), viola (vla.), cello (vc.), and double bass (cb.). Measure 1 (3/4 time) includes fl., cl., bassn., hr., trp., trmb., th., vln.I, vln.II, vla., vc., cb. Measure 2 (4/4 time) includes fl. (fff, mf), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. (mp, ff). Measure 3 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. (mp, ff). Measure 4 (3/4 time) includes fl., cl., bassn., hr., trp., trmb., th., vln.I, vln.II, vla., vc., cb. Measure 5 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 6 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 7 (3/4 time) includes fl., cl., bassn., hr., trp., trmb., th., vln.I, vln.II, vla., vc., cb. Measure 8 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 9 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 10 (3/4 time) includes fl., cl., bassn., hr., trp., trmb., th., vln.I, vln.II, vla., vc., cb. Measure 11 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 12 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 13 (3/4 time) includes fl., cl., bassn., hr., trp., trmb., th., vln.I, vln.II, vla., vc., cb. Measure 14 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb. Measure 15 (4/4 time) includes fl. (fff, mp), cl. (mp, ff), bassn. (fff, mp, ff), hr. (mp, ff), trp. (fff, mp, ff), trmb. (fff, mp, ff), th. (mp, ff), vln.I (fff, mp), vln.II (mp, ff), vla. (mp, ff, ff), vc. (mp, ff), cb.</p>

7  $\frac{4}{4}$       8  $\frac{4}{4} \textcircled{A}$       9  $\frac{5}{4}$

fl.  $\text{mp}$   $\text{ff}$

cl.  $\text{mp}$

bassn.  $\text{ff}$

7  $\frac{4}{4}$       8  $\frac{4}{4} \textcircled{A}$       9  $\frac{5}{4}$

hr.  $\text{mp}$

trp.  $\text{mp}$   $\text{ff}$

trmb.  $\text{ff}$

th.  $\text{mp}$

7  $\frac{4}{4}$       8  $\frac{4}{4} \textcircled{A}$       9  $\frac{5}{4}$

vln.I  $\text{mp}$   $\text{ff}$

12:12

vin.II  $\text{mp}$

vla.  $\text{mp}$

vc.  $\text{ff}$

ch.  $\text{mp}$

[10]

13

4 4 5

fl. *mp*

cl. *mf*

bassn. *mp*

13

4 4 5

hr. *mp*

trp. *mp*

trmb. *p*

tb.

13

4 4 5

vln. I *mp*

vln. II *mf*

vla. *mf*

vc. *p*

cb.



19

2 3 4 3

fl. fff *mf*

cl. *mp* *mf*

bassn. *p* *mp*

19

2 3 4 3

hr. fff *mf*

trp. fff *mf*

trmb. *p* *mp*

th. fff *mf*

19

2 3 4 3

vin.I fff *mf*

vin.II *mp ff* *mf*

vla. *mp ff* *mf*

vc. *p* *mp*

cb. fff *mf*

23 4 C 3 5

fl. *mp*

cl. *mp*

bassn. *mp*

23 4 C 3 4

hr. *ff* *mf*

trp. *mp*

trmb. *mp*

tb. *ff* *mf*

23 4 C 3 4

vln.I *mp*

vln.II *mp*

vla. *mp*

ve. *mp*

cb. *ff* *mf*

[26] 
 The musical score consists of two systems of four staves each. The instruments are arranged in pairs across the staves: flutes, clarinets, bassoon, and harp in the first system; trumpets, trombones, tuba, and strings in the second system. The piano part is located at the bottom right of each system.
   
**Measure 26:**  
 - Flute 1 and 2 play eighth-note patterns. Clarinet 1 and 2 play eighth-note patterns. Bassoon plays eighth-note patterns. Harp plays eighth-note patterns.  
 - Trumpet 1 and 2 play eighth-note patterns. Trombone 1 and 2 play eighth-note patterns. Tuba plays eighth-note patterns. Strings play eighth-note patterns.  
**Measure 27:**  
 - Flute 1 and 2 play eighth-note patterns. Clarinet 1 and 2 play eighth-note patterns. Bassoon plays eighth-note patterns. Harp plays eighth-note patterns.  
 - Trumpet 1 and 2 play eighth-note patterns. Trombone 1 and 2 play eighth-note patterns. Tuba plays eighth-note patterns. Strings play eighth-note patterns.  
 - The piano part in both systems includes dynamic markings such as *ff*, *p*, *mp*, *mf*, and *fff*. Measure 27 also includes a tempo marking of *11/12*.

29

**3**

**3 (D)**

**4**

fl. *mp*

cl. *mp*

bassn.

29

**3 (D)**

**4**

hr.

trp. *mp*

trmb.

tb.

29

**3 (D)**

**4**

vln.I *mp*

vln.II *mp*

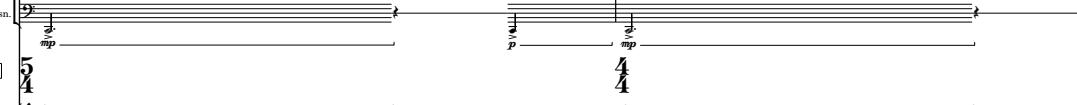
vla. *mp*

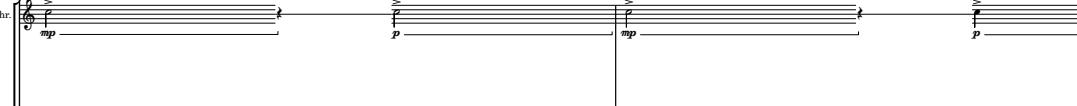
vc.

cb.

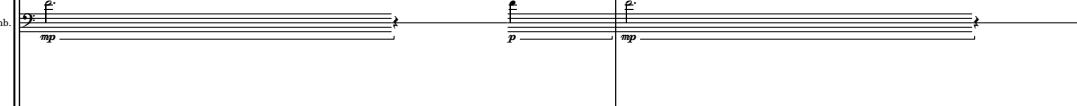
[32] 

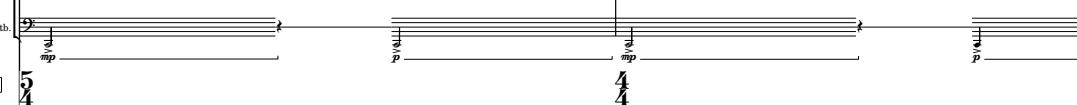




[32] 

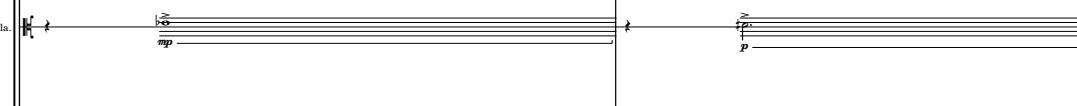


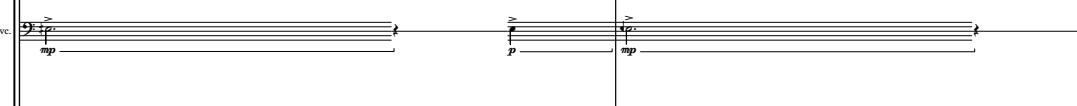


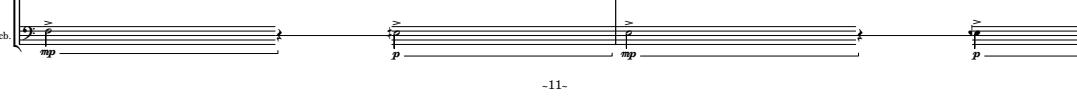


[32] 









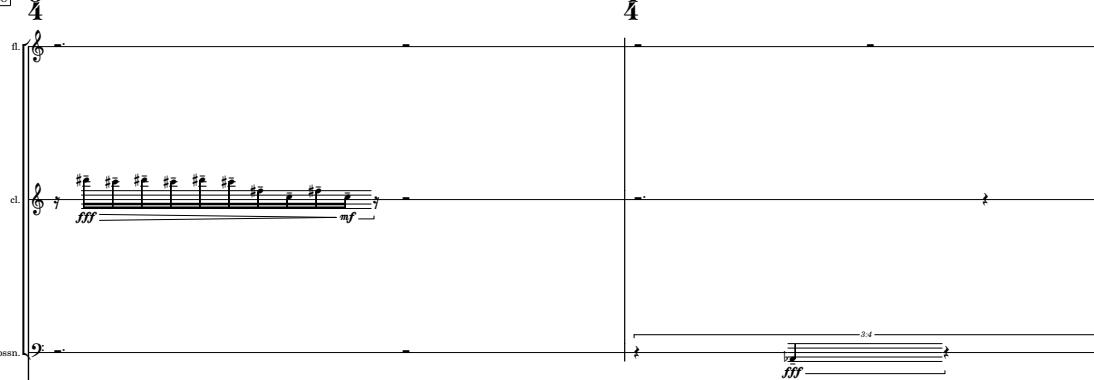


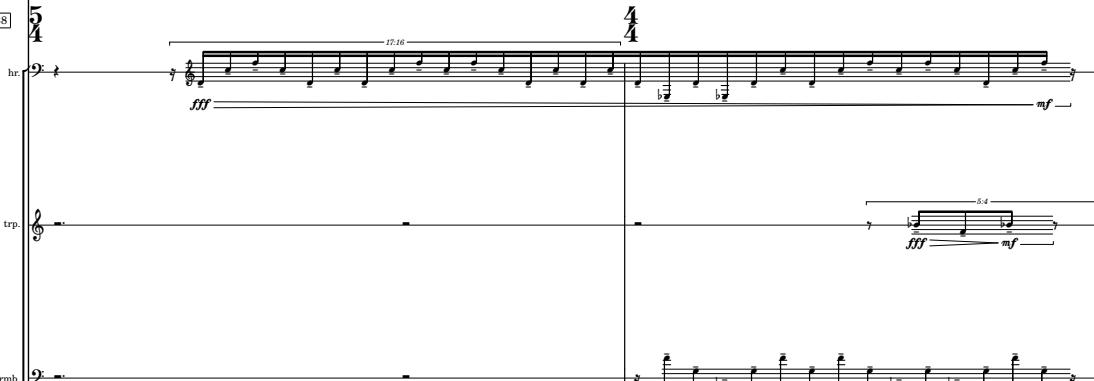
36

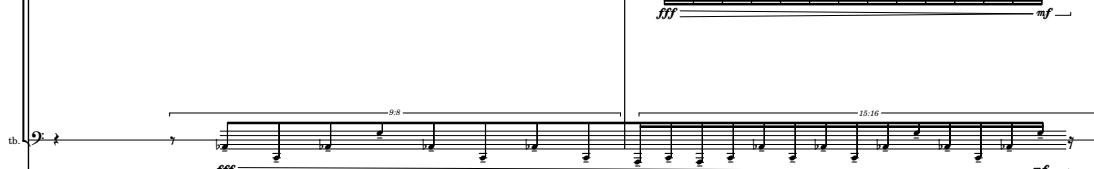
fl.      cl.      bssn.

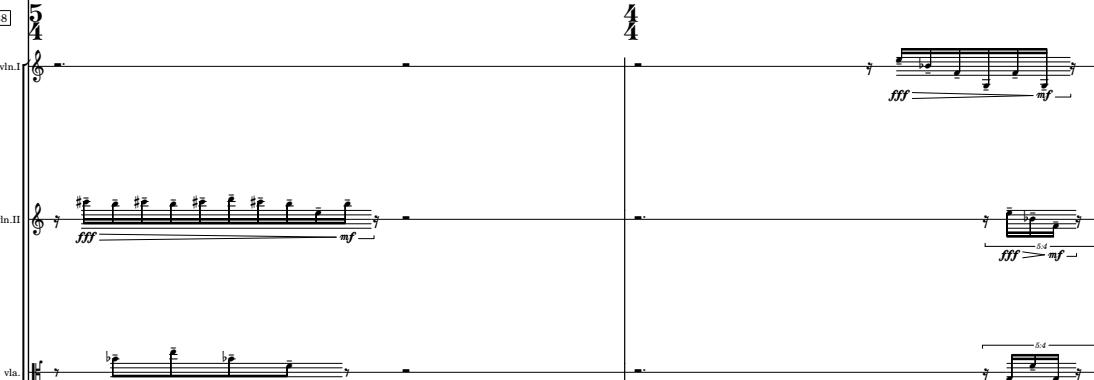
hr.      trp.      trmb.      tb.

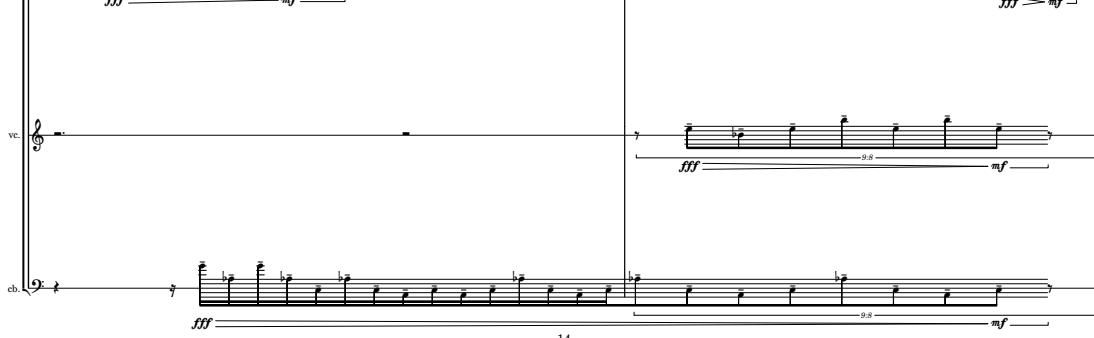
vln.I      vln.II      vla.      ve.      cb.

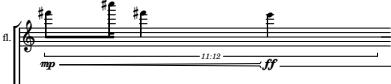
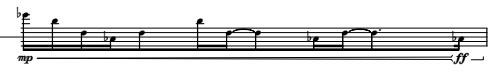
[38] 

[38] 

[38] 

[38] 

[38] 

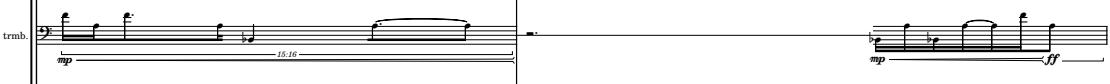
[40]  

cl. 

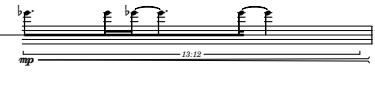
bassn. 

[40]  

trp. 

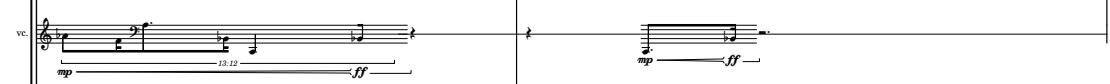
trmb. 

th. 

[40]  

vin.II 

vla. 

vc. 

ch. 

42 5 4

f. - ff - 9.8 - ff -

cl. - ff - 7.8 - ff -

bassn. - ff - 5.4 - ff -

42 5 4

hr. - ff -

trp. - ff - 5.4 - ff -

trmb. - ff - 17.16 - ff -

th. - ff - 7.8 - ff -

42 5 4

vln.I. - ff - 3.4 - ff -

vln.II. - ff - 5.4 - ff -

vla. - ff - 7.8 - ff -

vc. - ff - 11.12 - ff -

ch. - ff - 7.8 - ff -





|        |           |          |          |  |
|--------|-----------|----------|----------|--|
|        | <b>50</b> | <b>2</b> | <b>4</b> | <b>4</b>   |
| fl.    |           |          |          | $\begin{array}{c} \sharp \# \# \\ \# \# \# \end{array}$<br>$mfp$ |
| cl.    |           | $12.12$  |          | $mfp$  |
| bassn. |           |          |          | $7.8$<br>$mfp$   |
| hr.    | $19.20$   | $2$      | $4$      | $mp$   |
| trp.   |           |          |          |  |
| trmb.  |           |          |          | $15.16$<br>$mfp$   |
| tb.    | $6.5$     |          |          | $mp$   |
| vln.I  |           |          |          | $\begin{array}{c} \sharp \# \\ \# \# \# \end{array}$<br>$p$      |
| vln.II |           | $mf$     |          | $mp$   |
| vla.   |           | $mf$     |          | $mp$   |
| ve.    |           |          |          | $17.18$<br>$mf$  |
| ch.    |           | $mf$     |          |  |

[53] 

[53] 

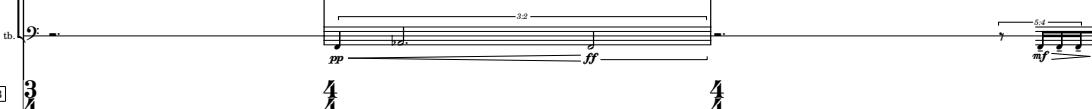
[53] 

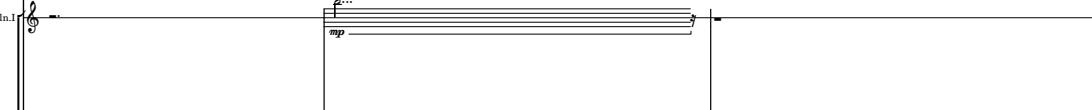
[53] 

[53] 

[53] 

[53] 

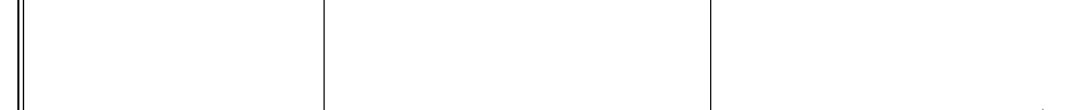
[53] 

[53] 

[53] 

[53] 

[53] 

[53] 

-20-

[56] (G)

fl. *p* *ff*

cl. *mf* *p* *11:12*

bassn. *ff* *3:2*

hr. *mf* *p* *5:4*

trp. *pp* *ff* *3:2* *11:12*

trmb. *ff* *3:2*

th. *p*

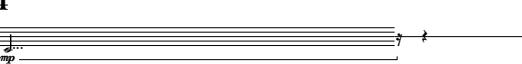
vln.I *pp* *ff*

vin.II *p* *3:2*

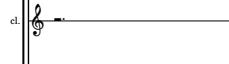
vla. *p* *11:12*

vc. *ff* *3:2*

ch. *p* *3:2*

[58]  

fl.

[58]  

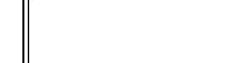
cl.

bassn.

[58]  

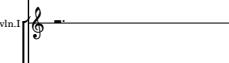
hr.

trp.

[58]  

trmb.

th.

[58]  

vln.I

vln.II

vla.

vc.

ch.

This musical score page contains two systems of five measures each, separated by a vertical bar line. The first system starts at measure 58 with a 5/4 time signature. The second system begins at measure 59 with a 5/4 time signature. The instrumentation includes flute, clarinet, bassoon, horn, trumpet, trombone, tuba, violin I, violin II, viola, cello, and double bass. The notation shows various rhythmic patterns, dynamics (e.g., *p*, *mp*, *mf*, *ff*), and performance instructions like "3.2" and "ff". Measure 58 consists of two measures of flute eighth-note patterns, followed by two measures of bassoon eighth-note patterns, and one measure of tuba eighth-note patterns. Measure 59 consists of two measures of horn eighth-note patterns, followed by two measures of trumpet eighth-note patterns, and one measure of tuba eighth-note patterns. Measures 60 and 61 show various instruments taking turns with eighth-note patterns, with some measures featuring dynamic changes and performance markings like "ff" and "3.2".



[63] 4

f -

cl. -

bassn. 2 3 2.2 ff

[63] 4

hr. 2 3 pp ff

trp. -

trmb. 2.2 pp ff

th. 2 3 pp ff

vln.I 2 3 mp mf

vln.II -

vla. -

ve. 2.2 pp ff

ch. 2 3 pp ff

5

mp mf

17.16 p

mp

mp mf

mp

mp mf

mp

mp mf

mp

mp

mp

65

fl.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

cl.

bassn.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

hr.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

trp.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

trmb.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

tb.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

66

vn.I  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

vln.II  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

vla.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

vc.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

cb.  $\frac{2}{4}$   $\frac{3}{4}$  (II)  $\frac{3}{4}$

68

fl.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

cl.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

bassn.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

hr.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

trp.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

trmb.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

tb.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

vin.I  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

vin.II  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

vla.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

vc.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

ch.  $\frac{2}{4}$   $\frac{3}{4}$   $\frac{4}{4}$   $\frac{3}{4}$

72 4(1) 3 5

fl. 13.12 p

cl. mp

bass. 2.2 pp ff

72 4(1) 3 5

hr. mp mf p

trp. p mp pp 2.2

trmb. 2.2 ff mp

th. mp mf p

72 4(1) 3 5

vln.I 11.12 p

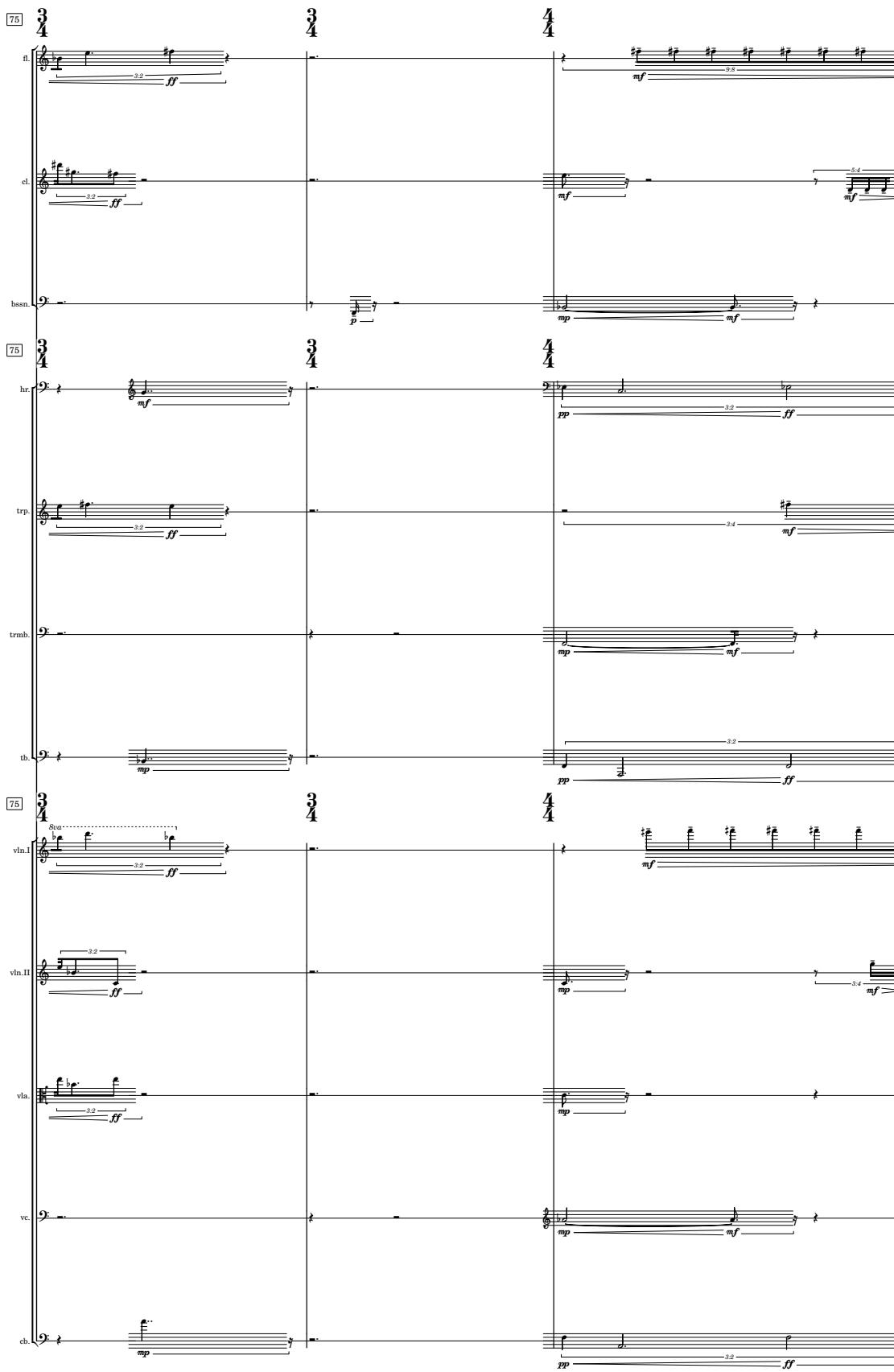
vln.II 7.0 p mp

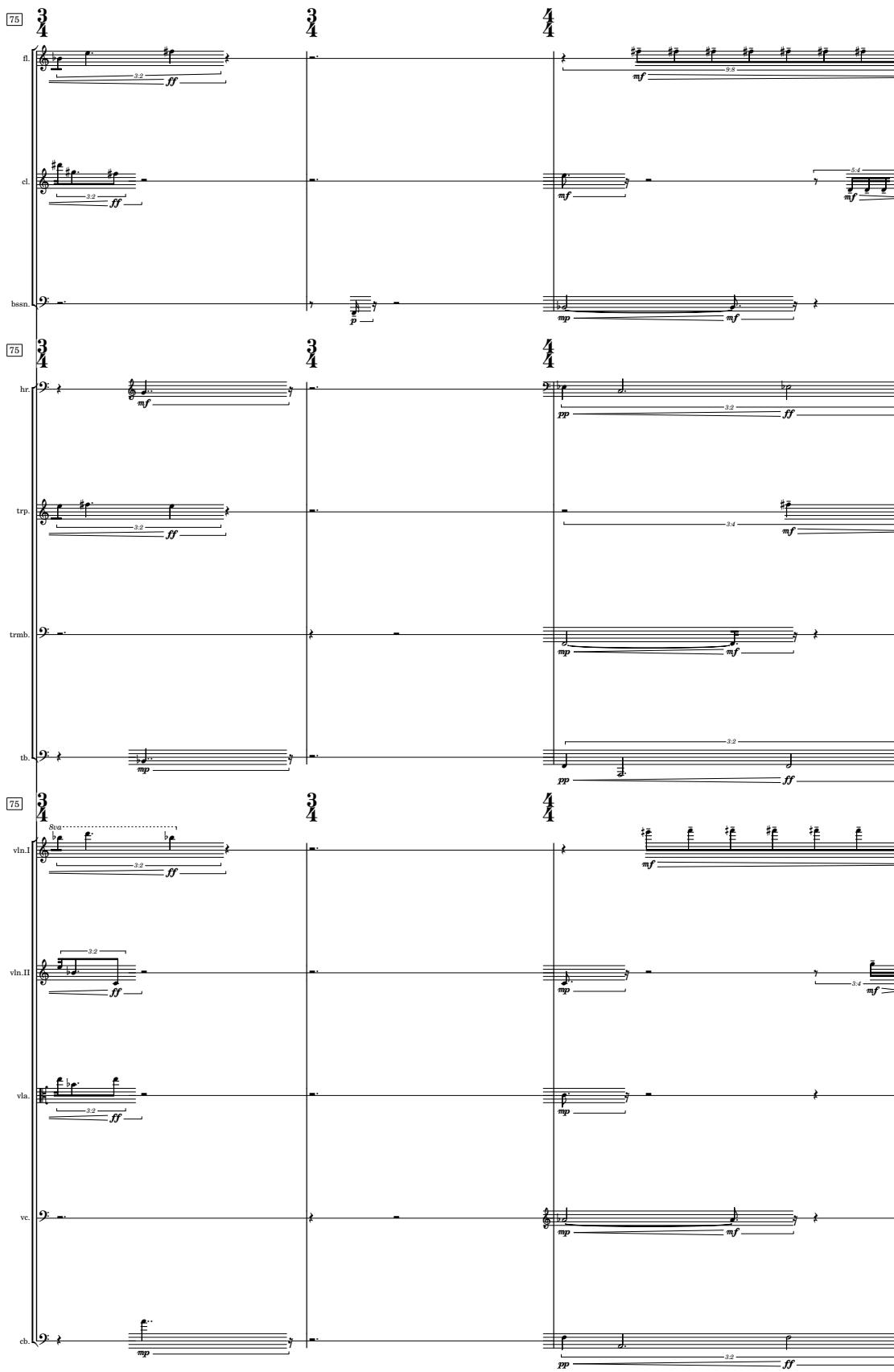
vla. p mp

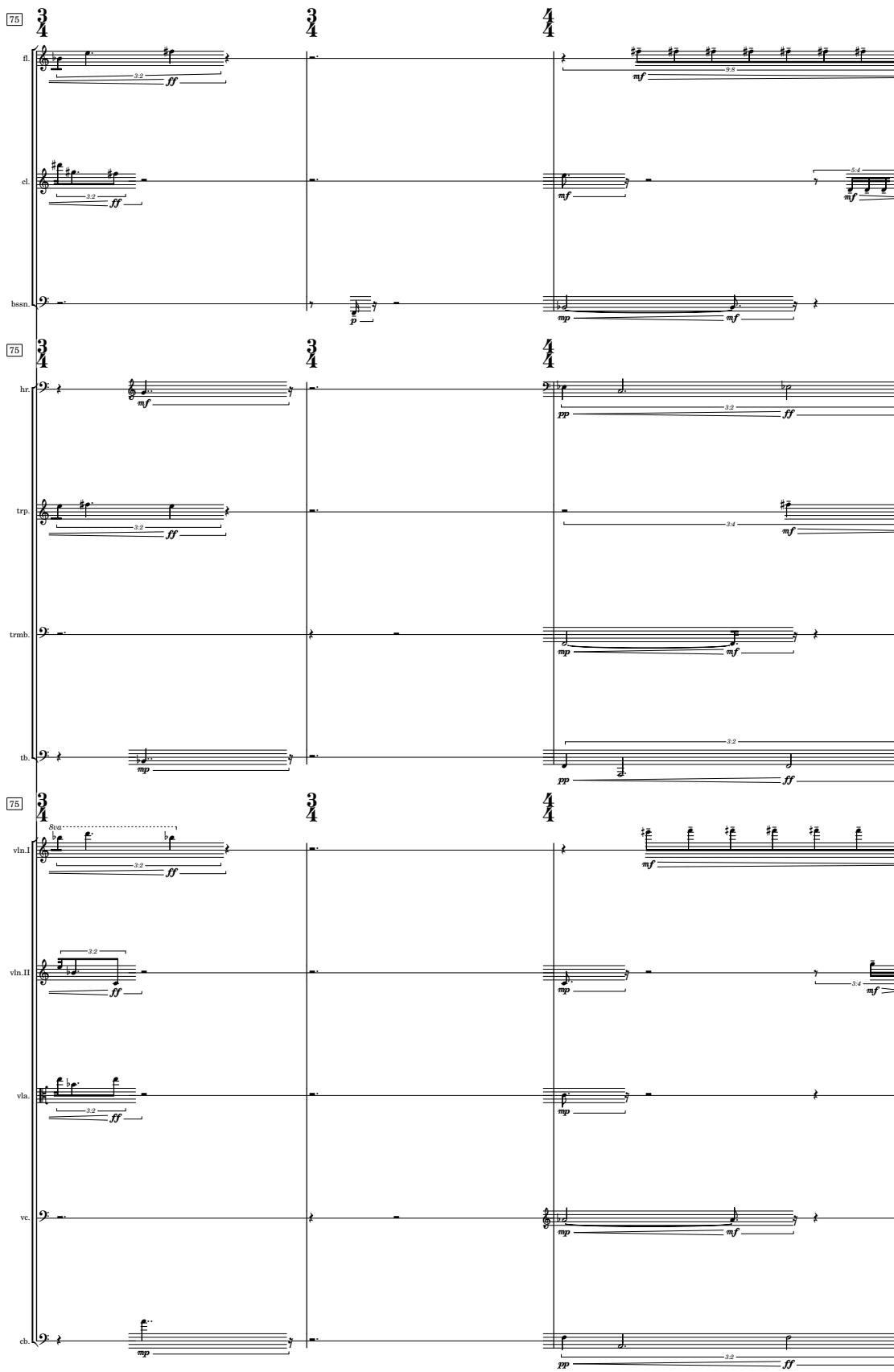
vc. 2.2 ff mp

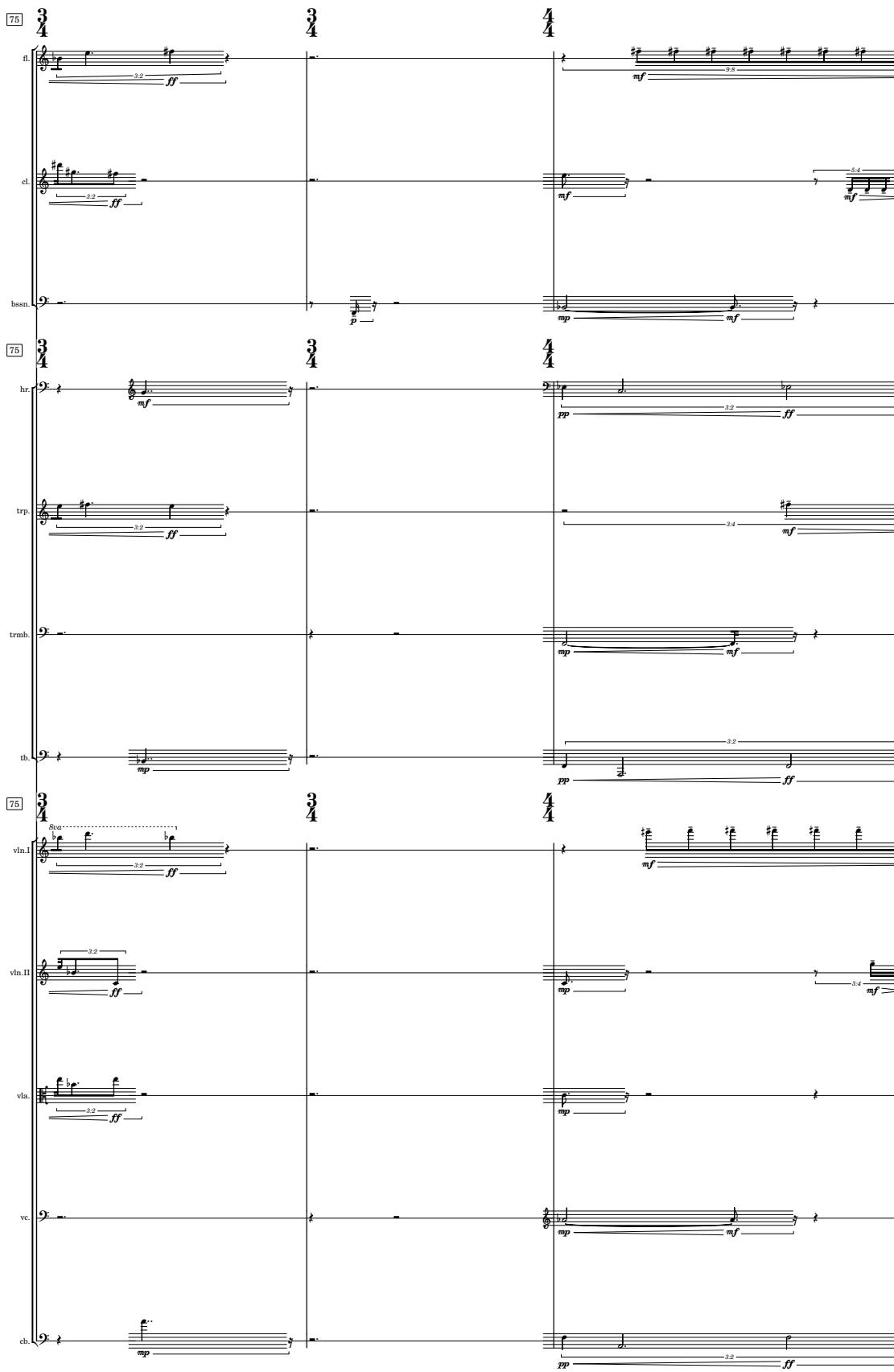
ch. mp mf

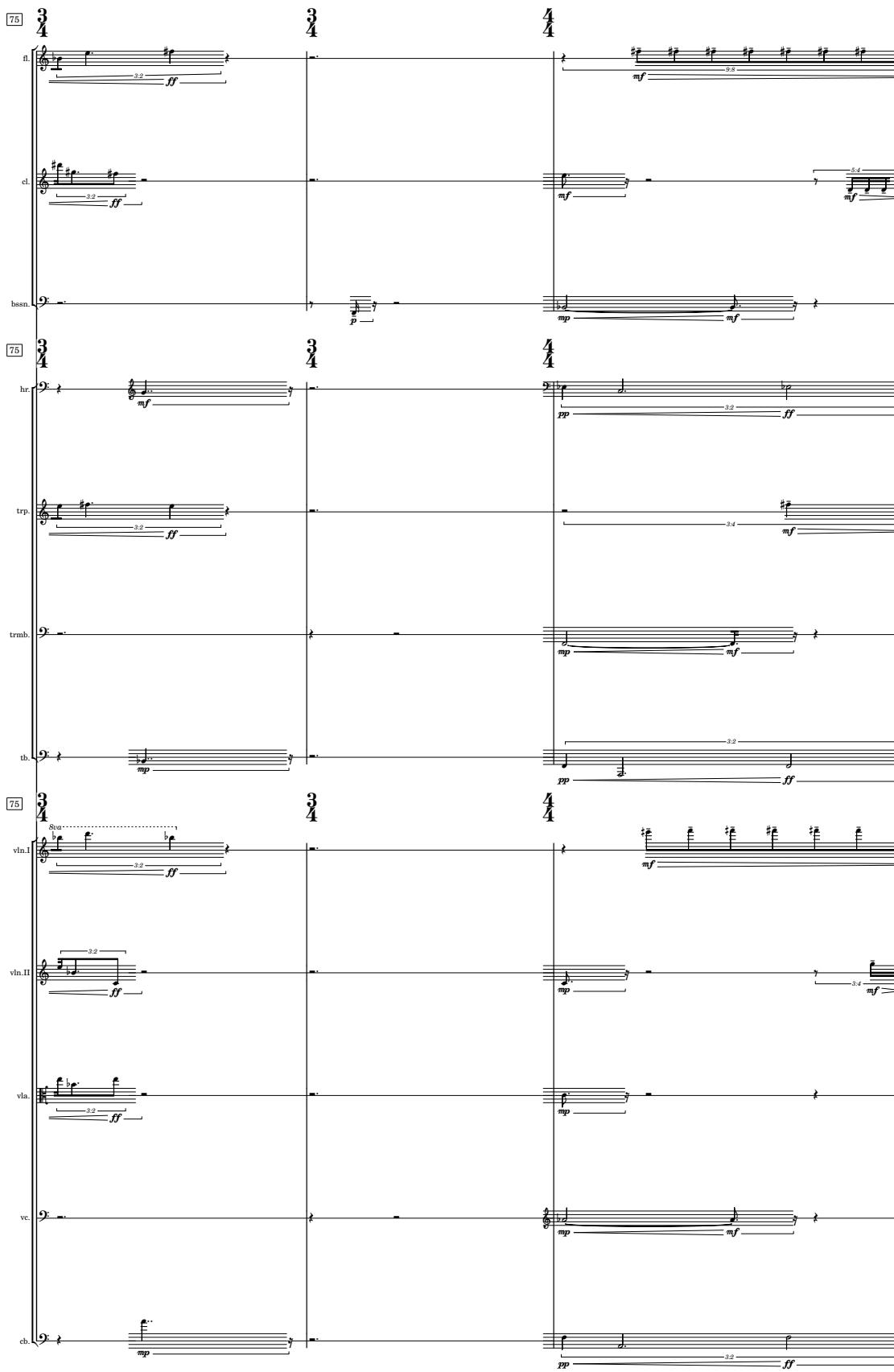
Soprano Alto Tenor Bass

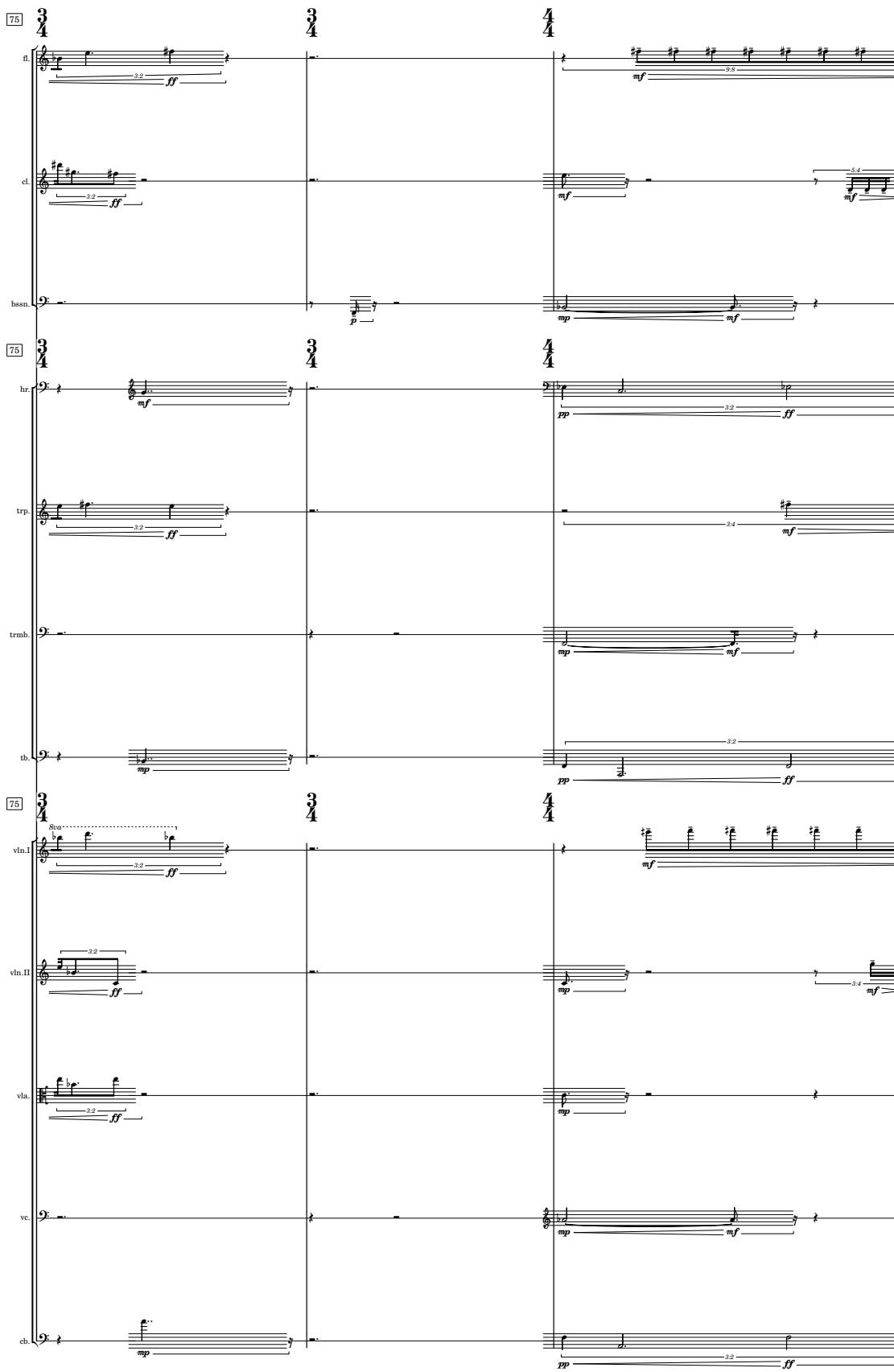
75 

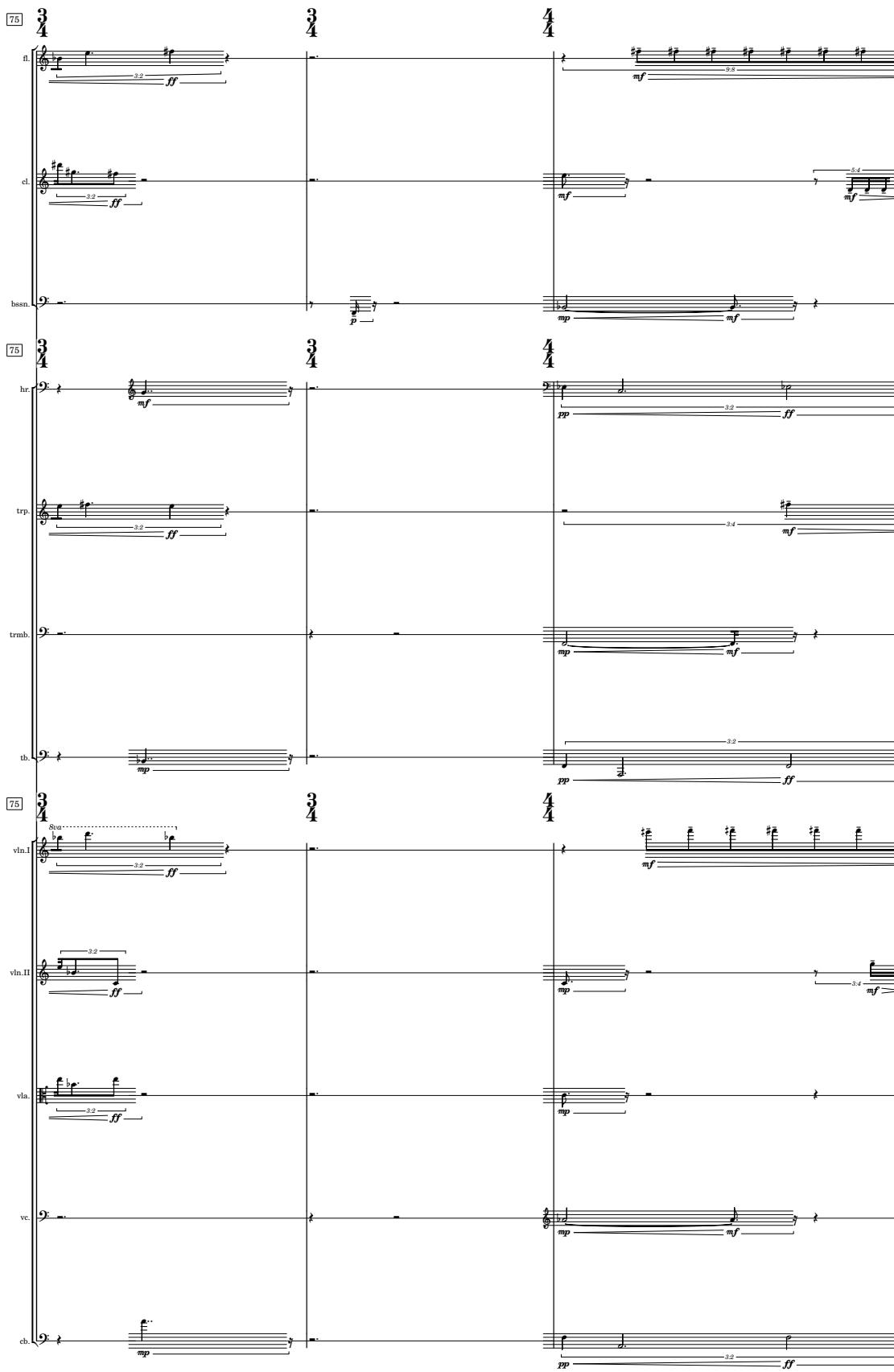
76 

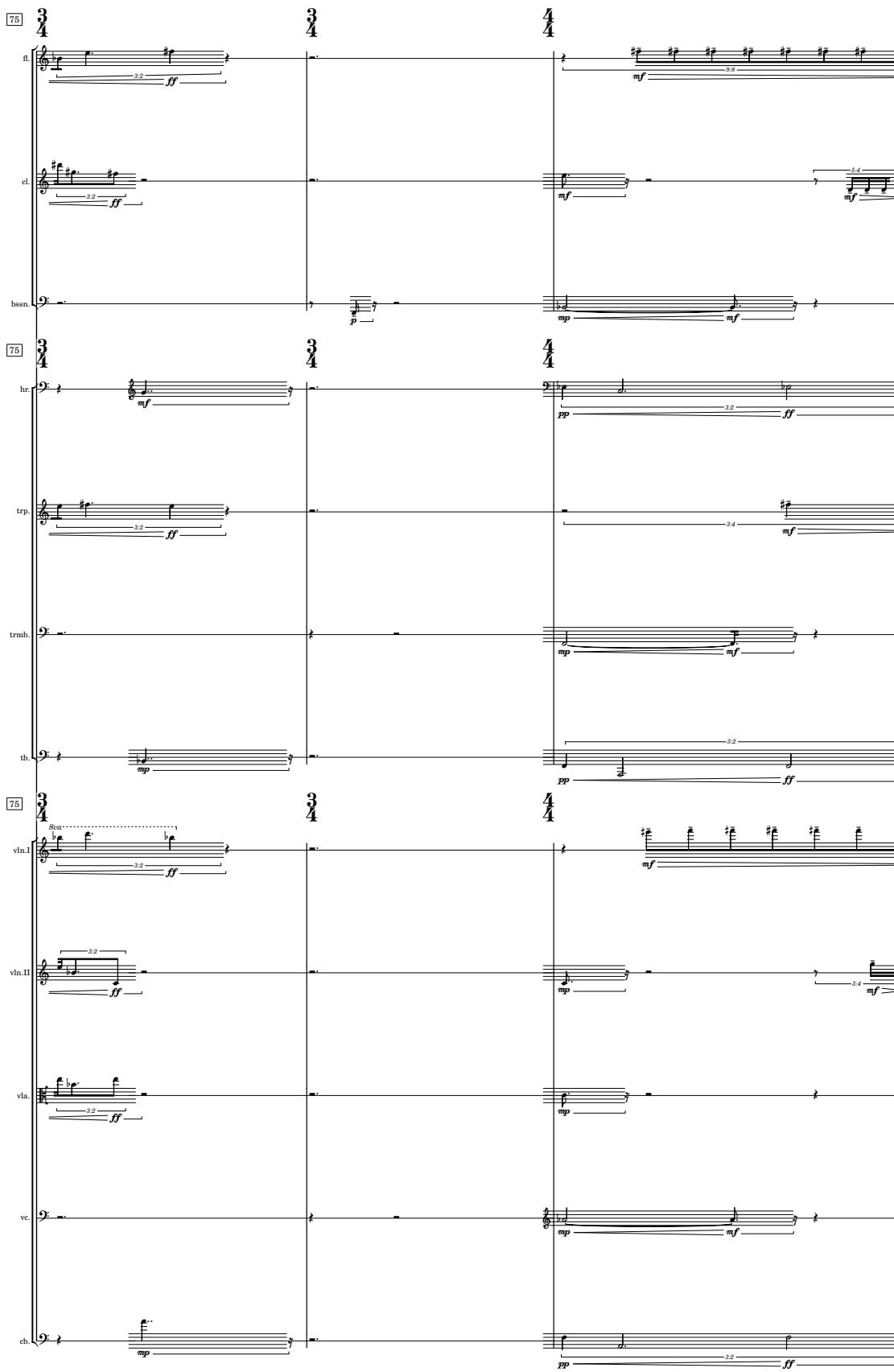
77 

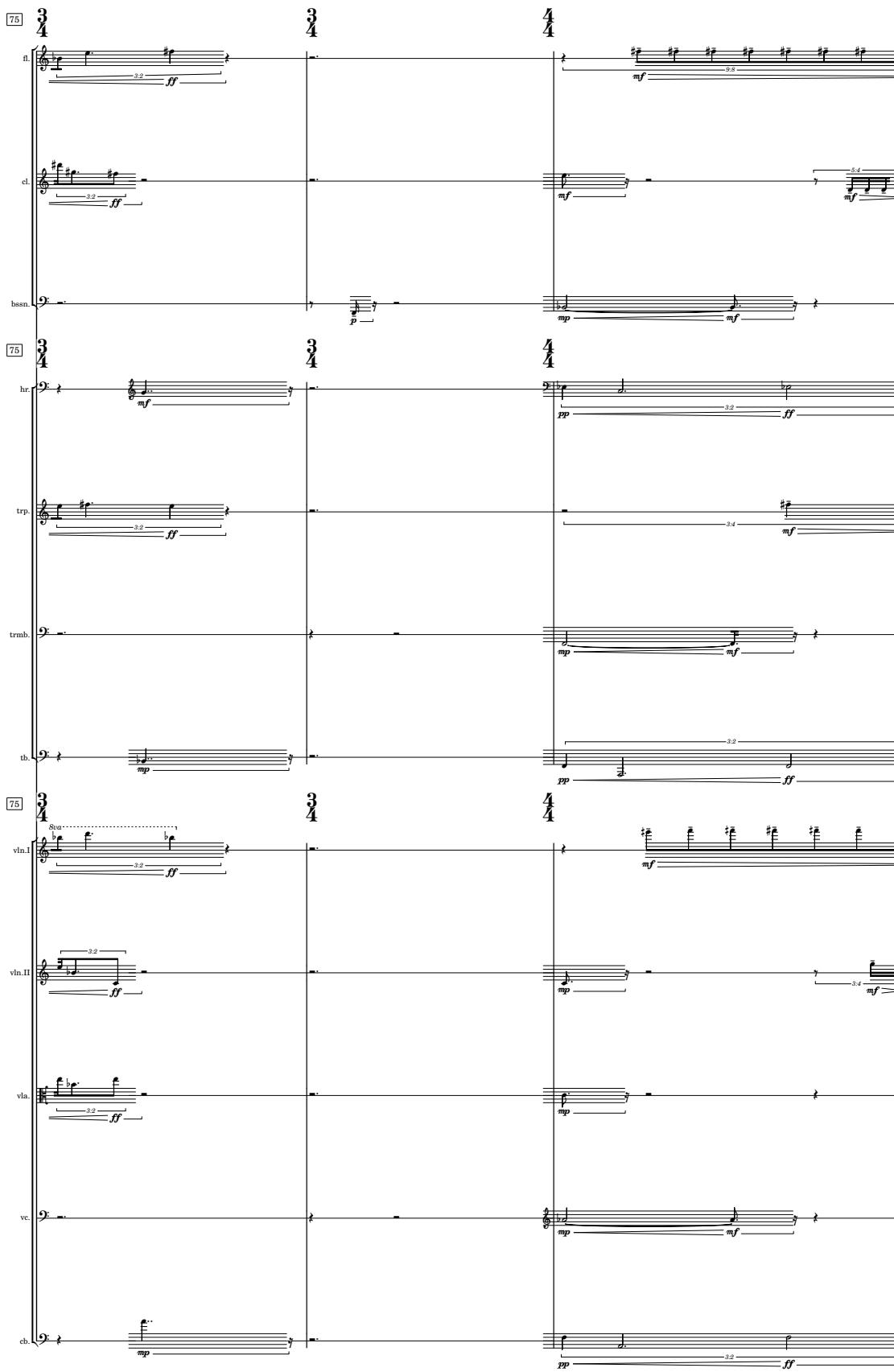
78 

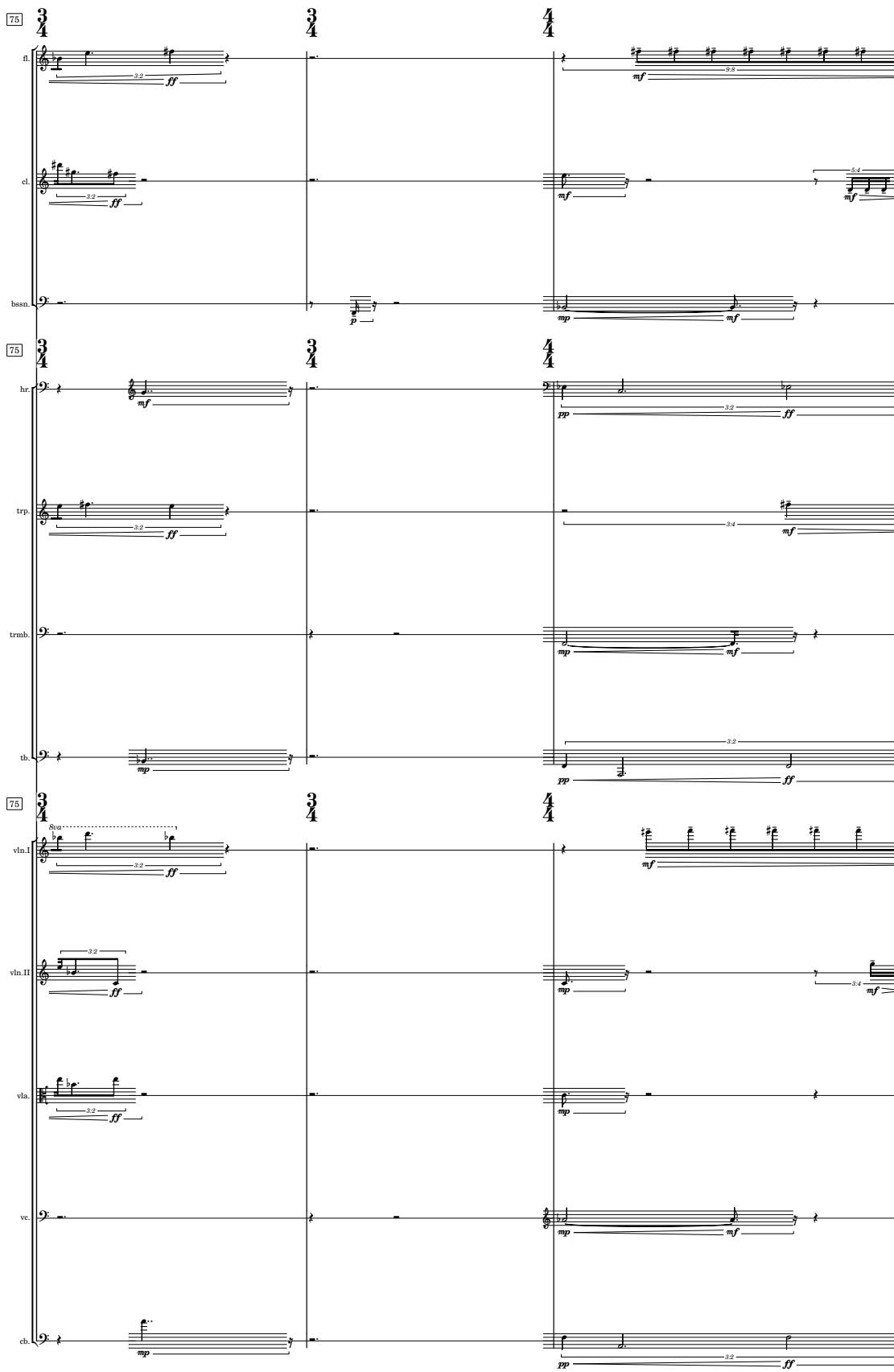
79 

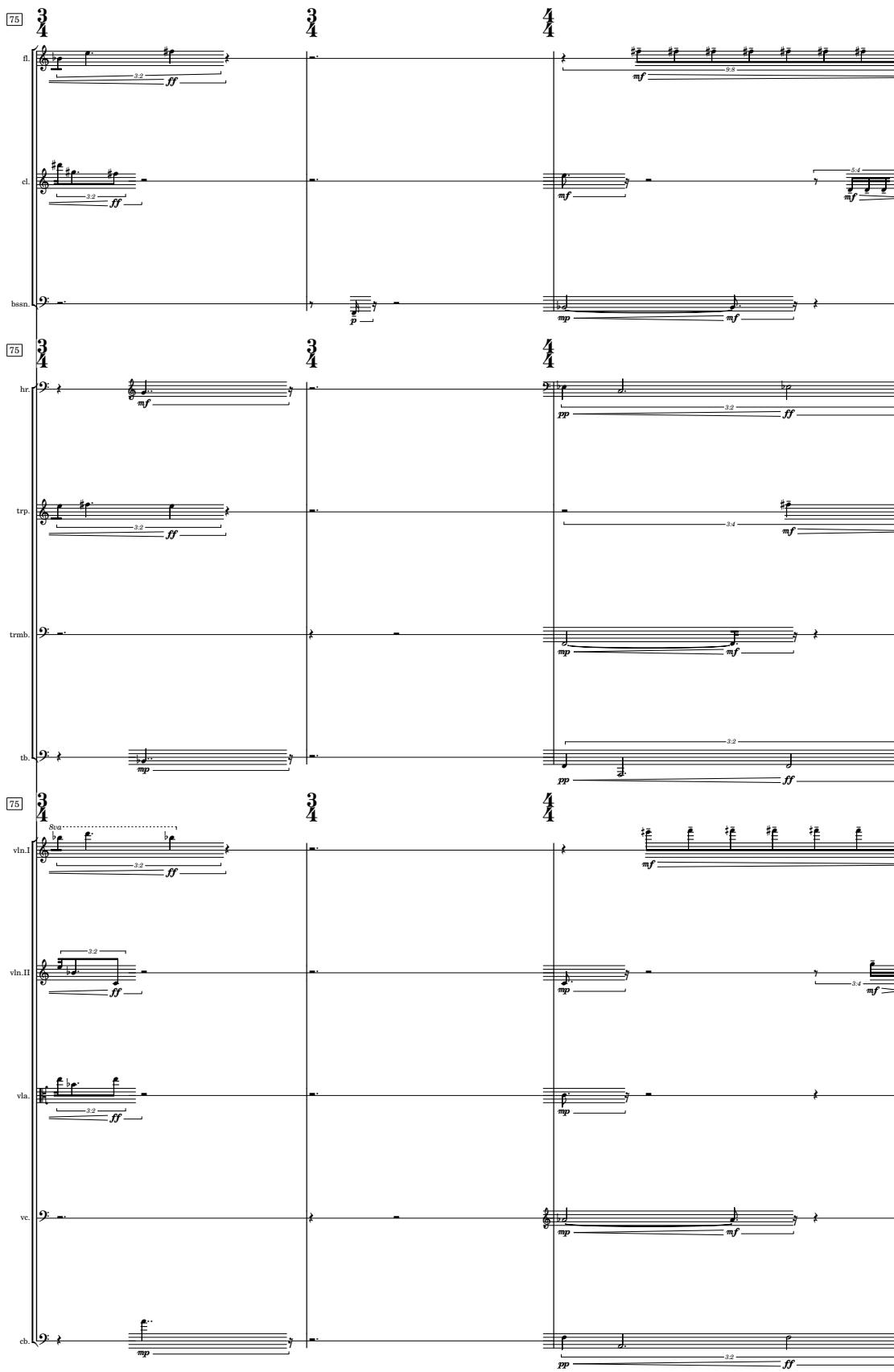
80 

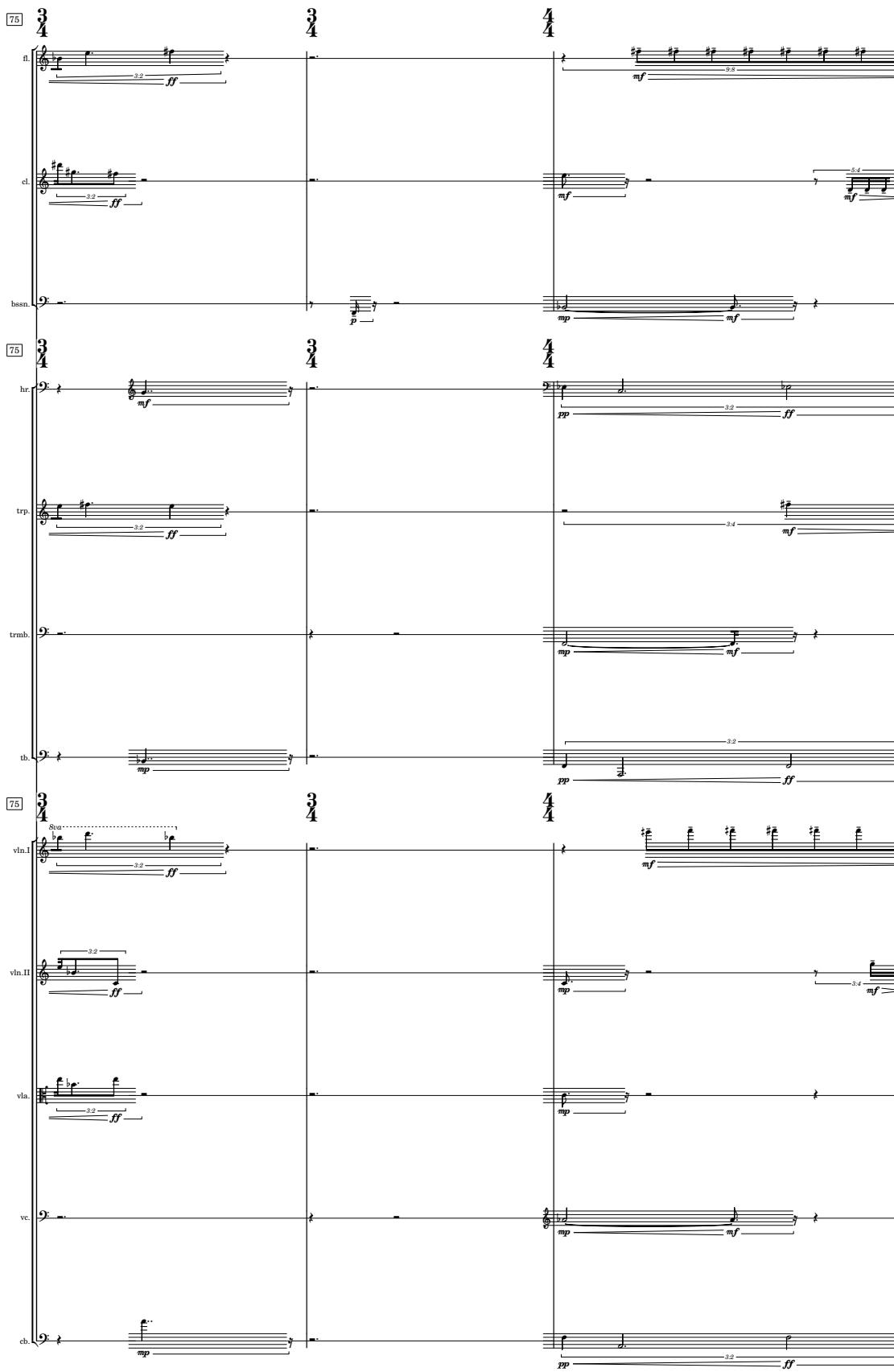
81 

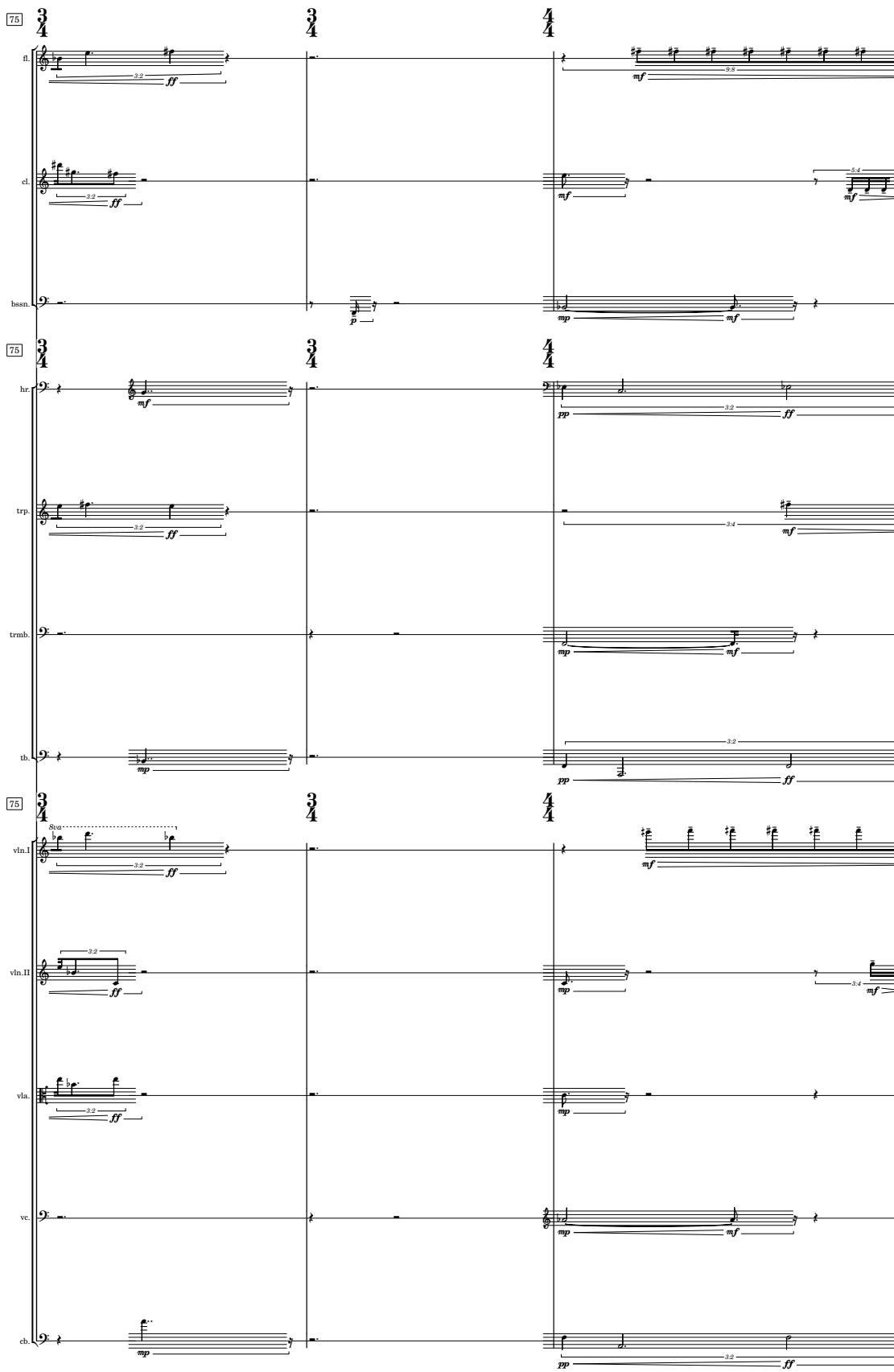
82 

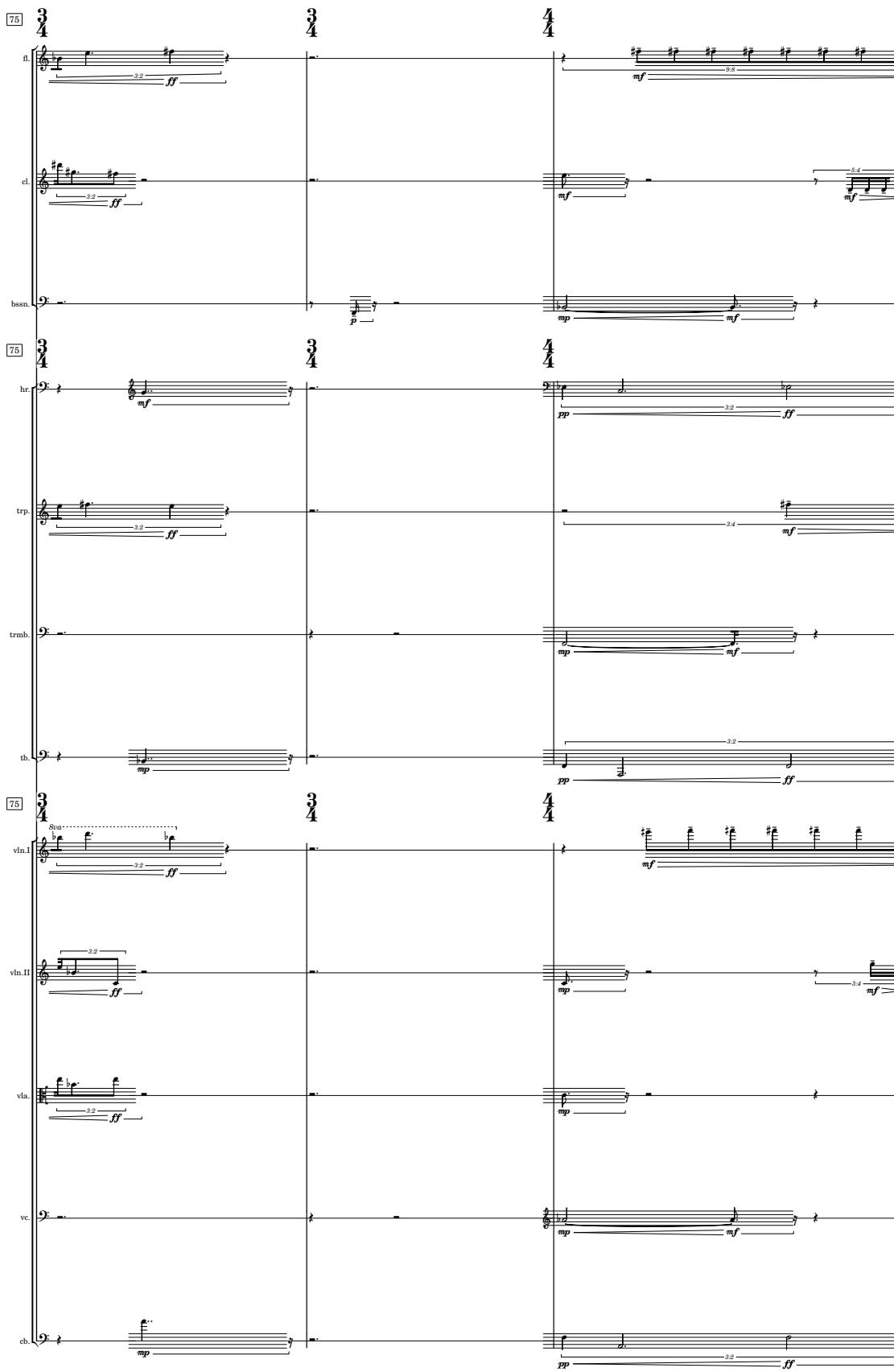
83 

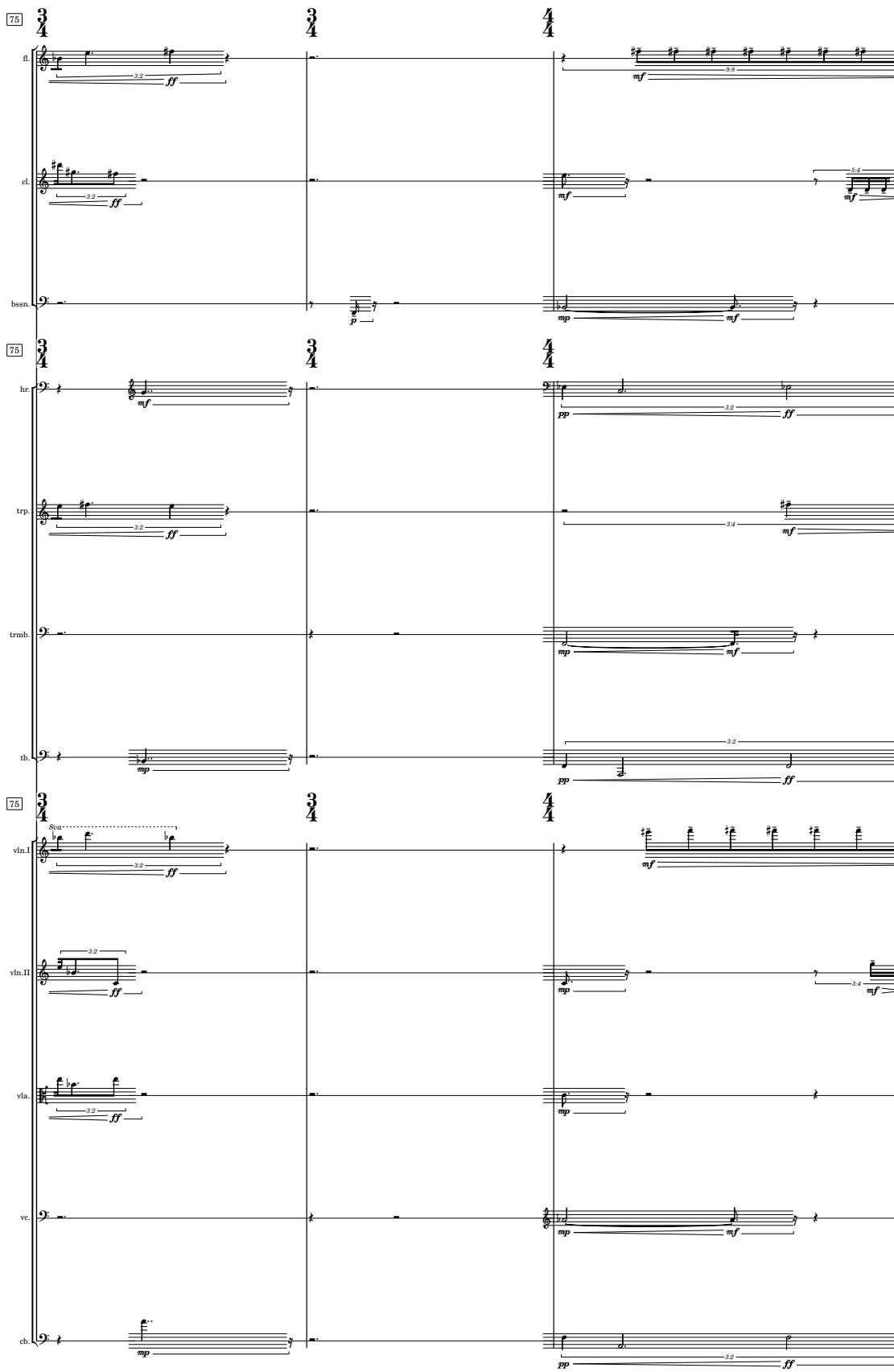
84 

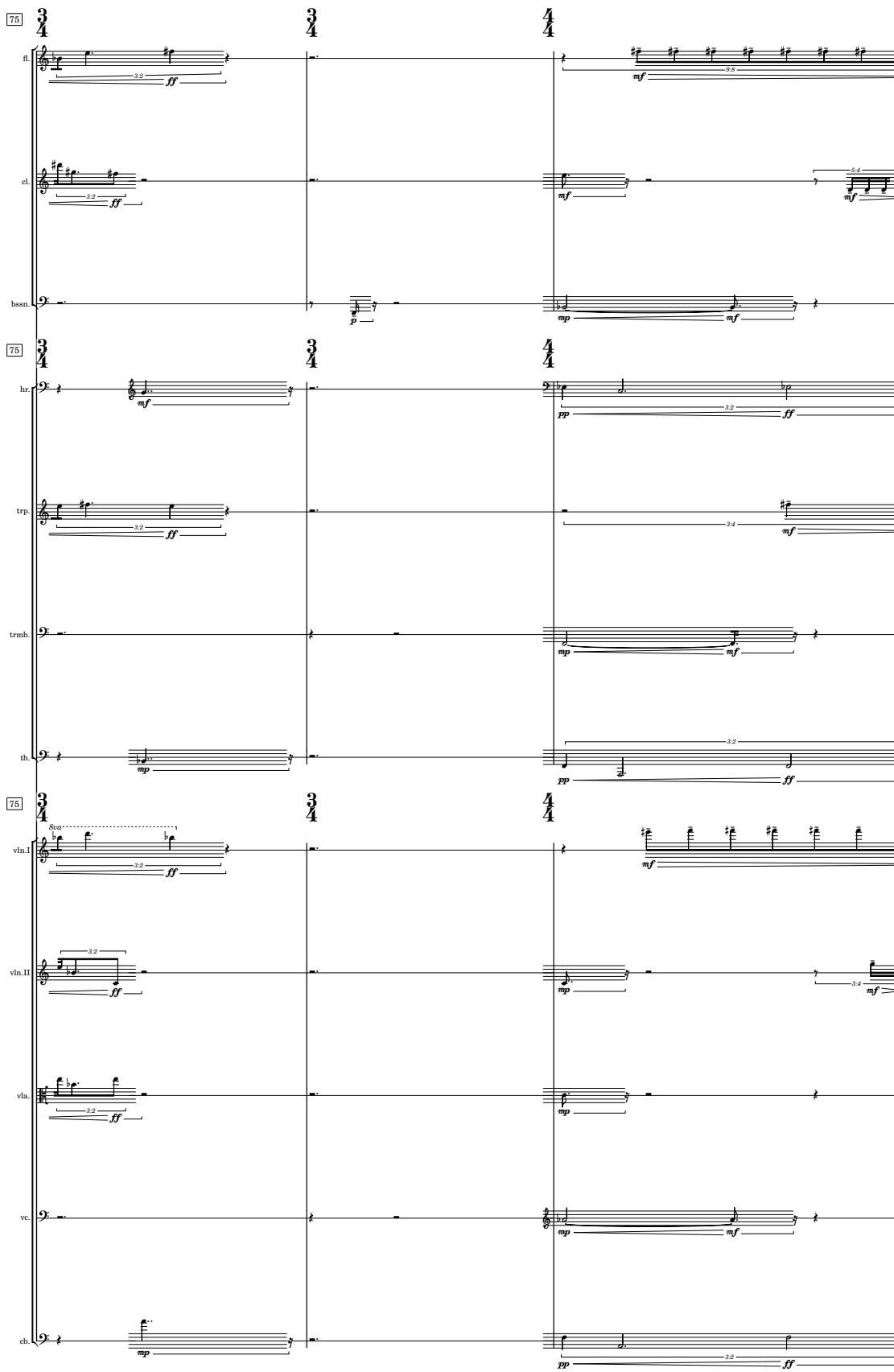
85 

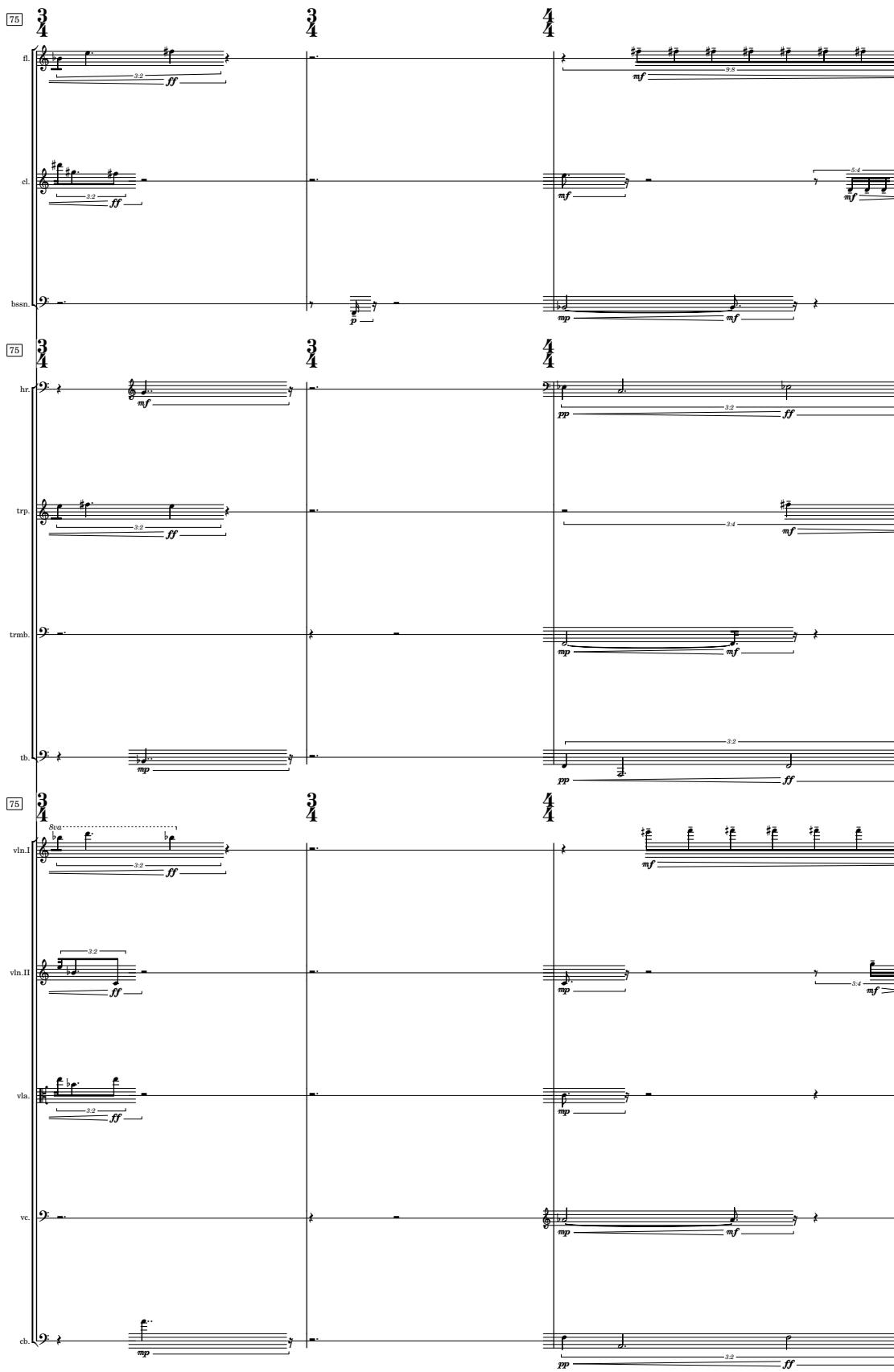
86 

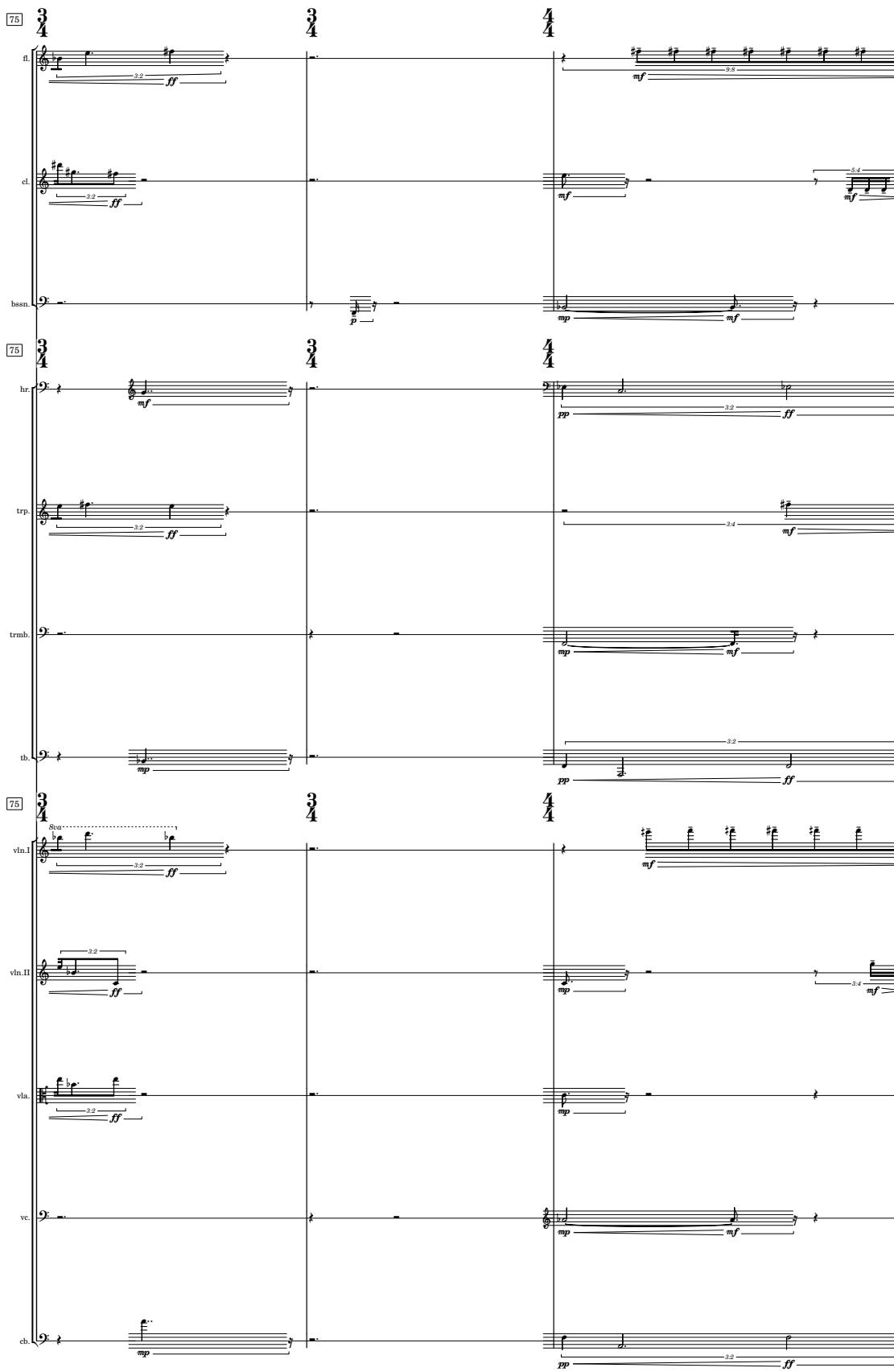
87 

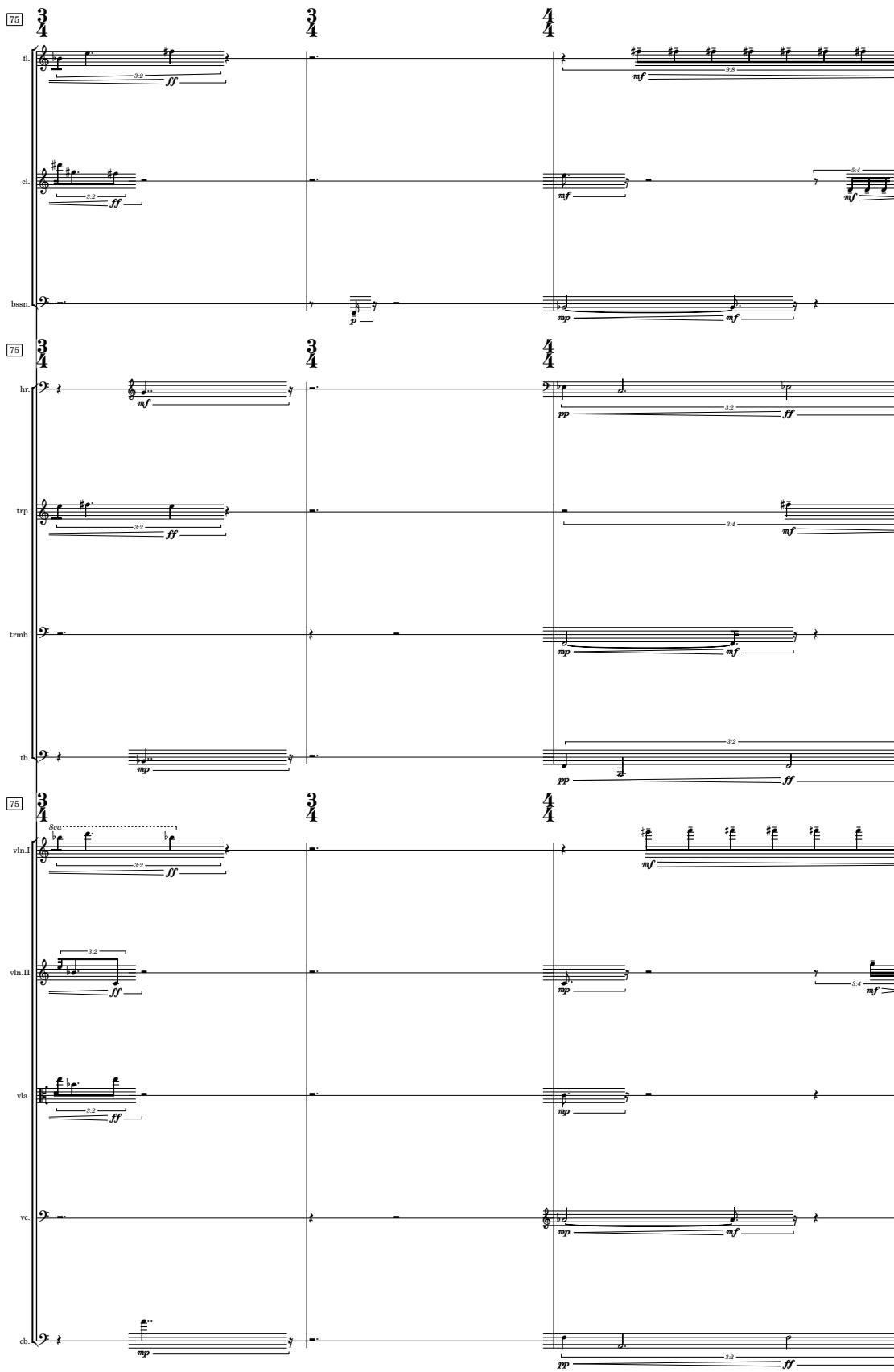
88 

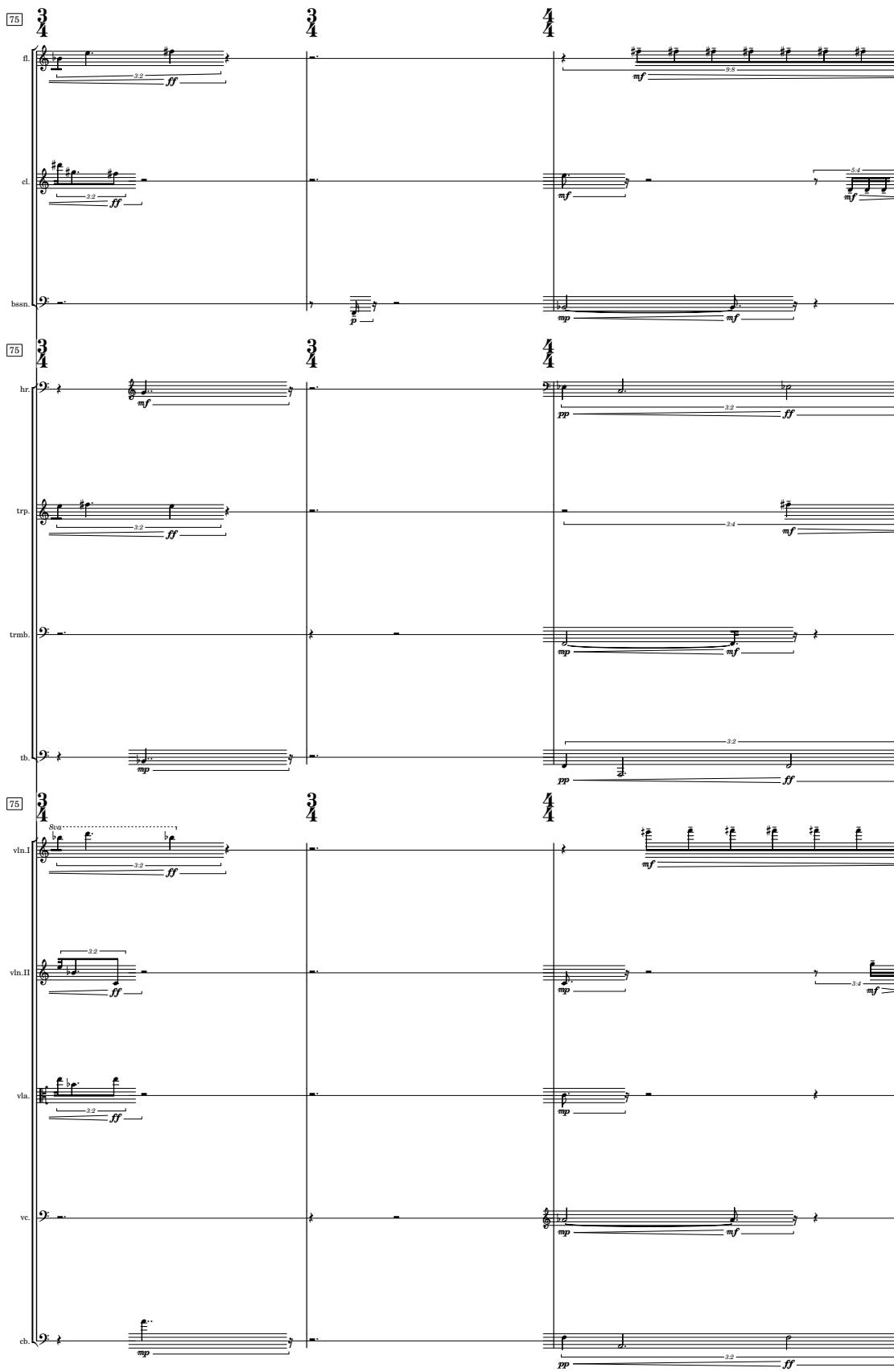
89 

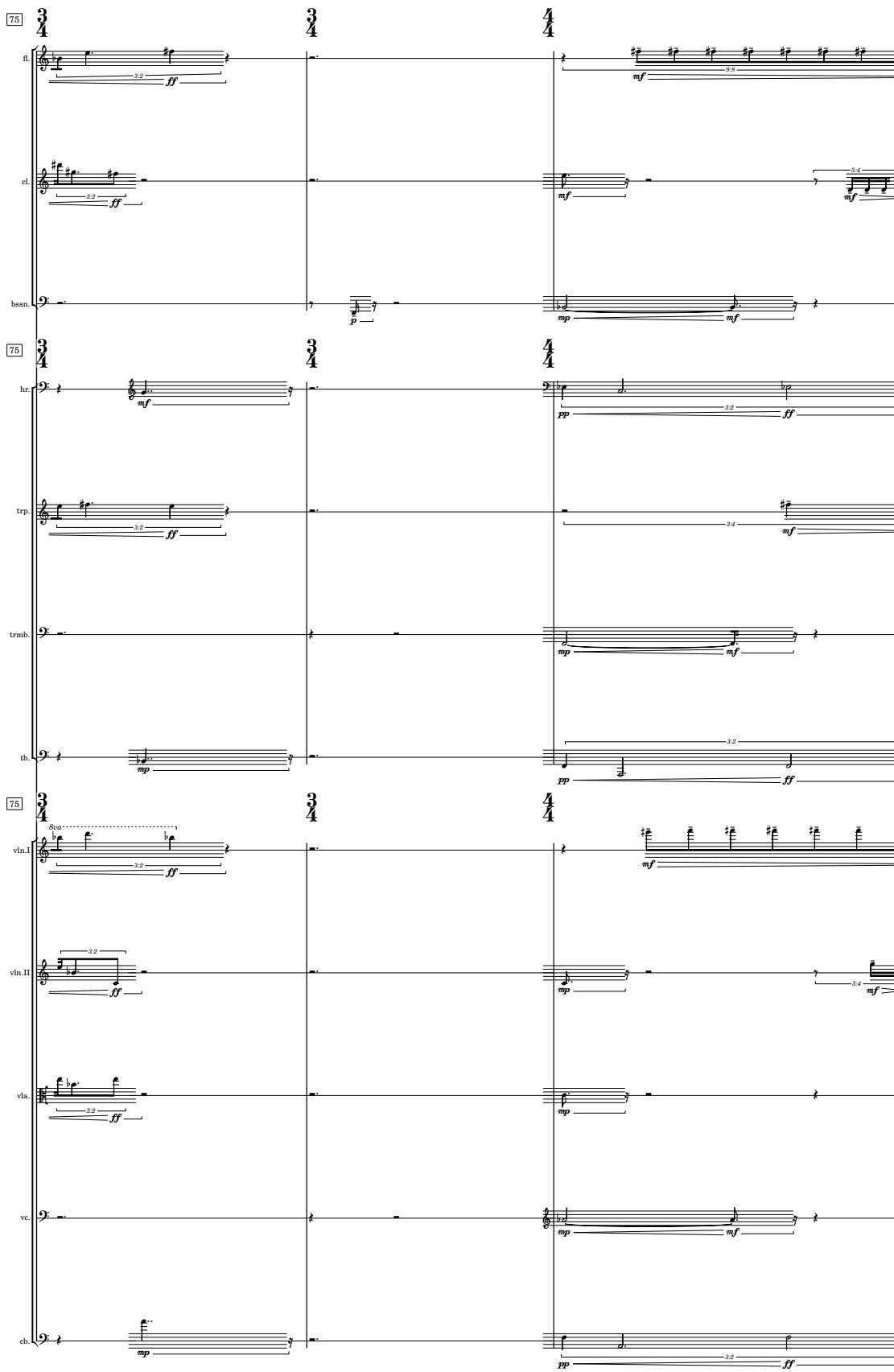
90 

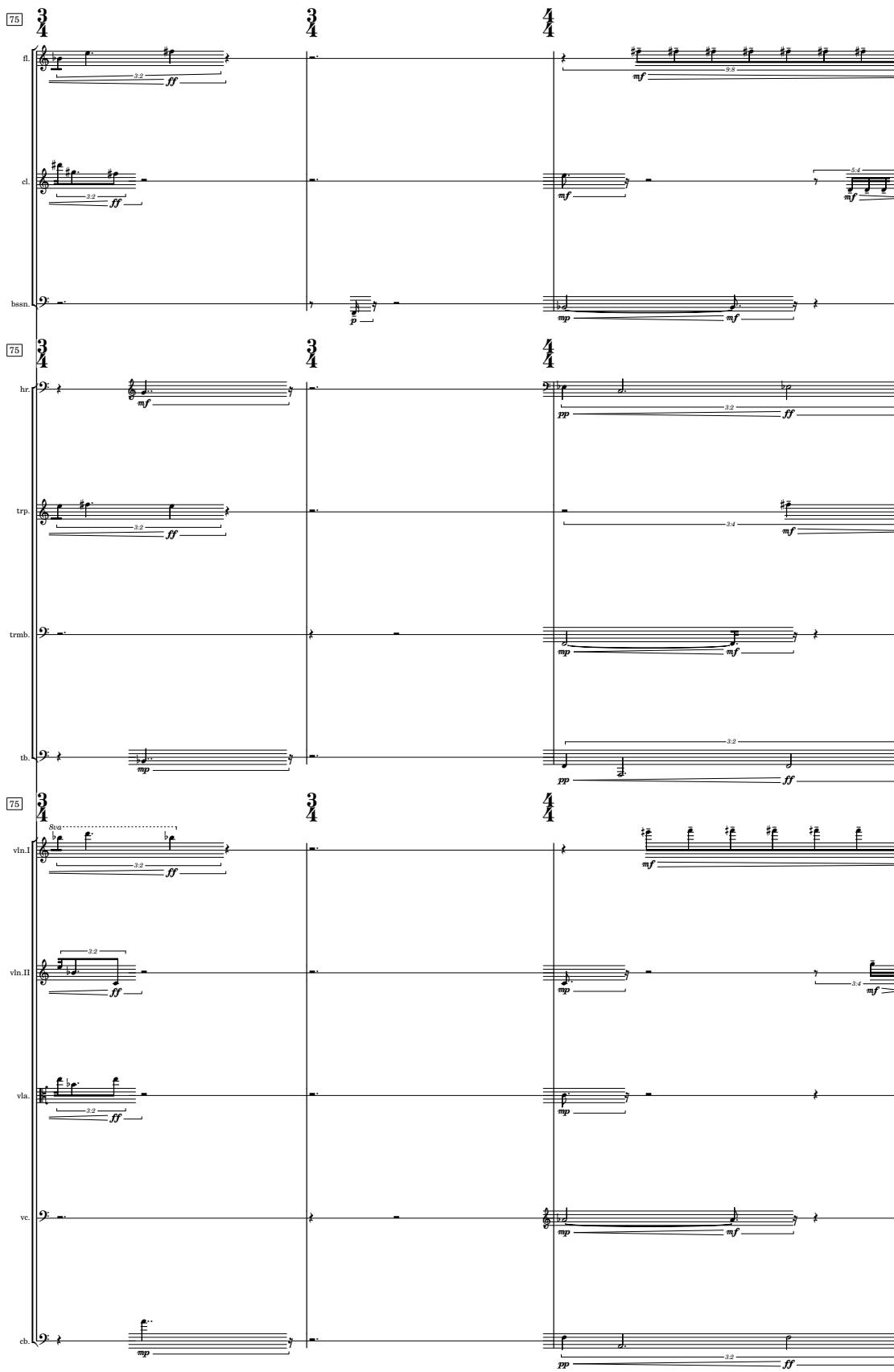
91 

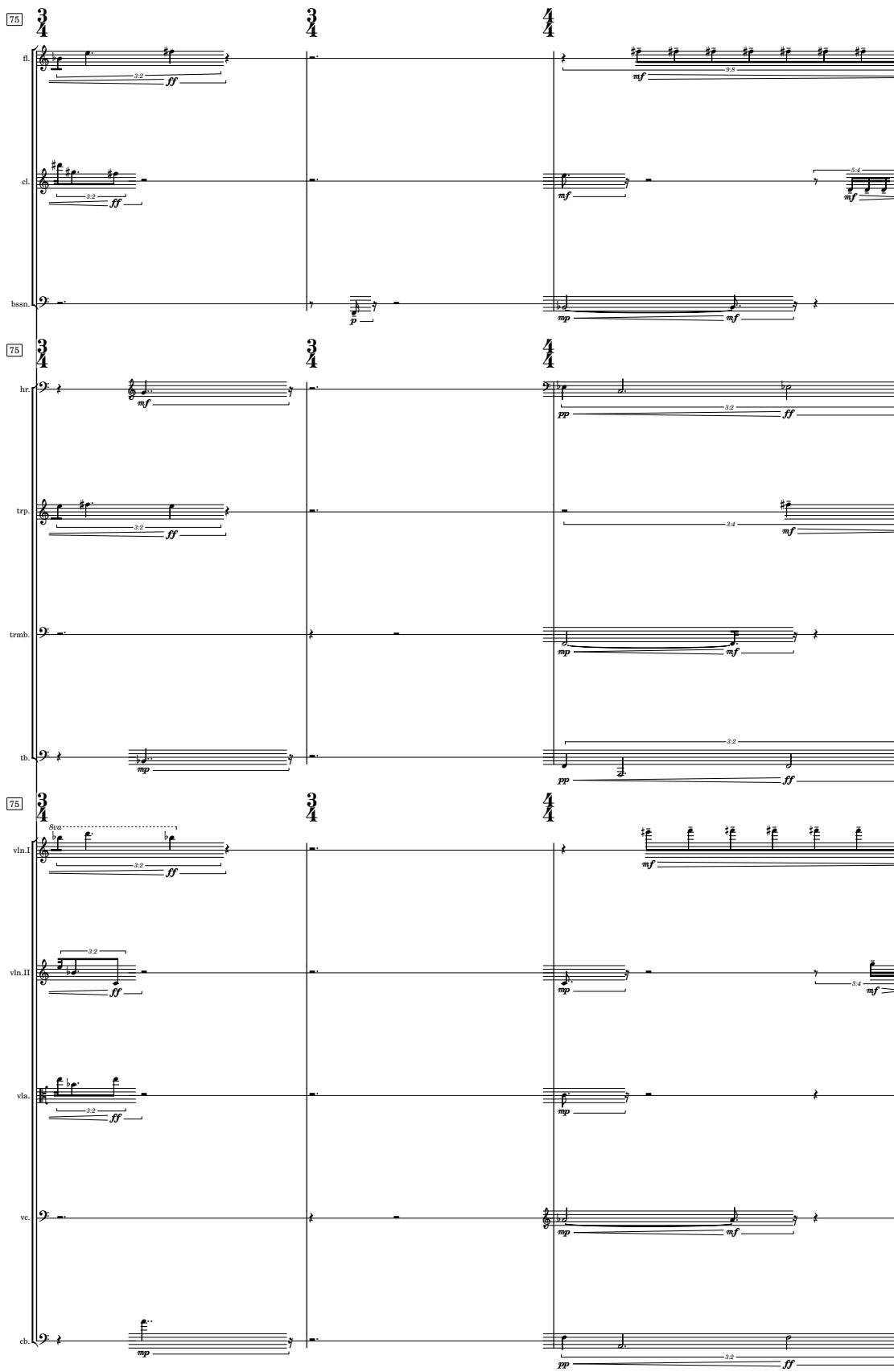
92 

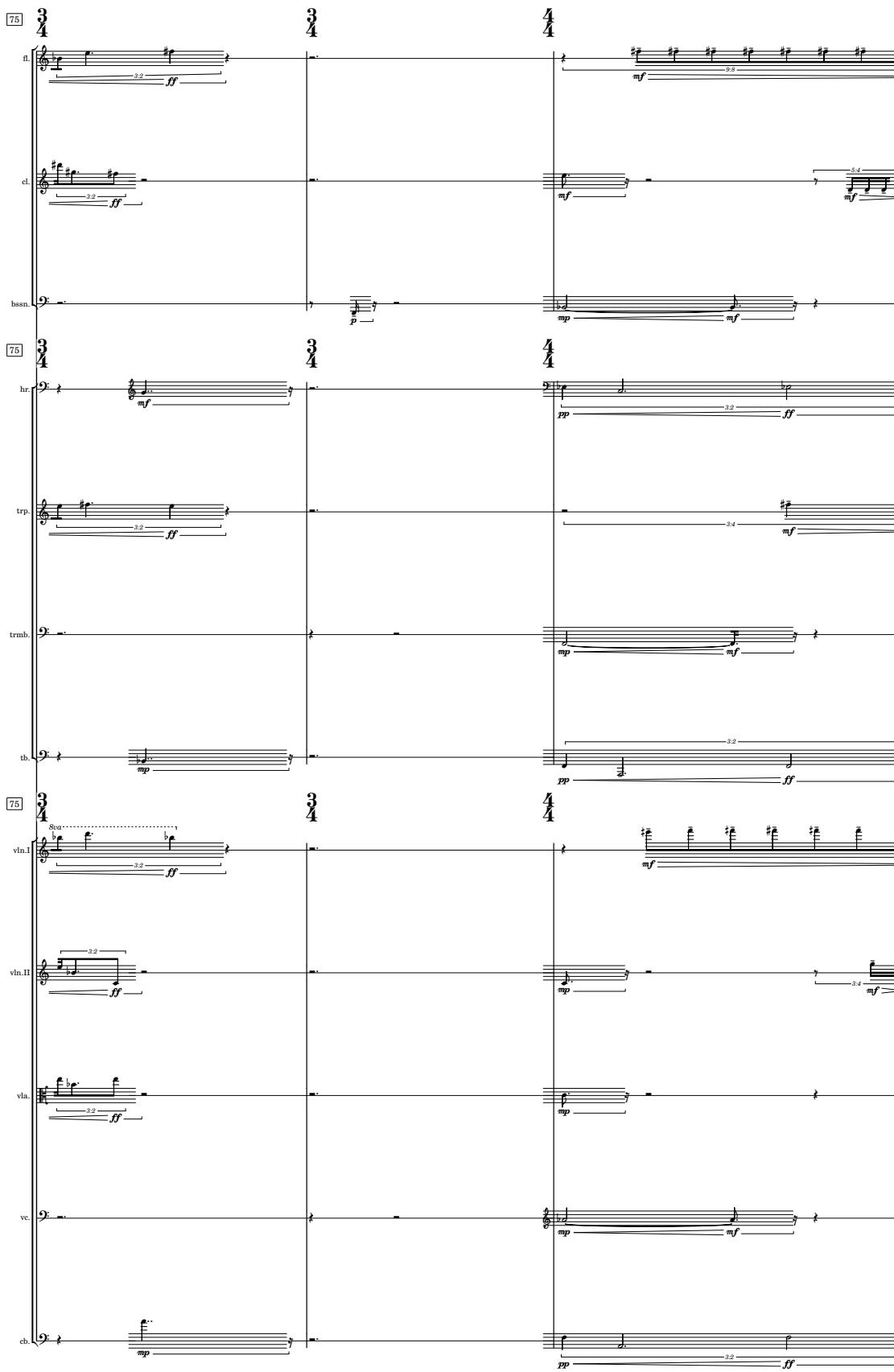
93 

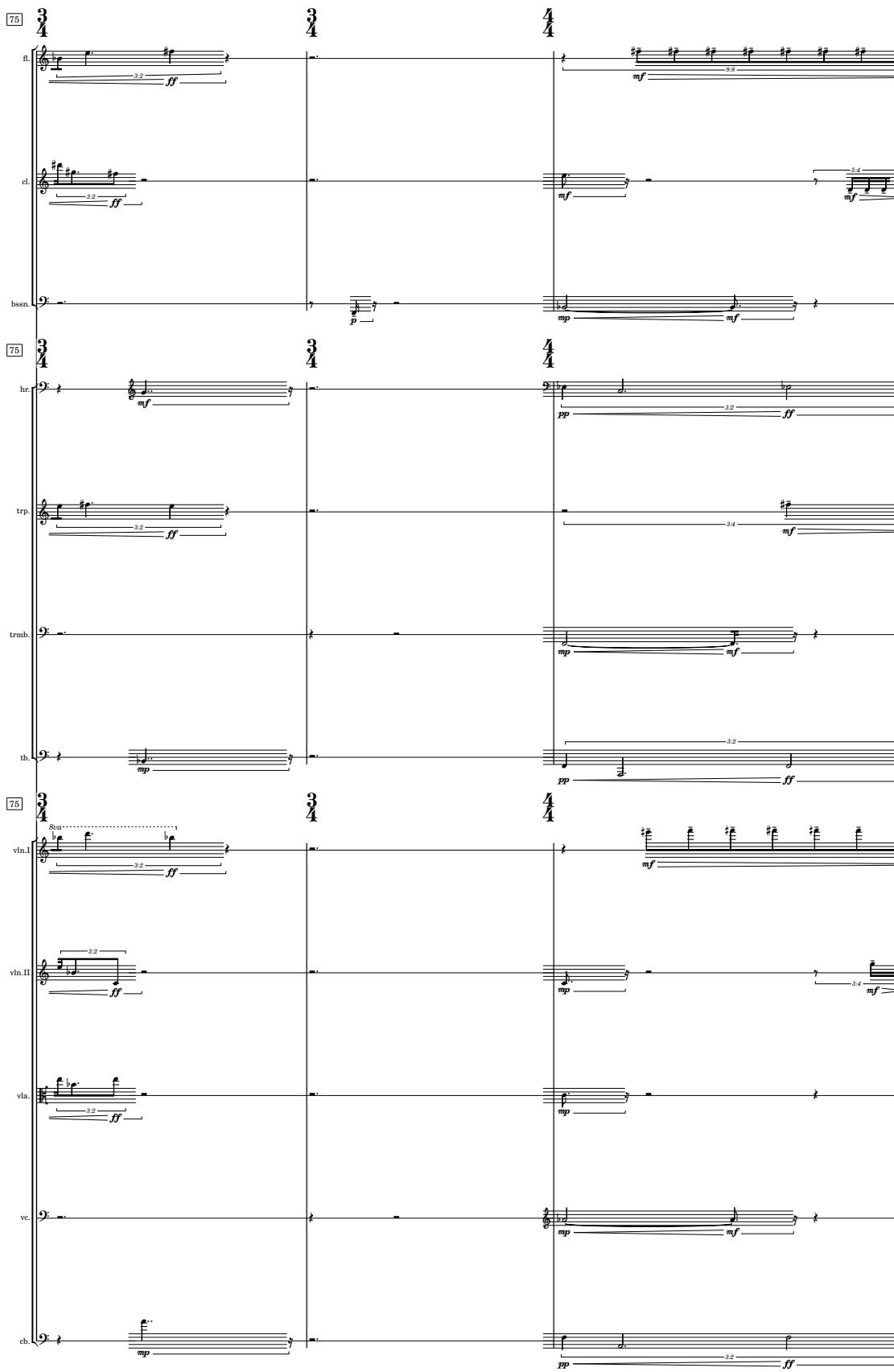
94 

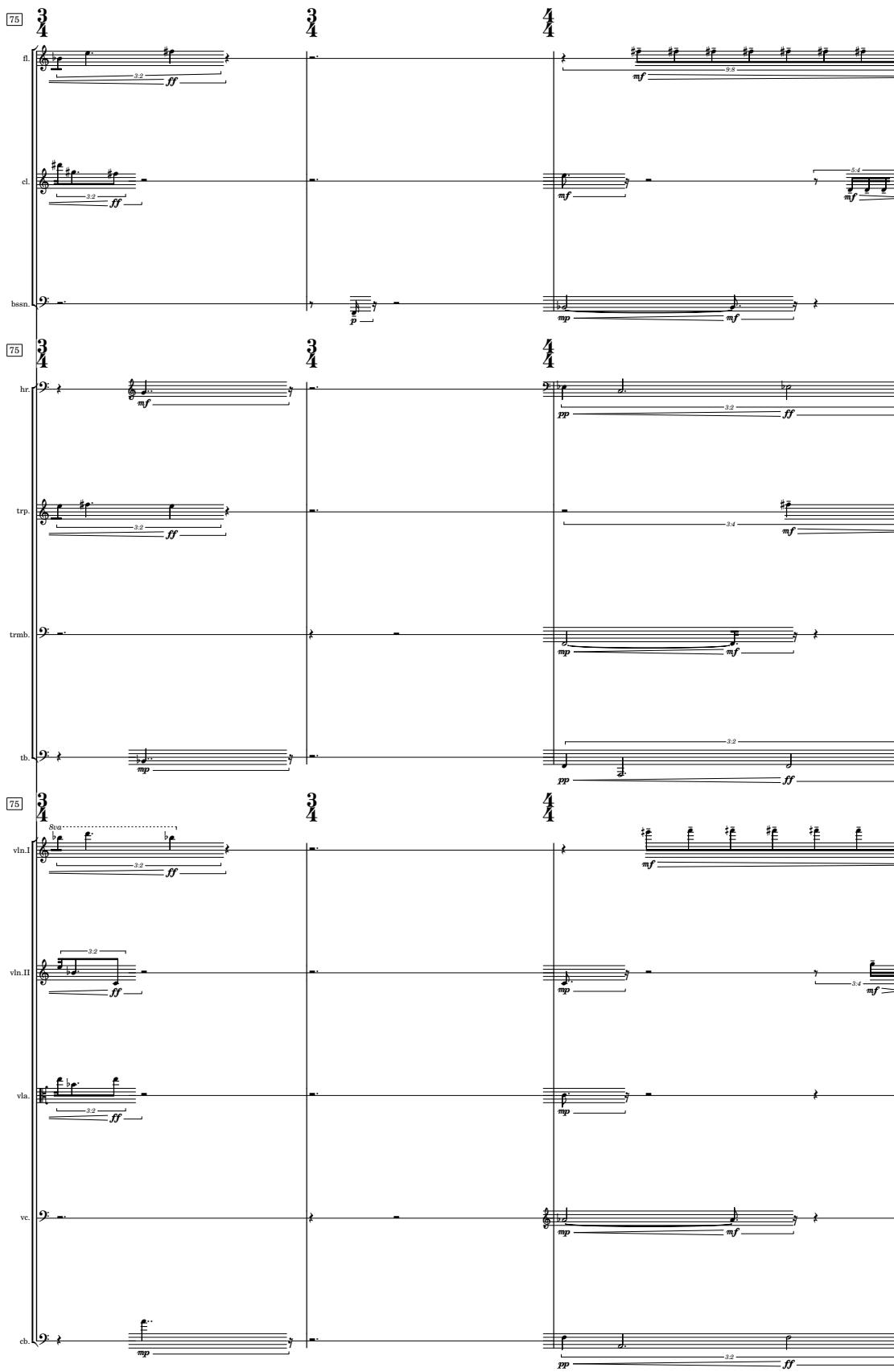
95 

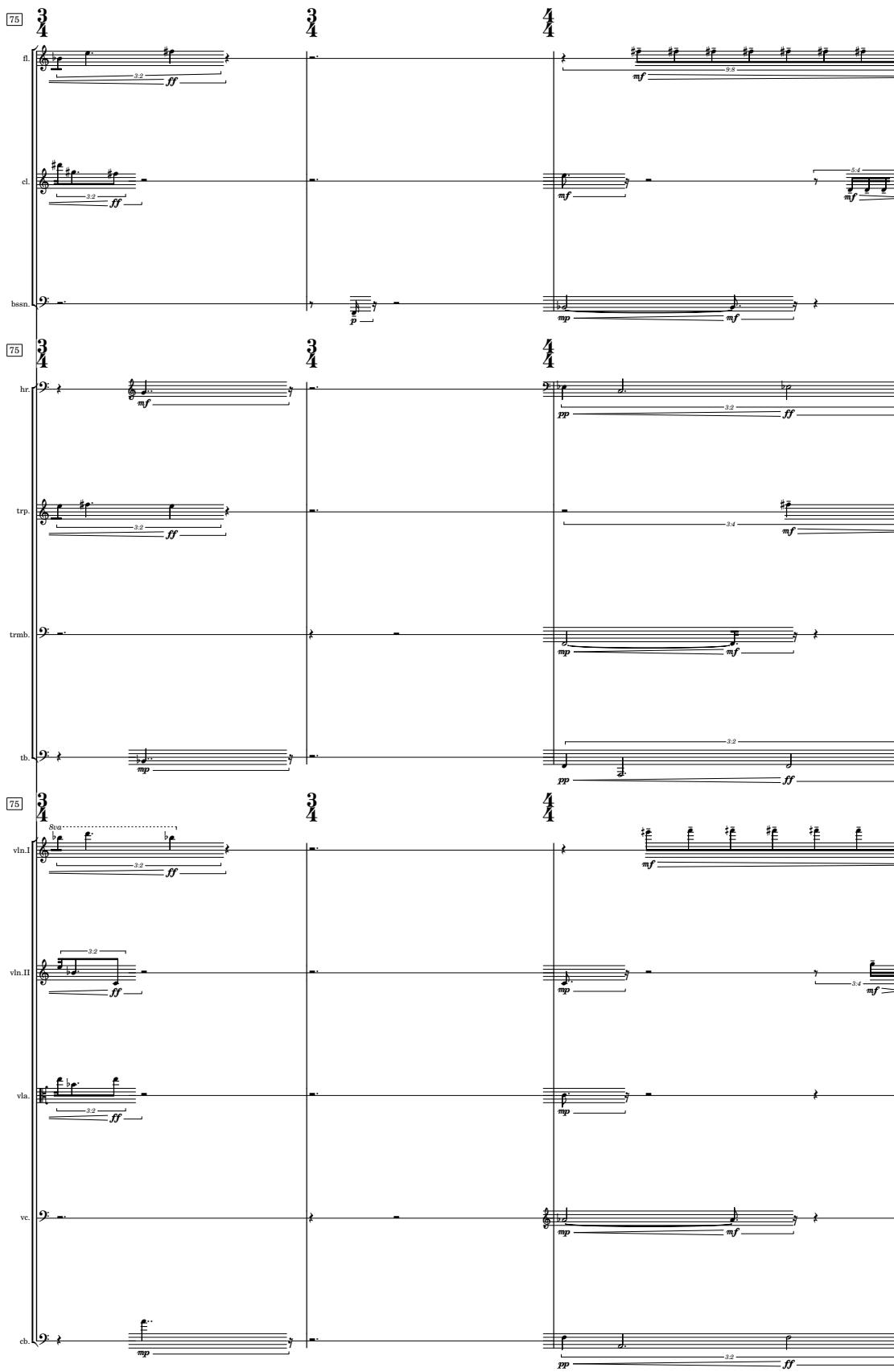
96 

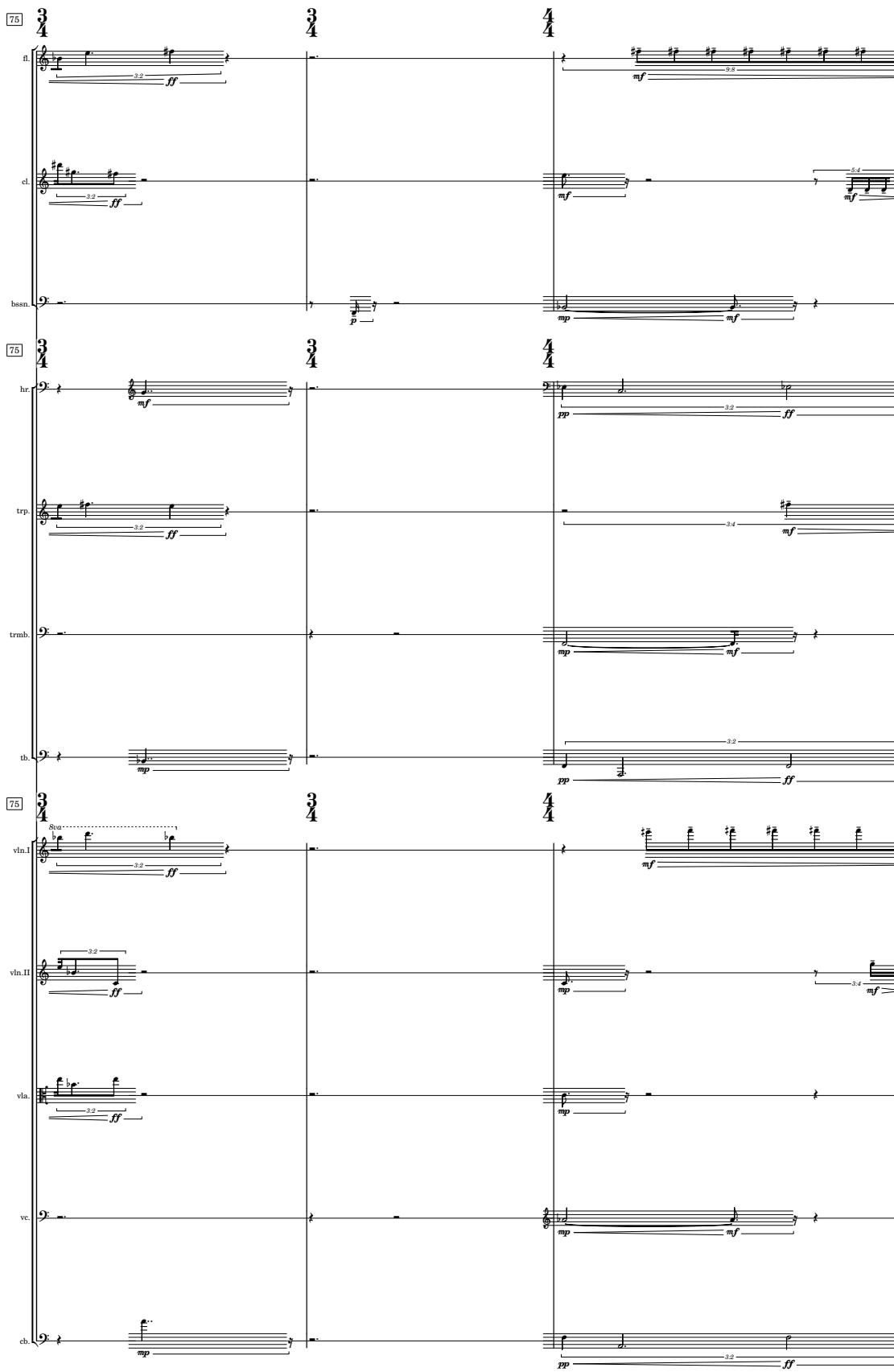
97 

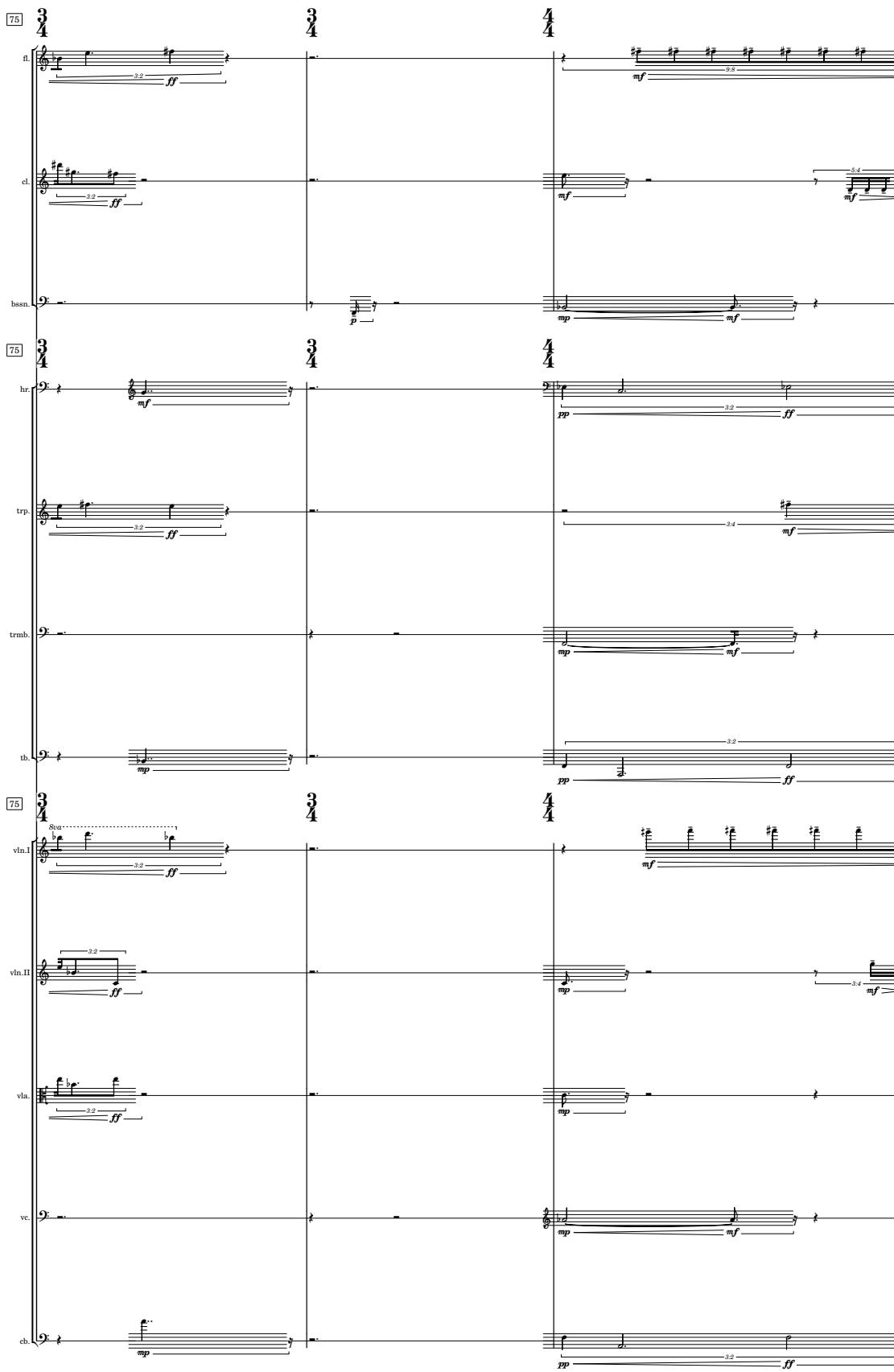
98 

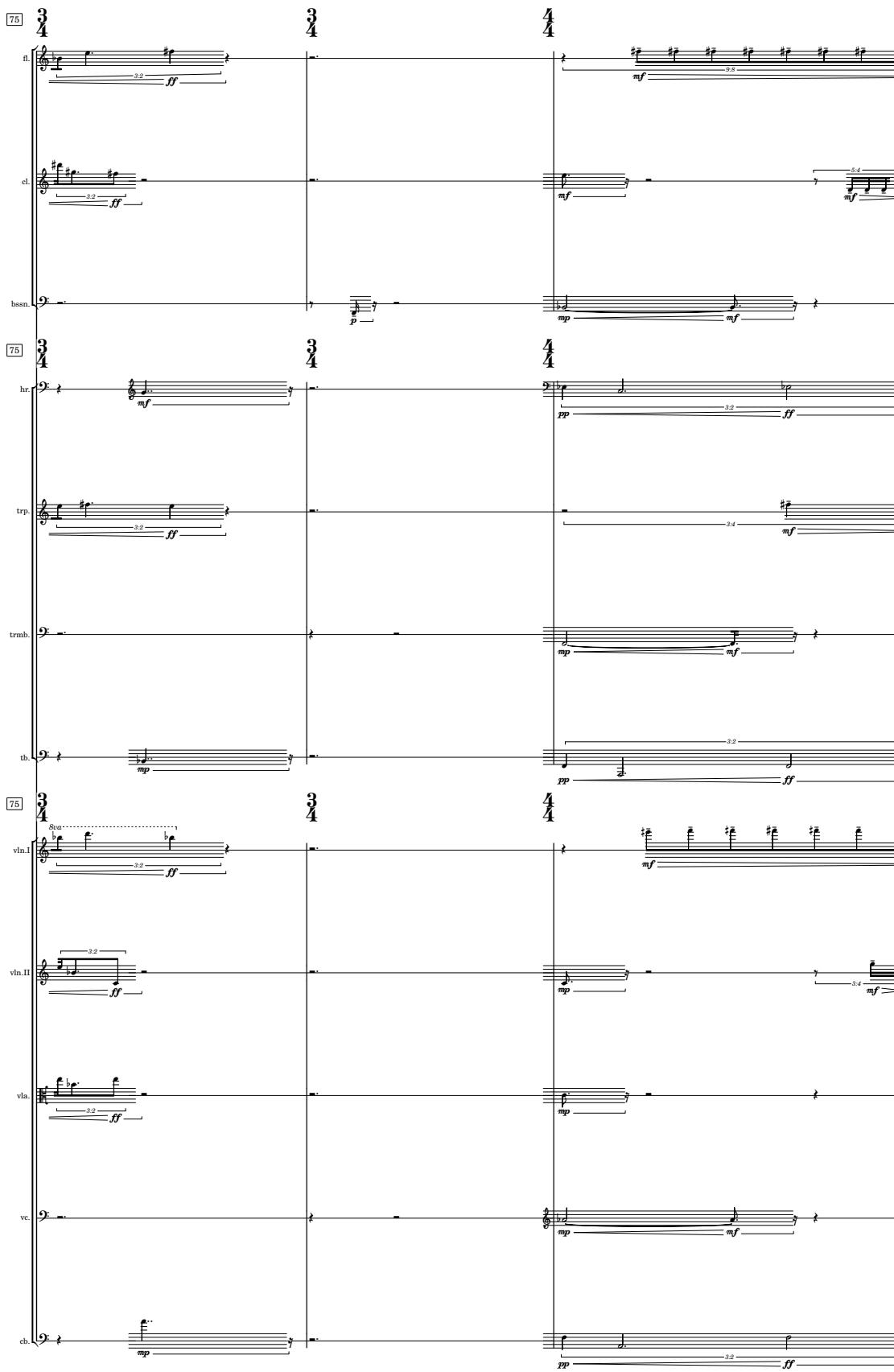
99 

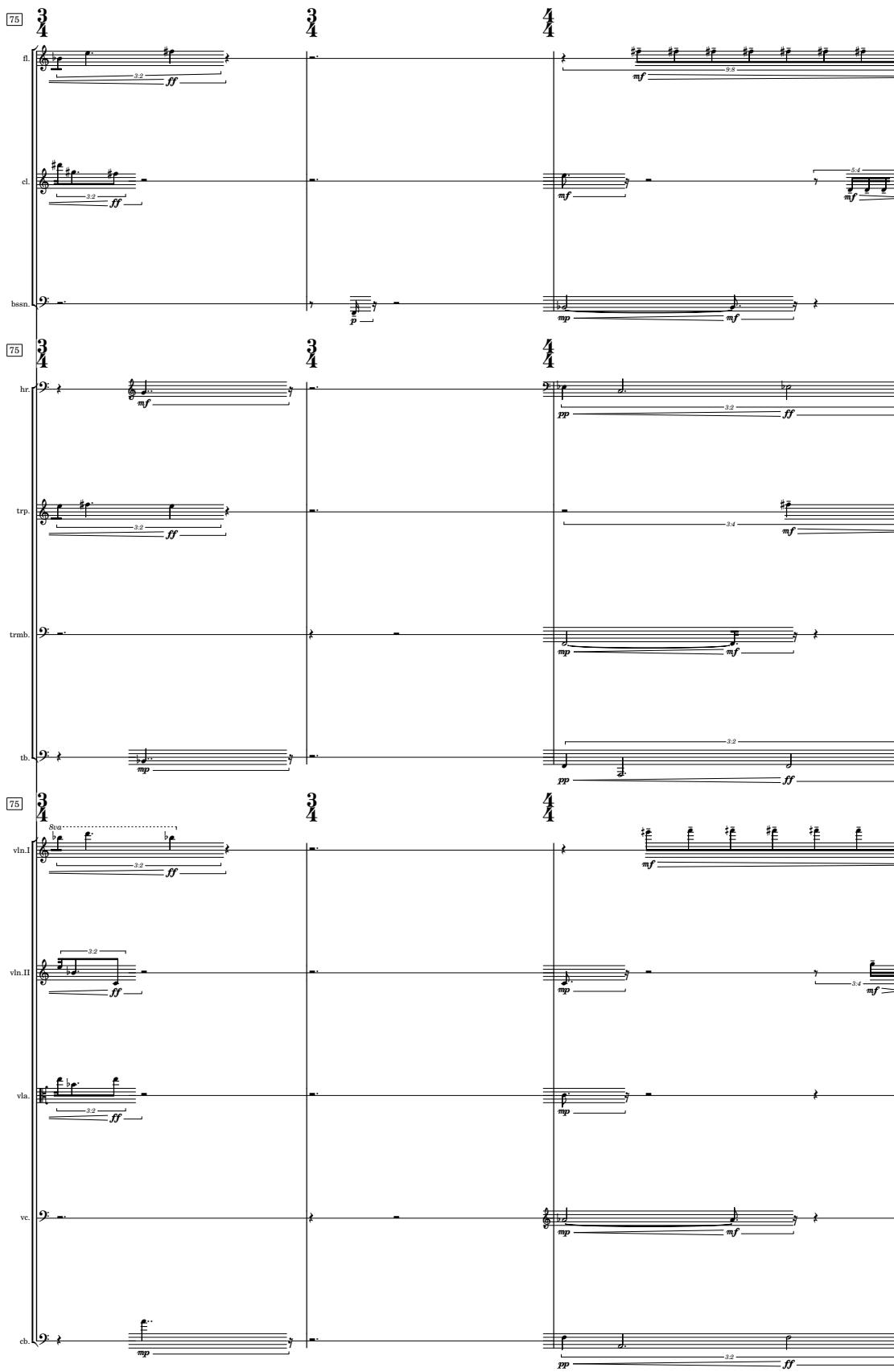
100 

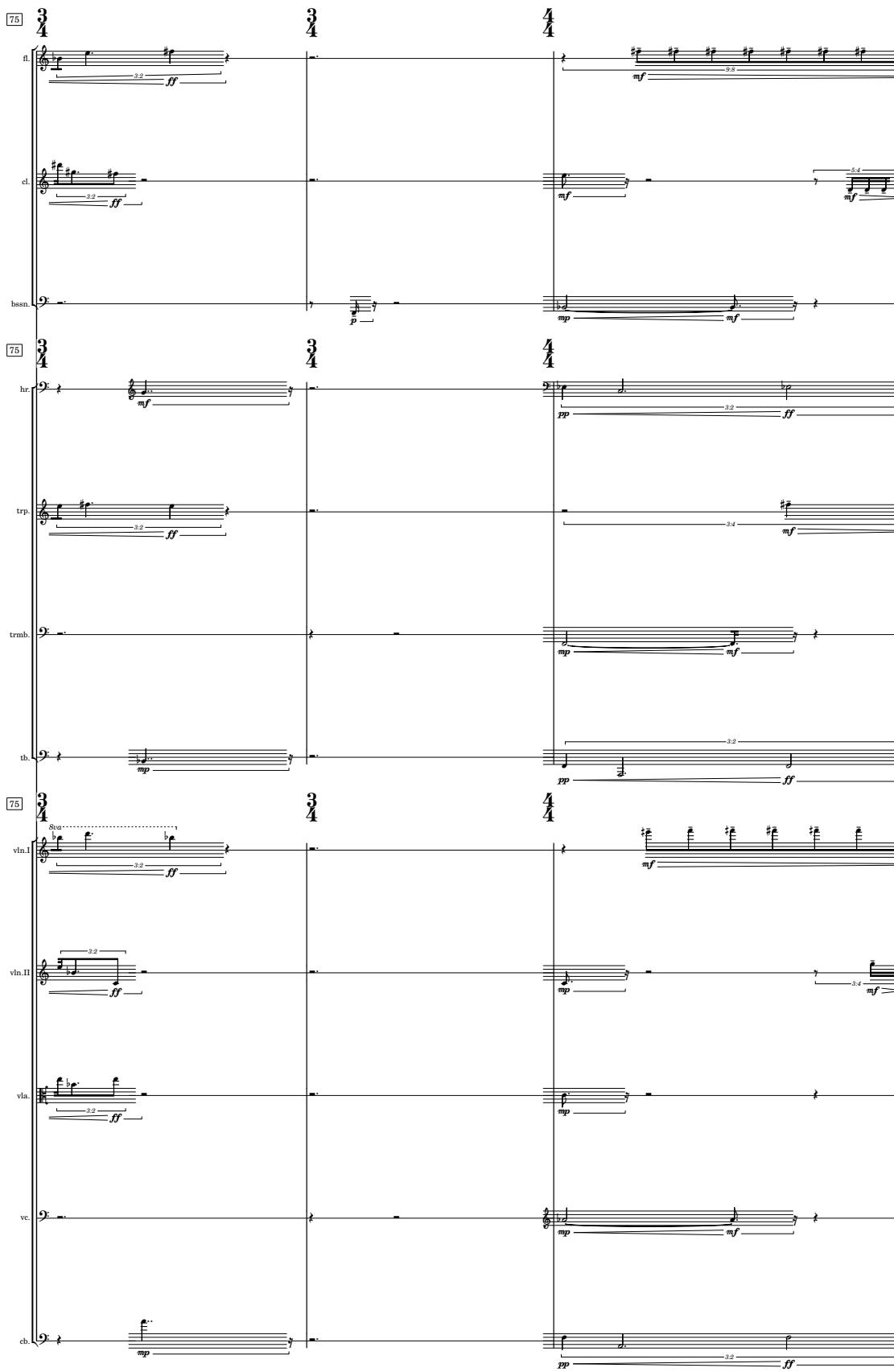
101 

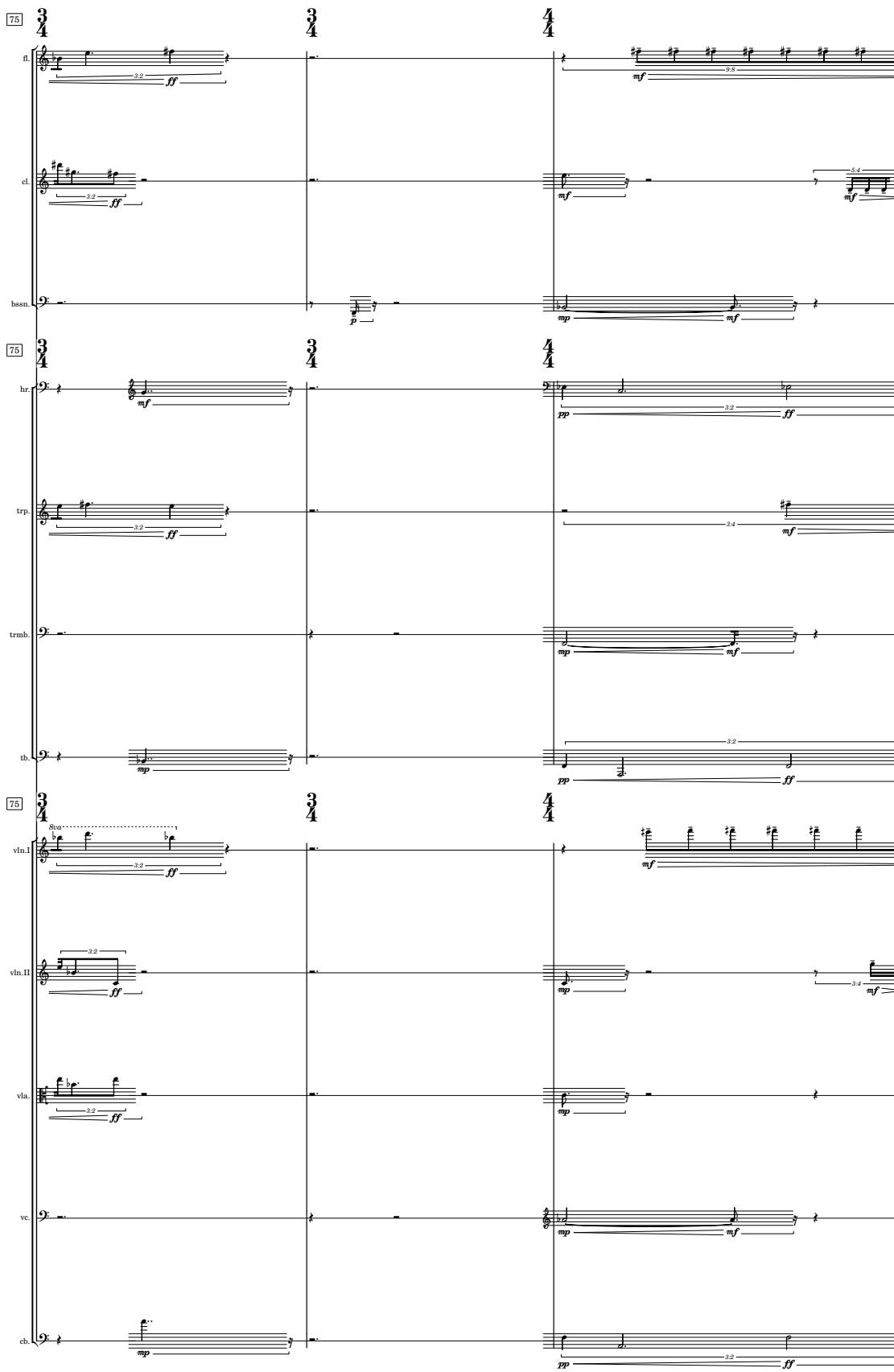
102 

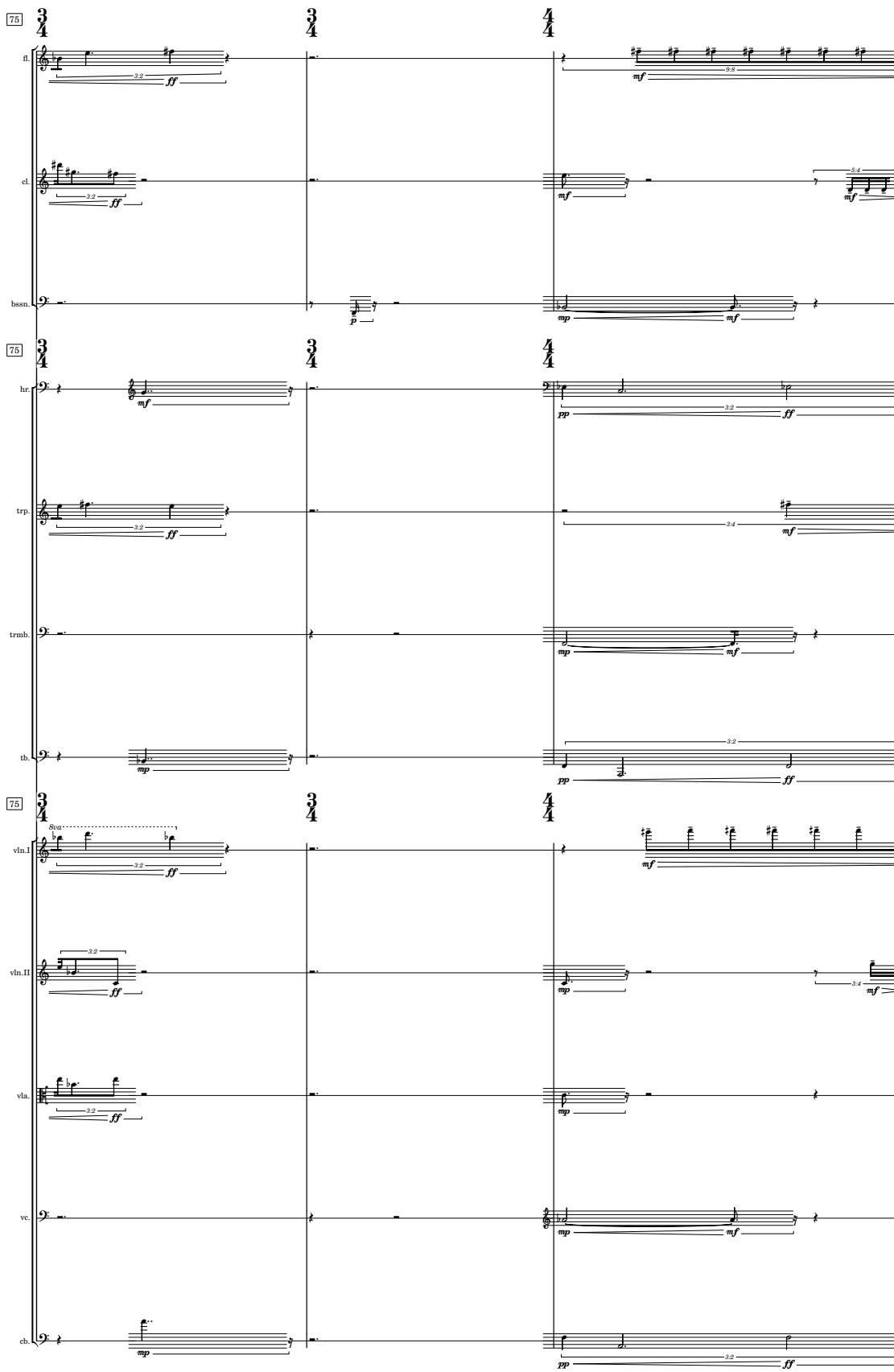
103 

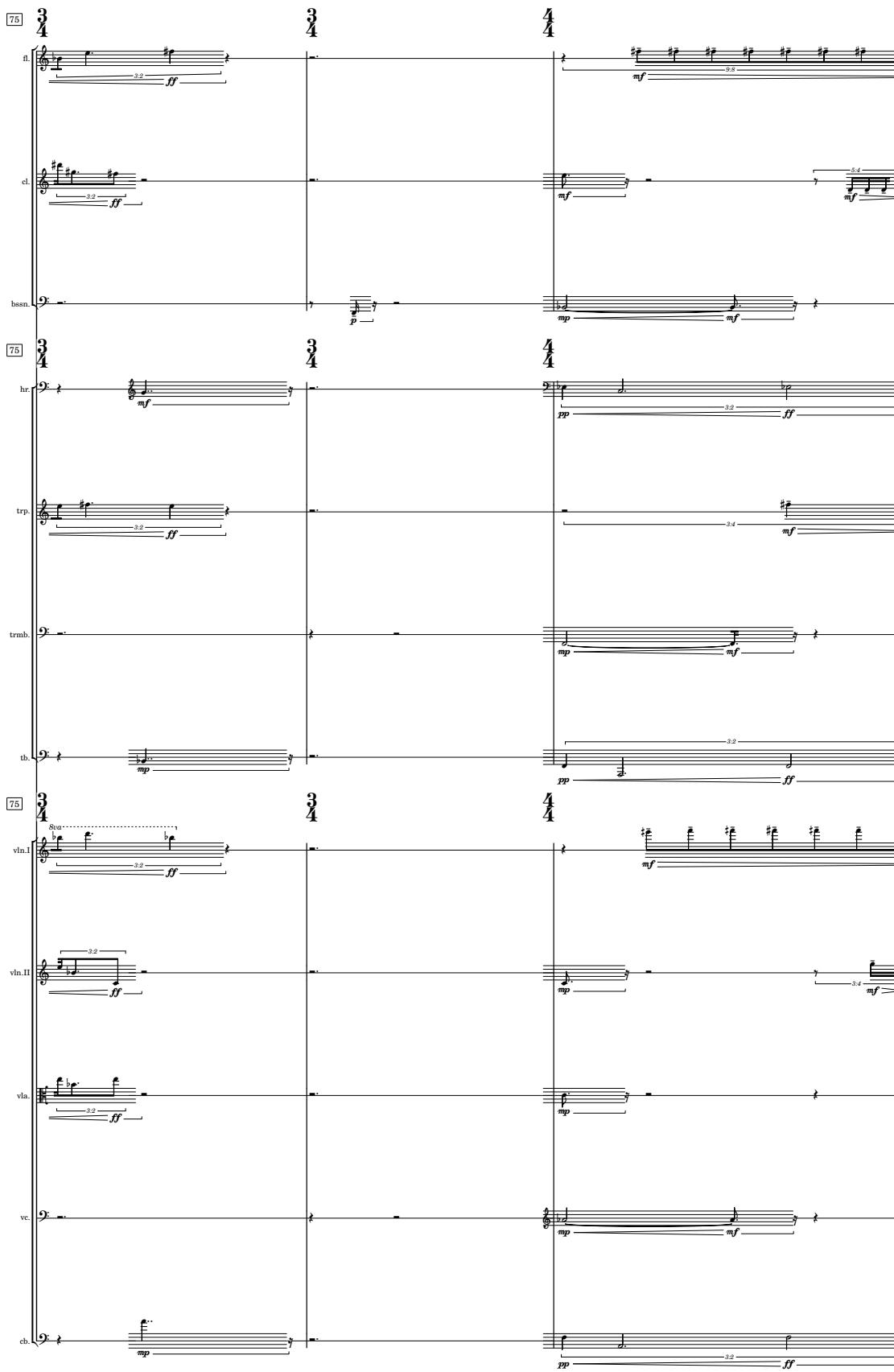
104 

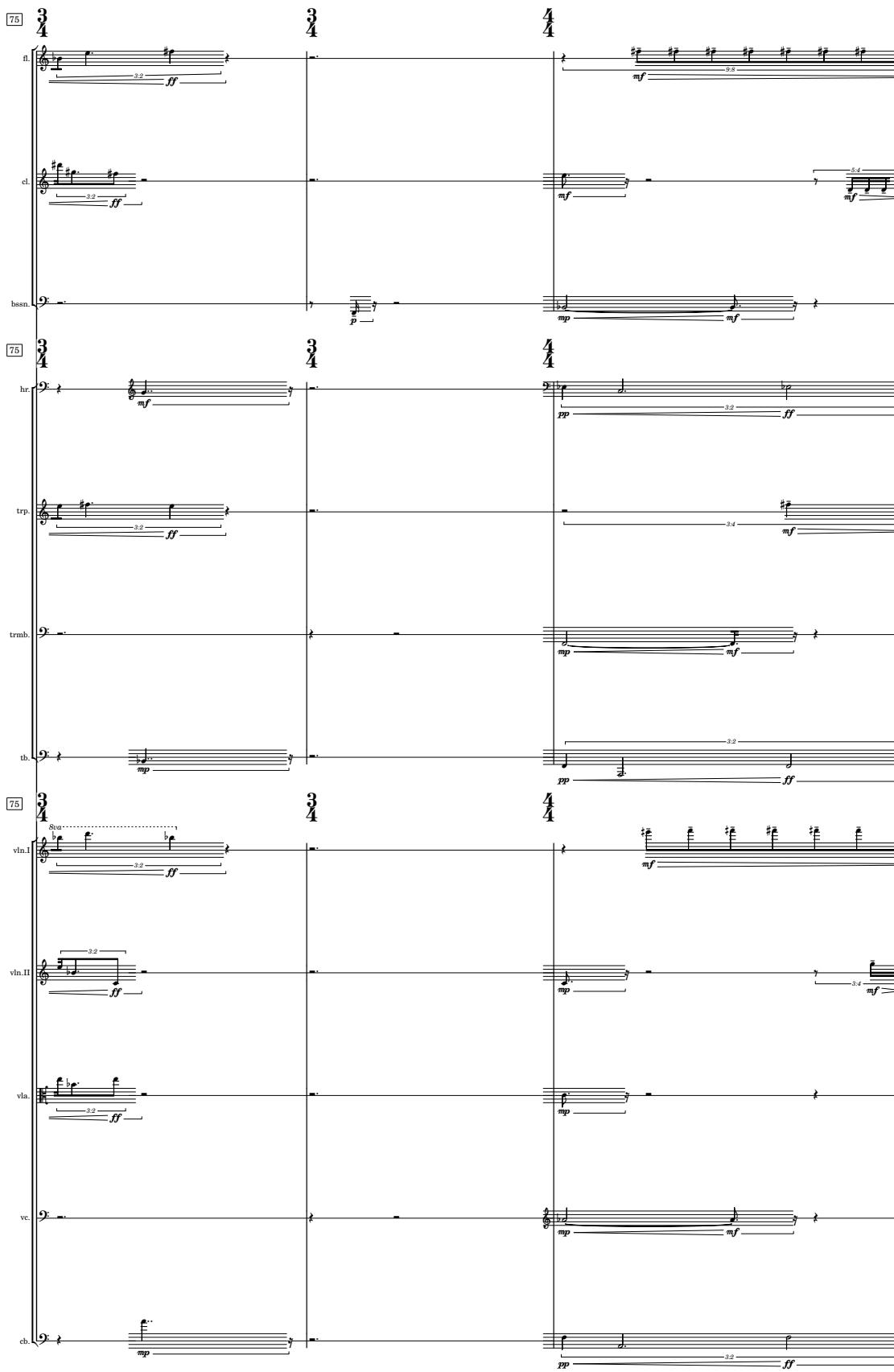
105 

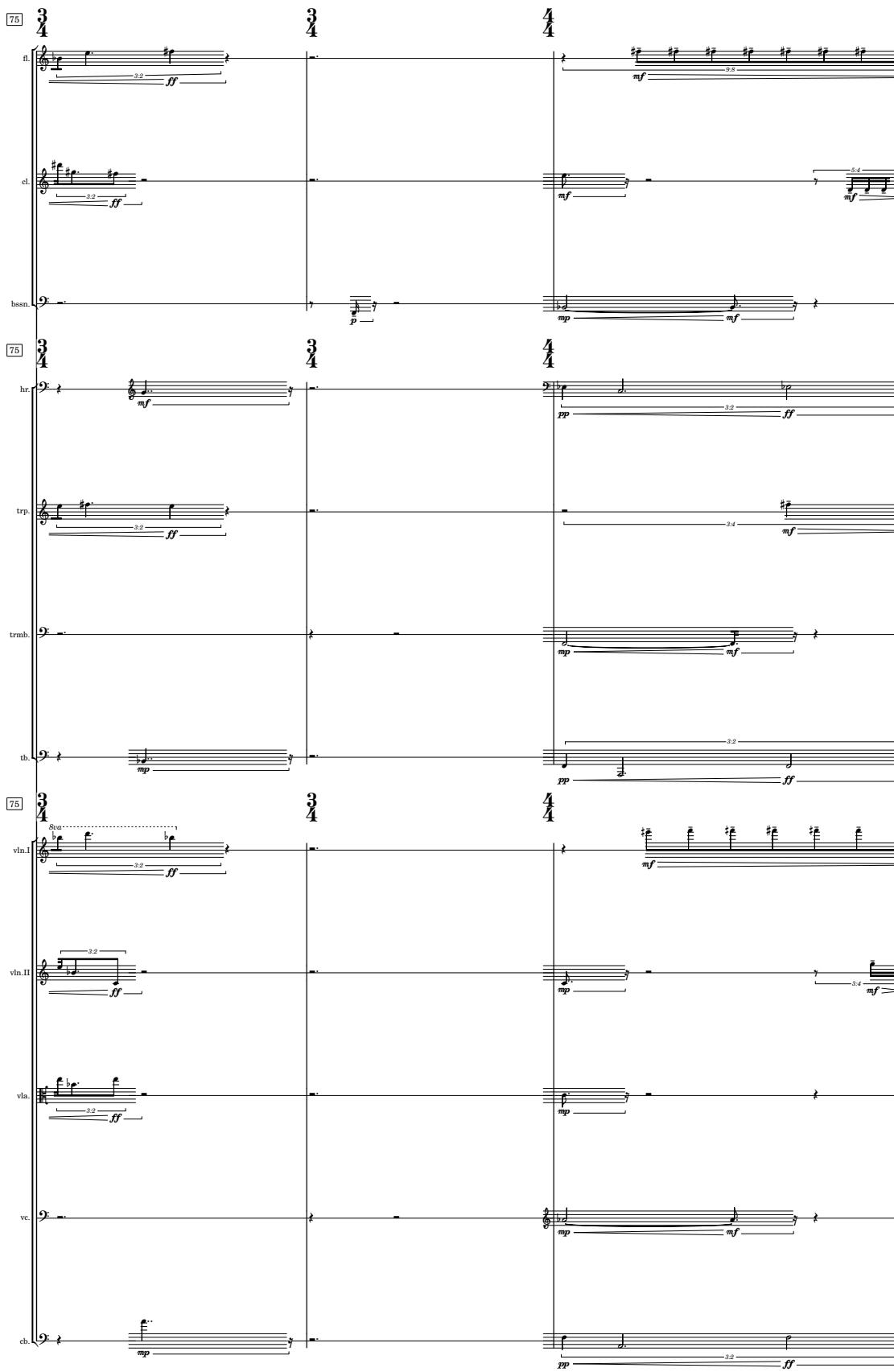
106 

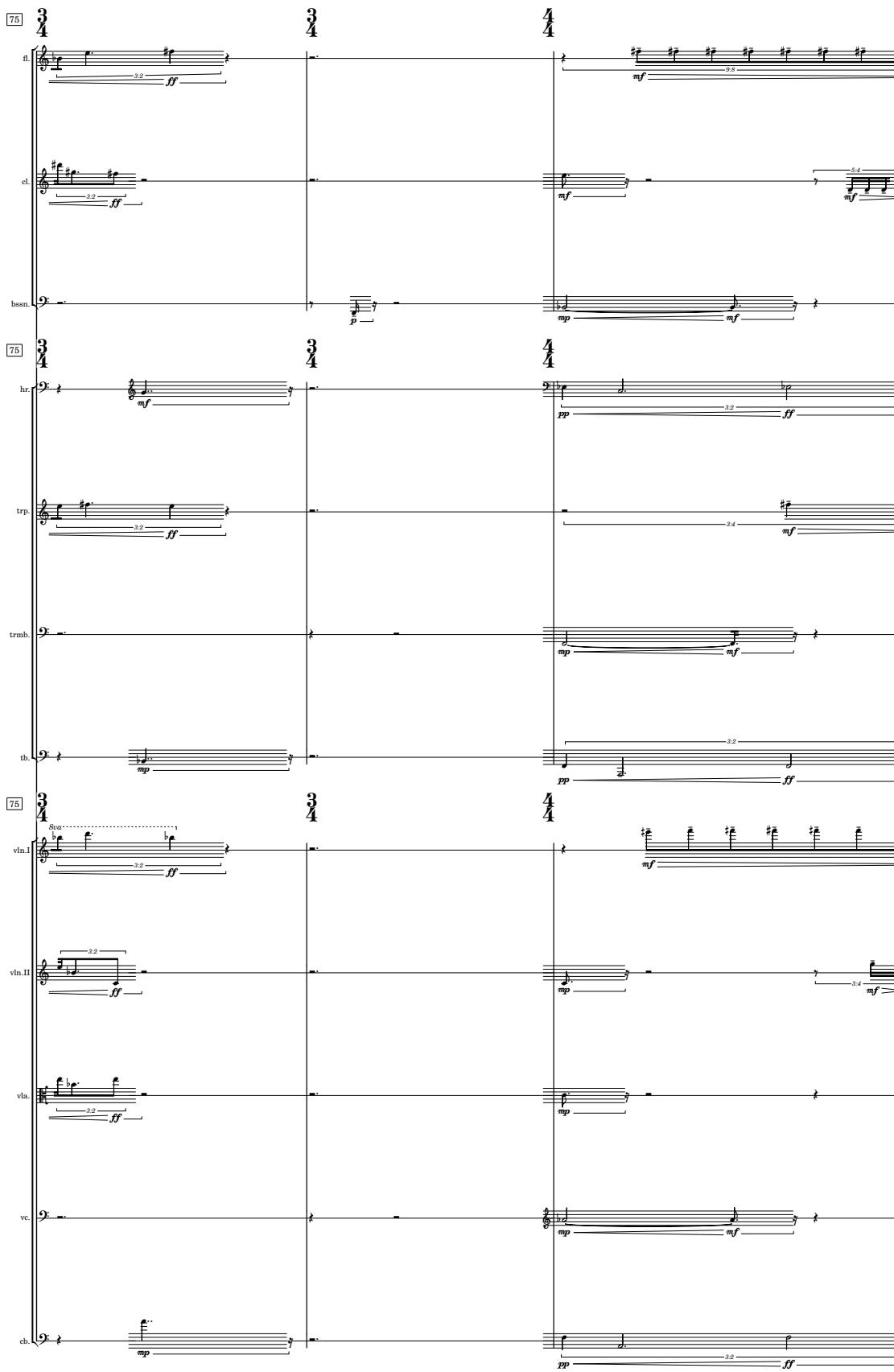
107 

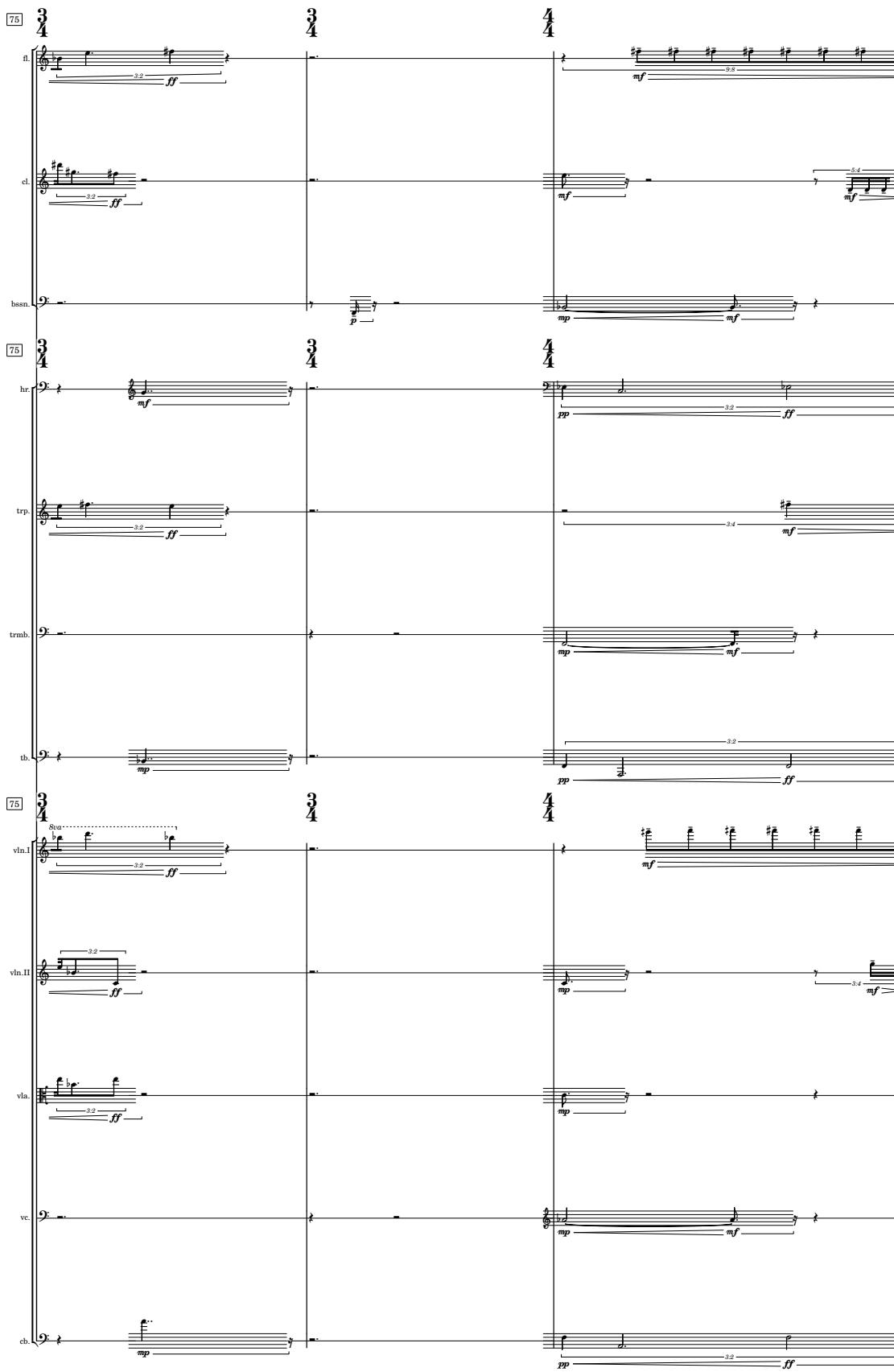
108 

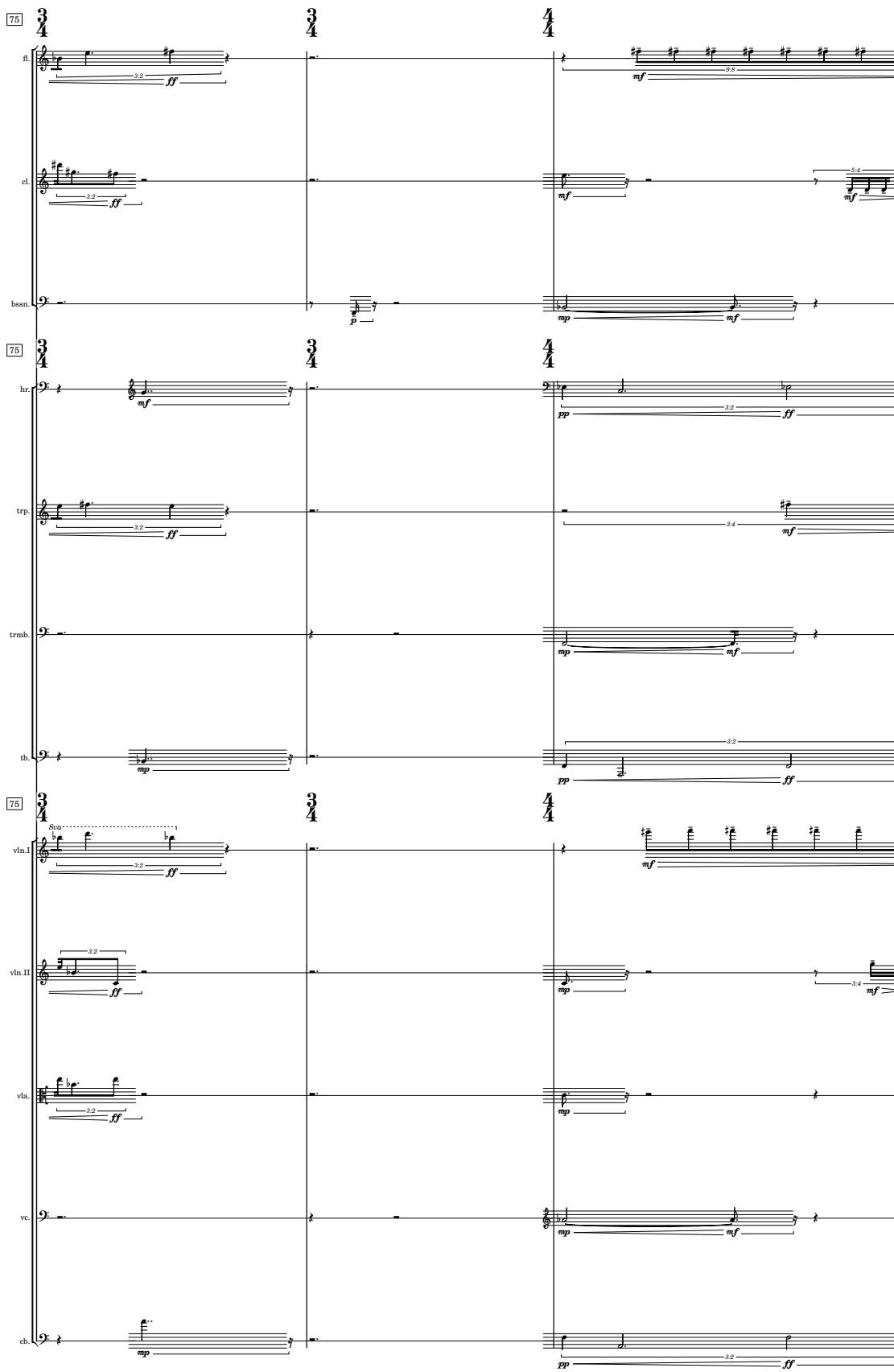
109 

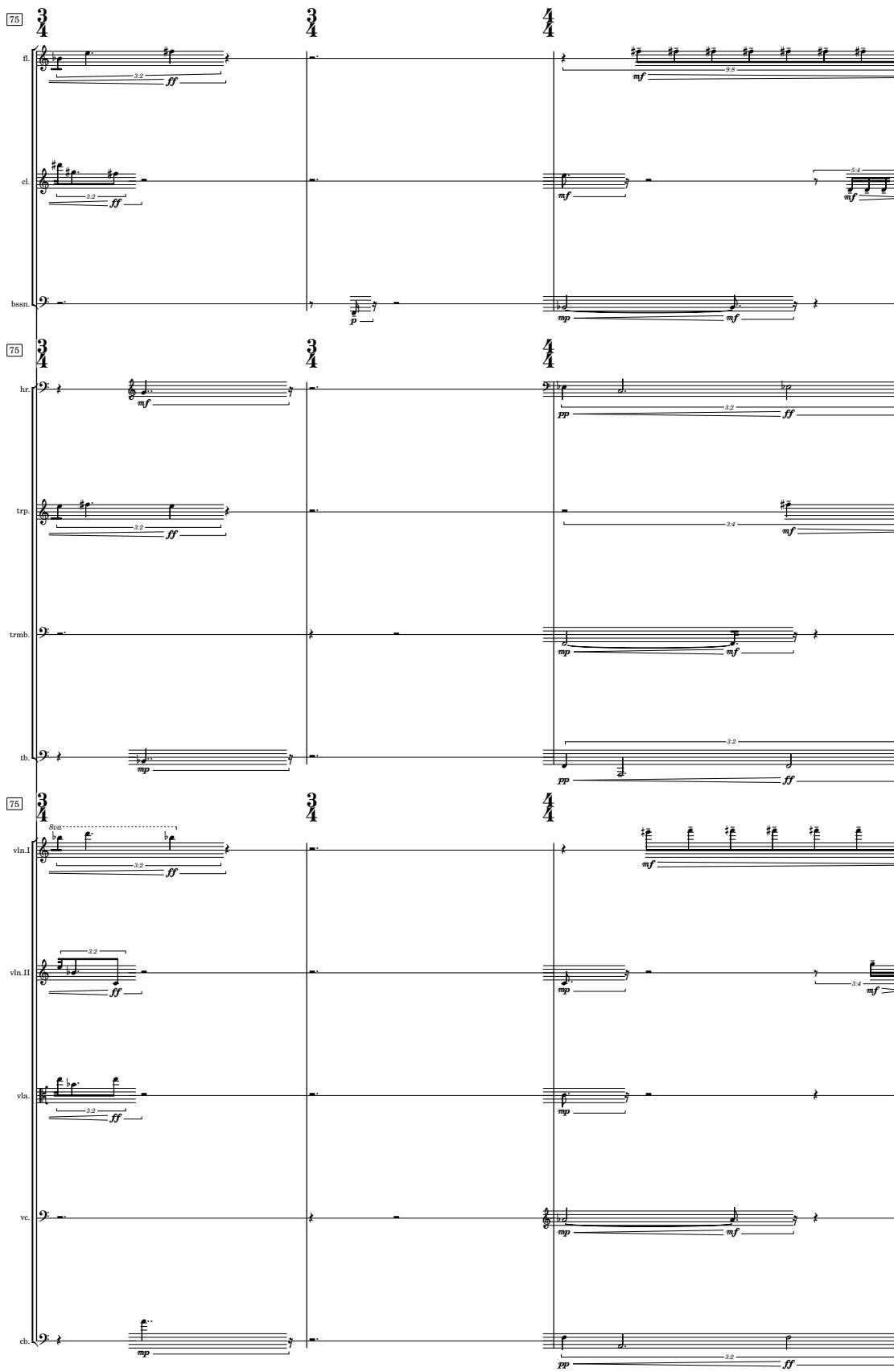
110 

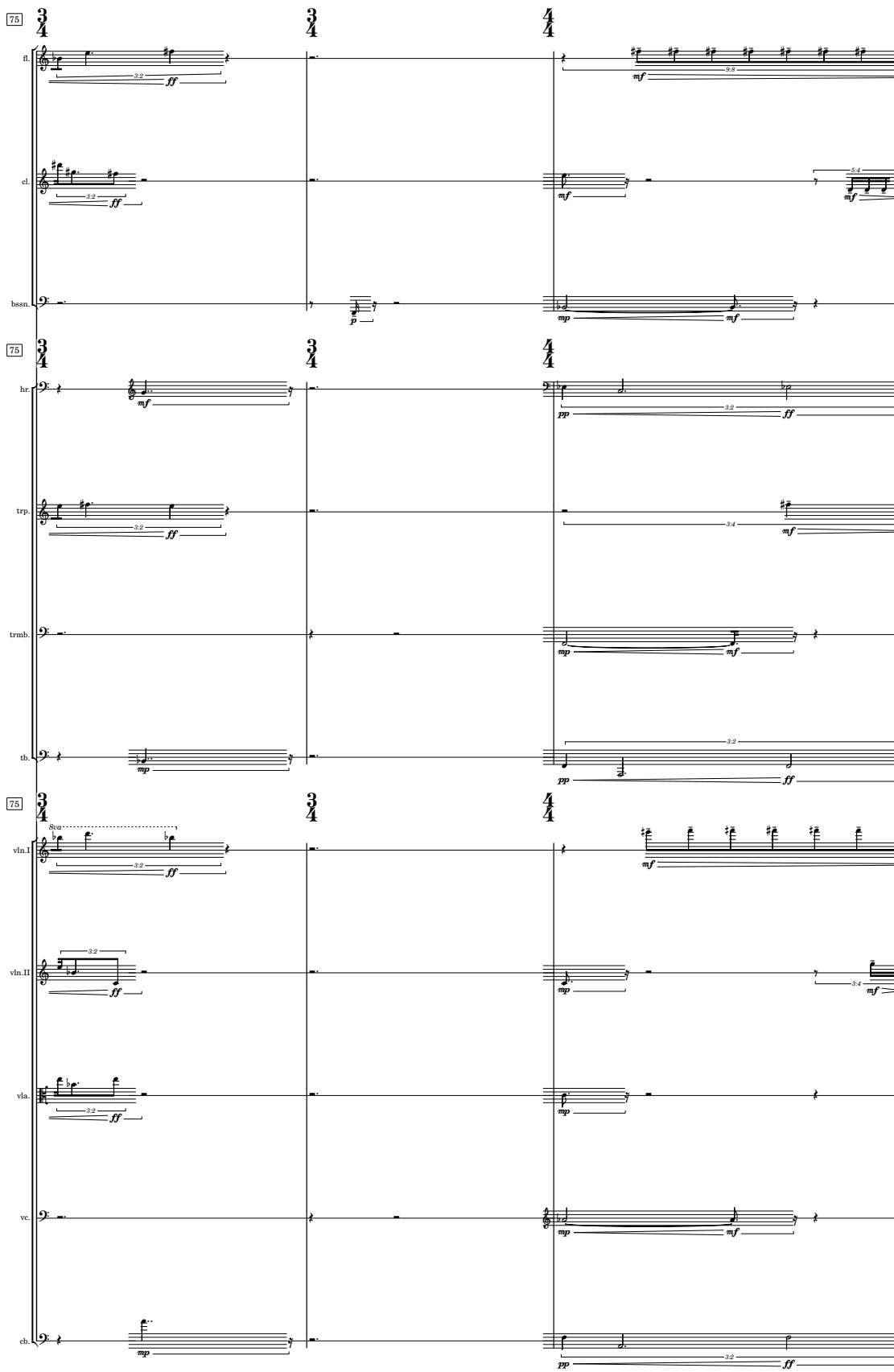
111 

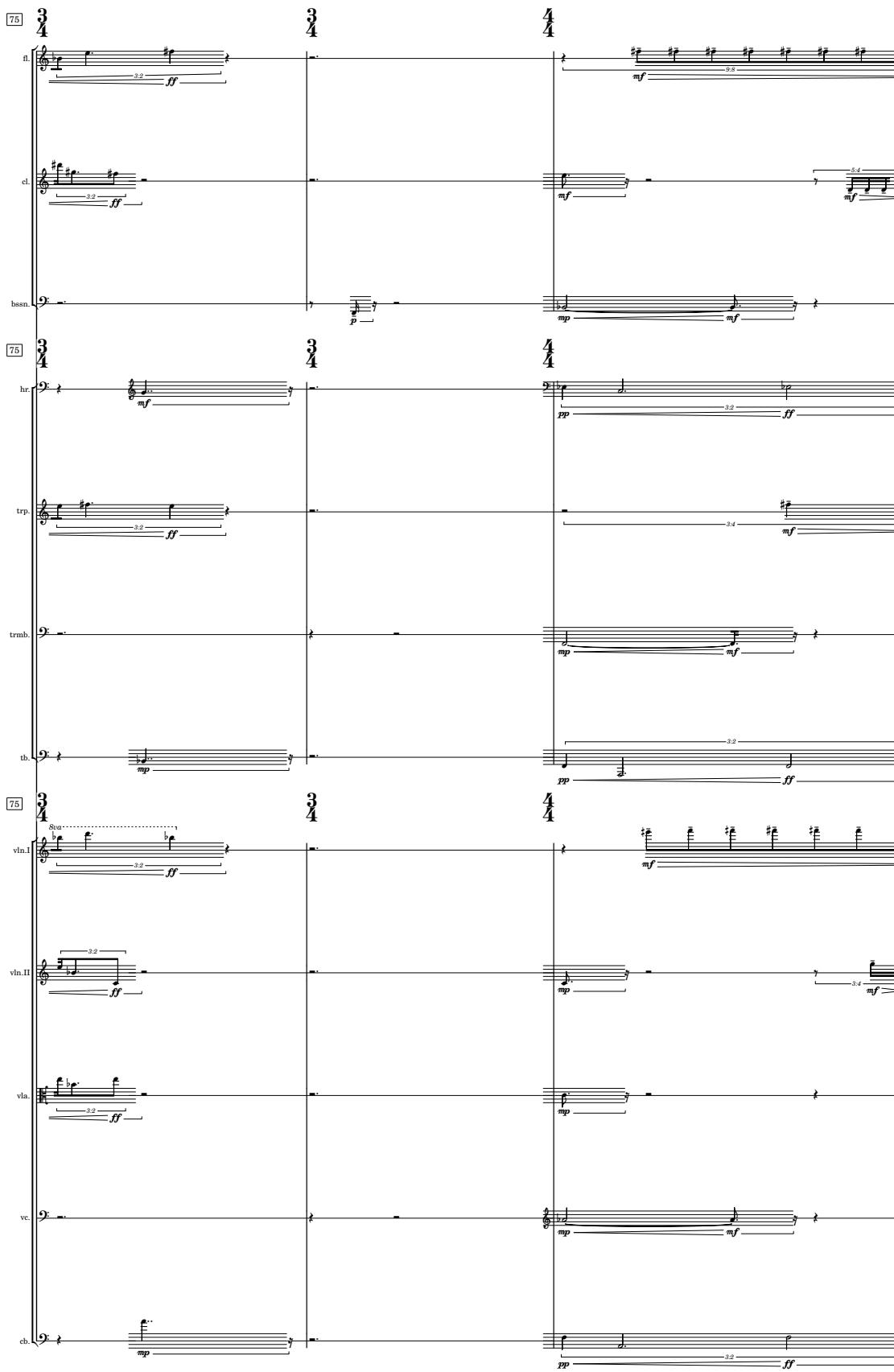
112 

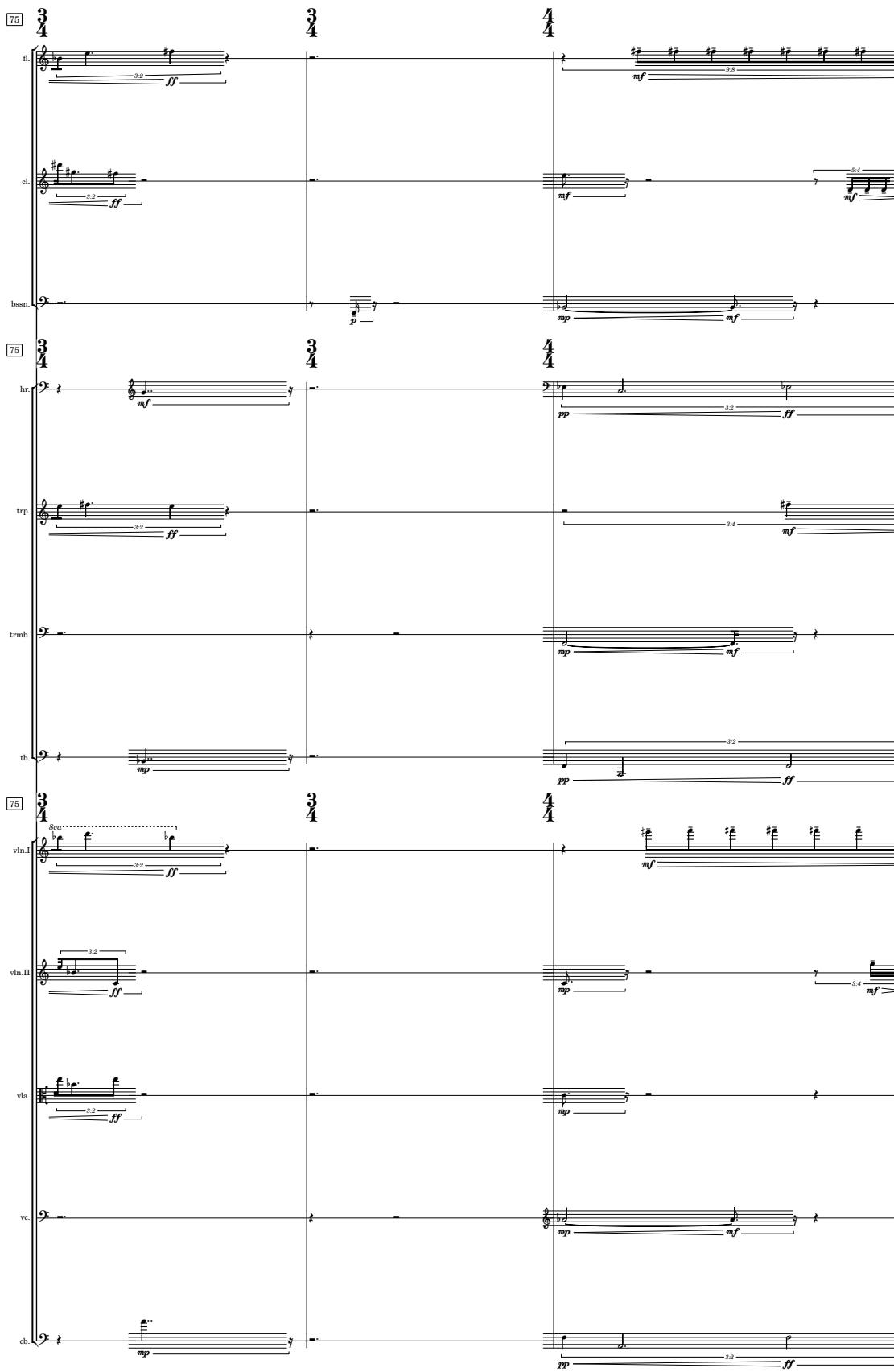
113 

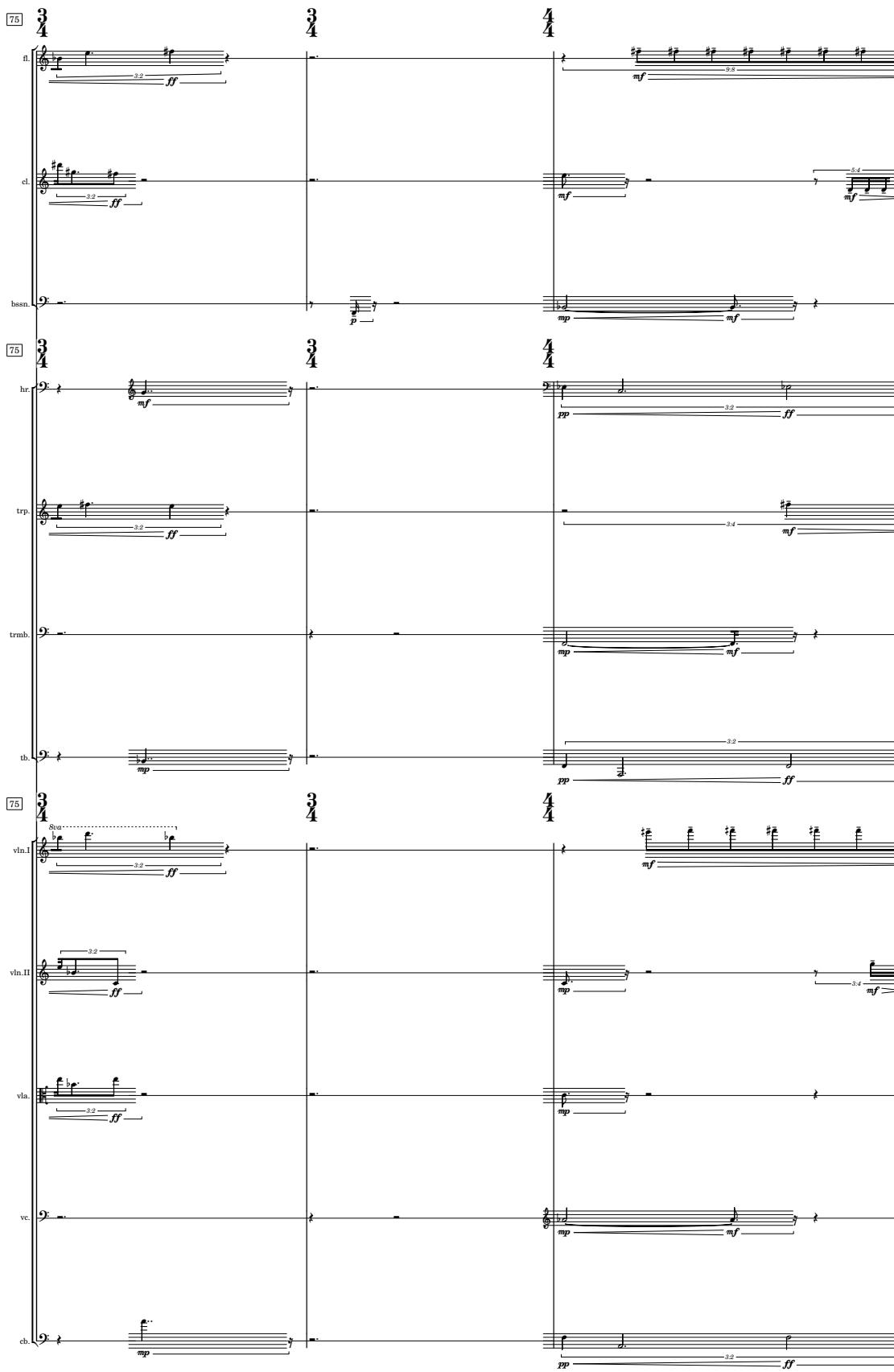
114 

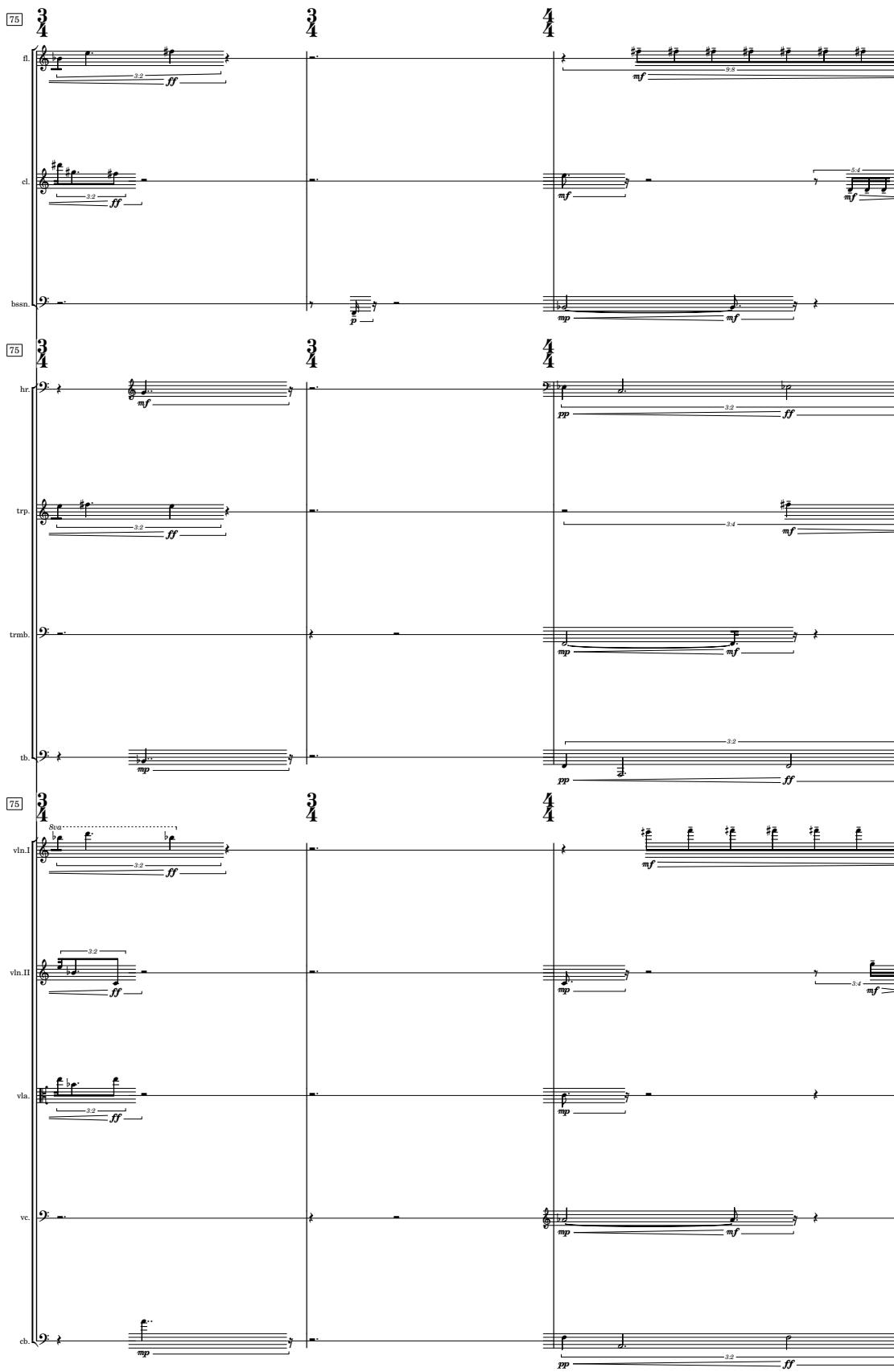
115 

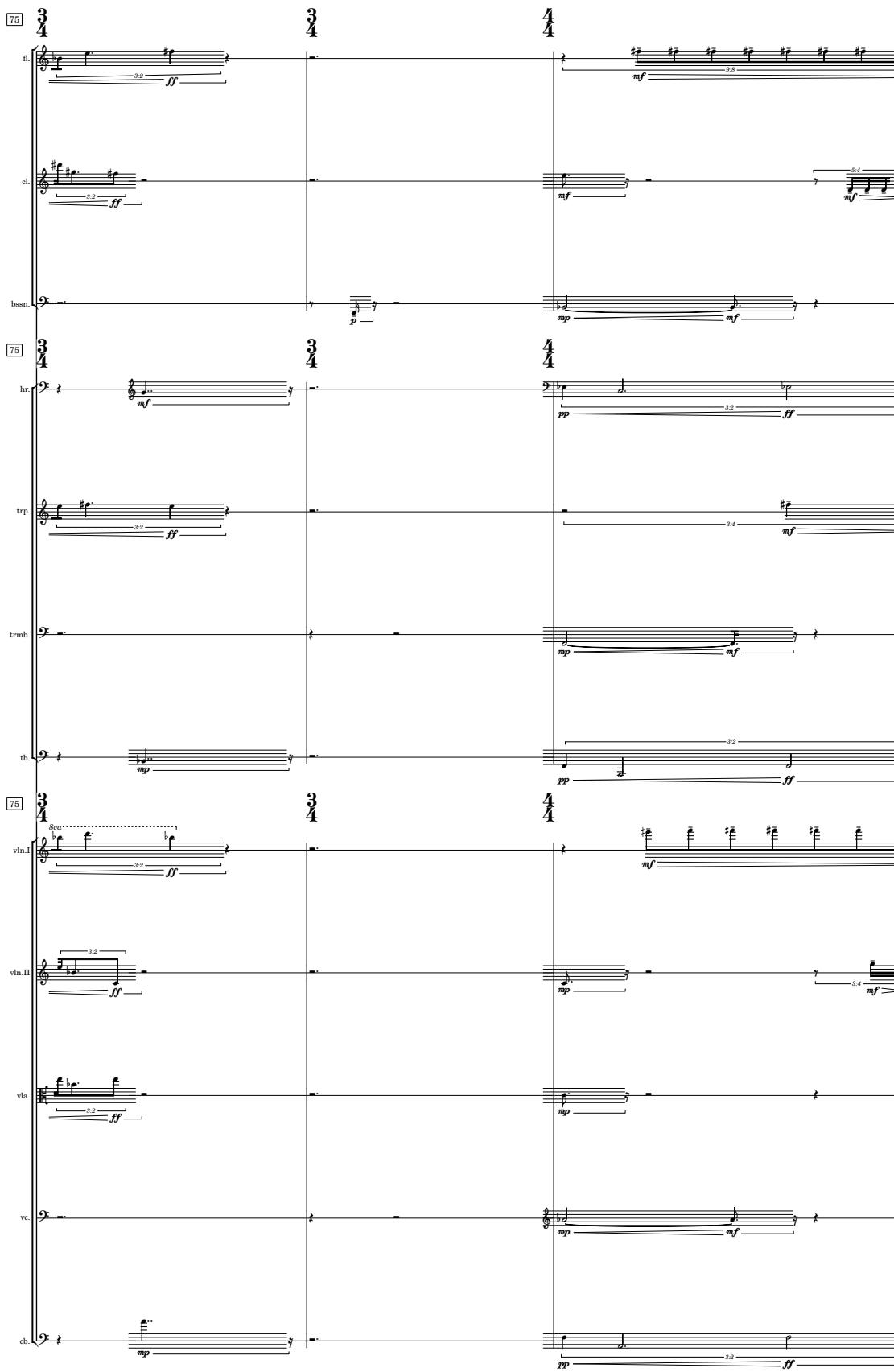
116 

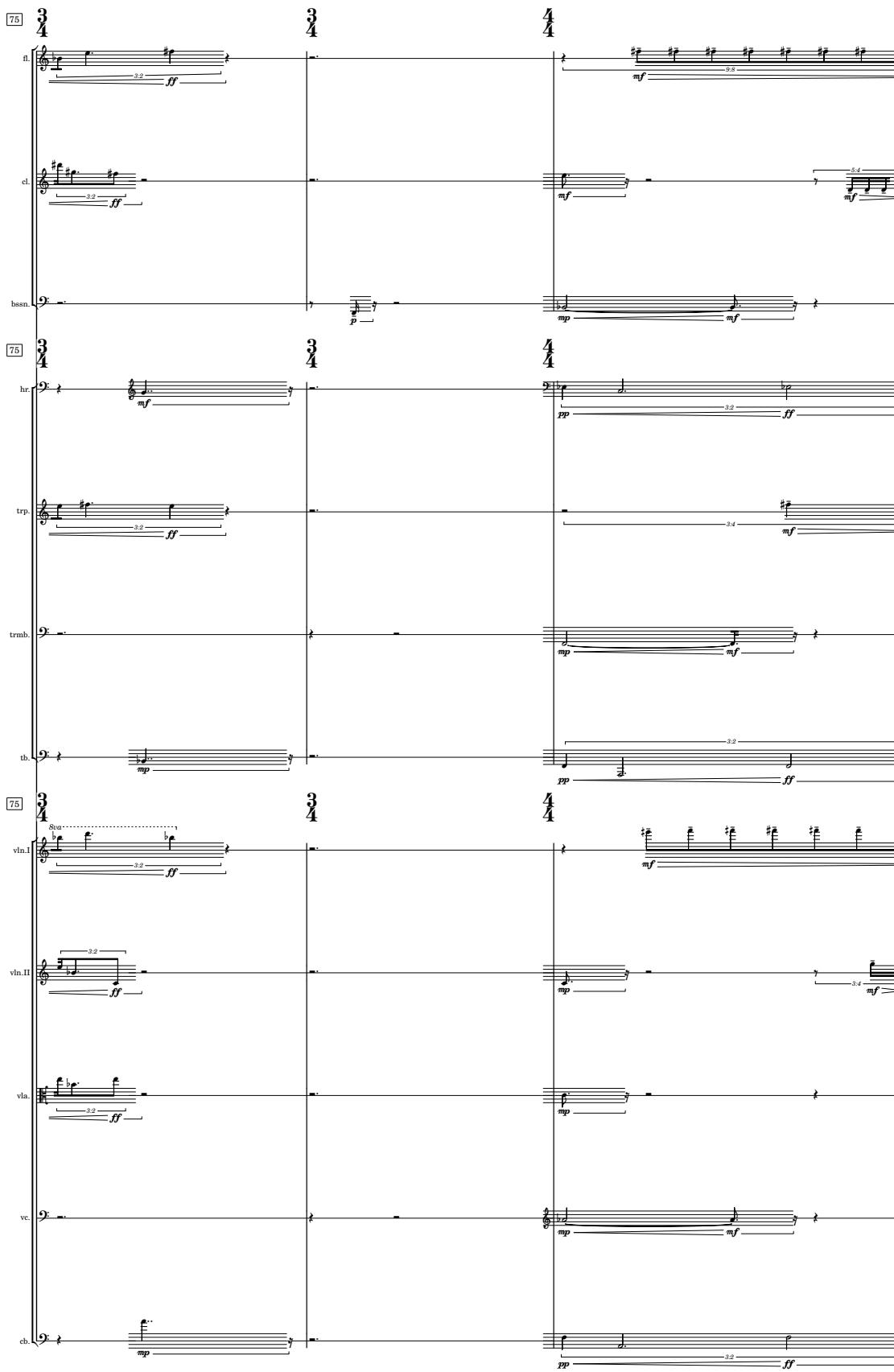
117 

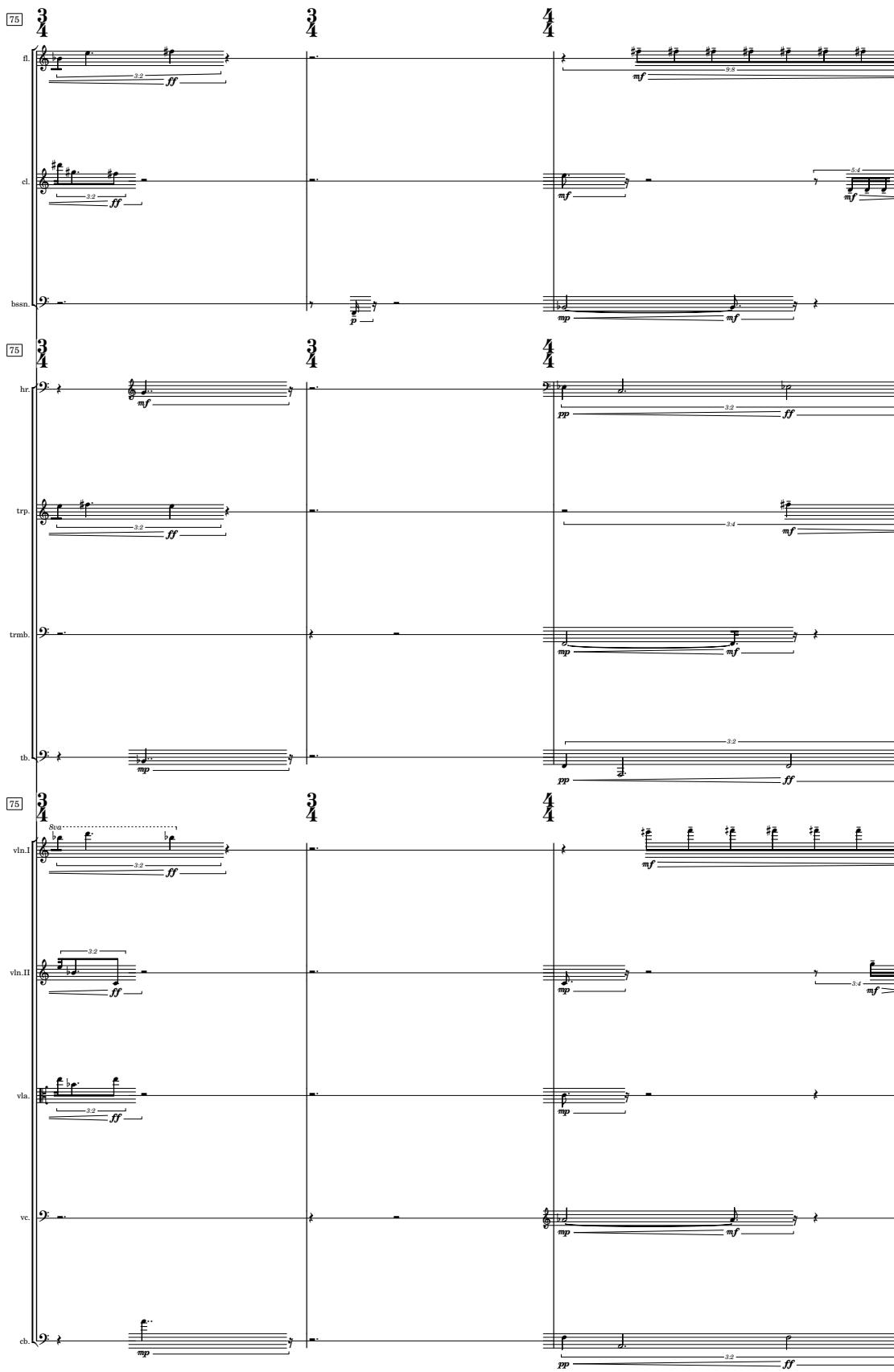
118 

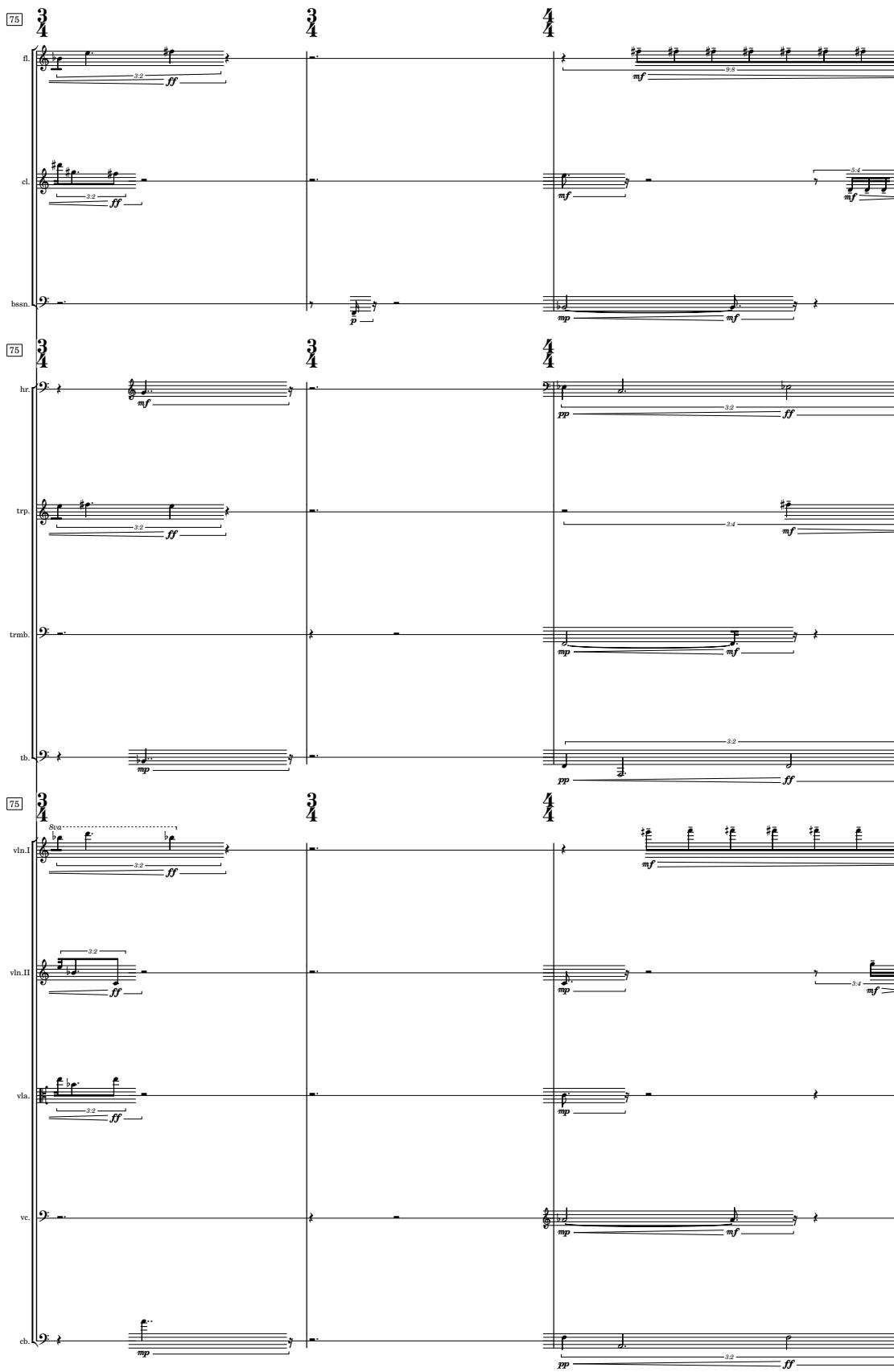
119 

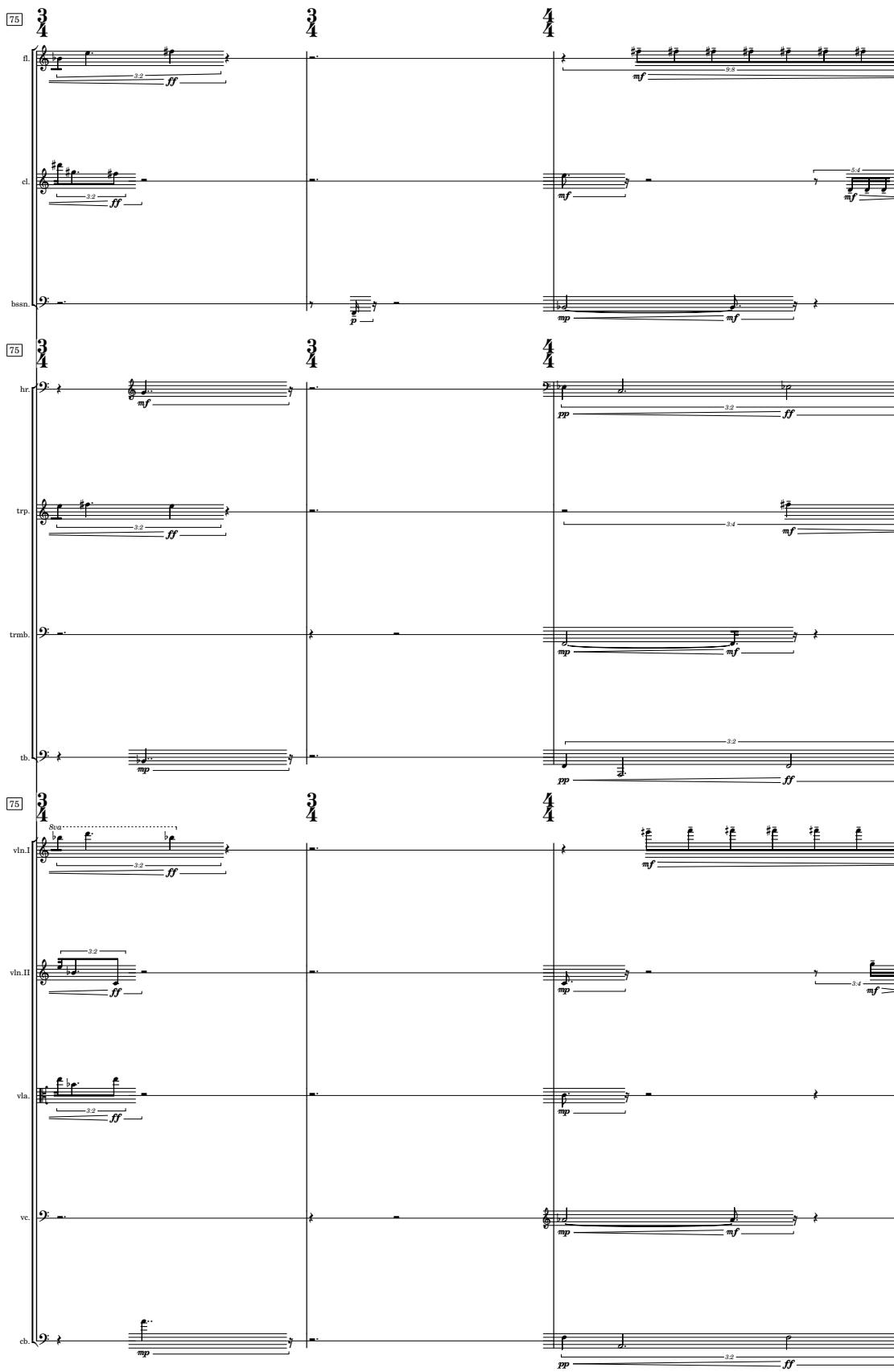
120 

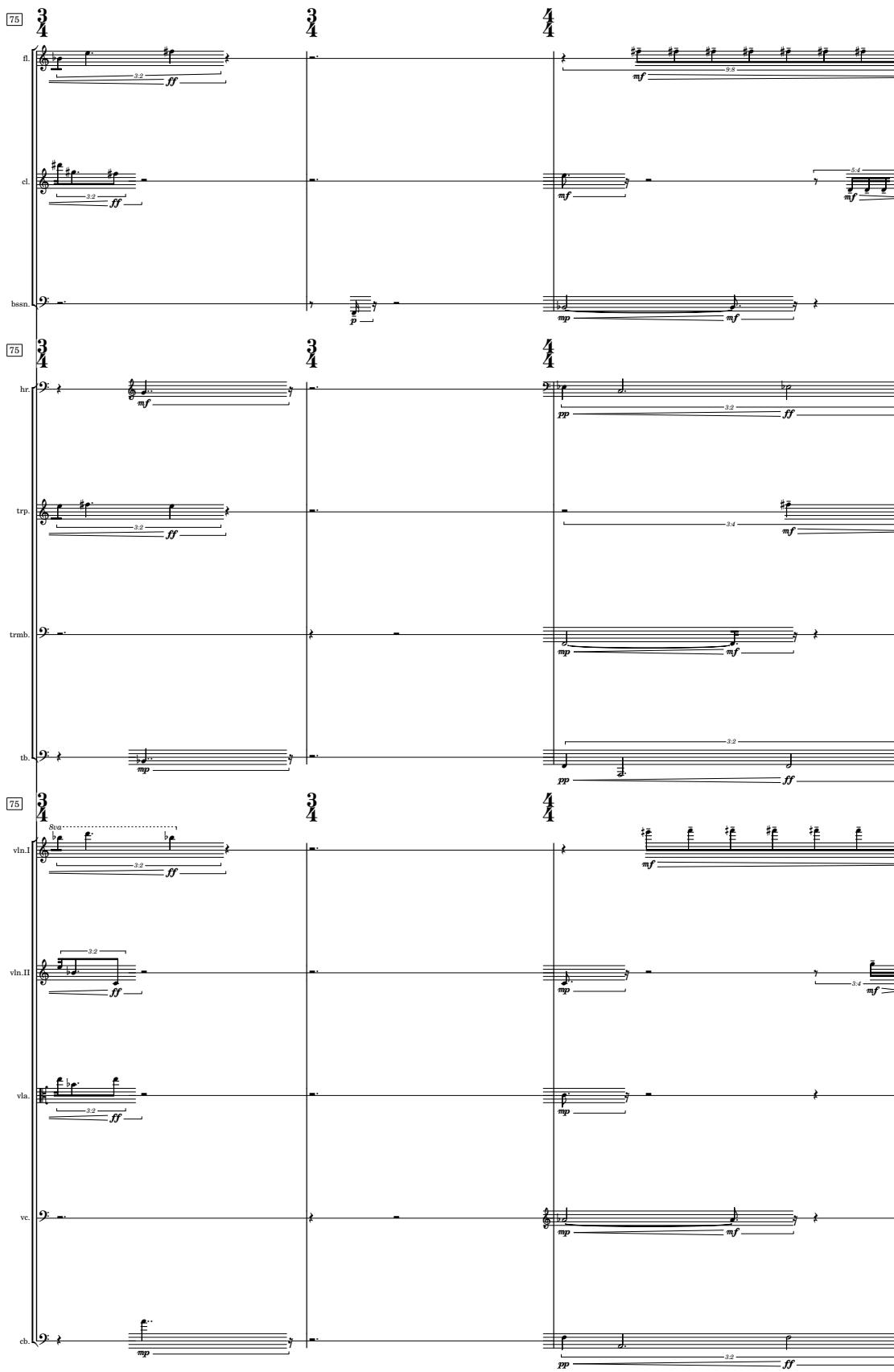
121 

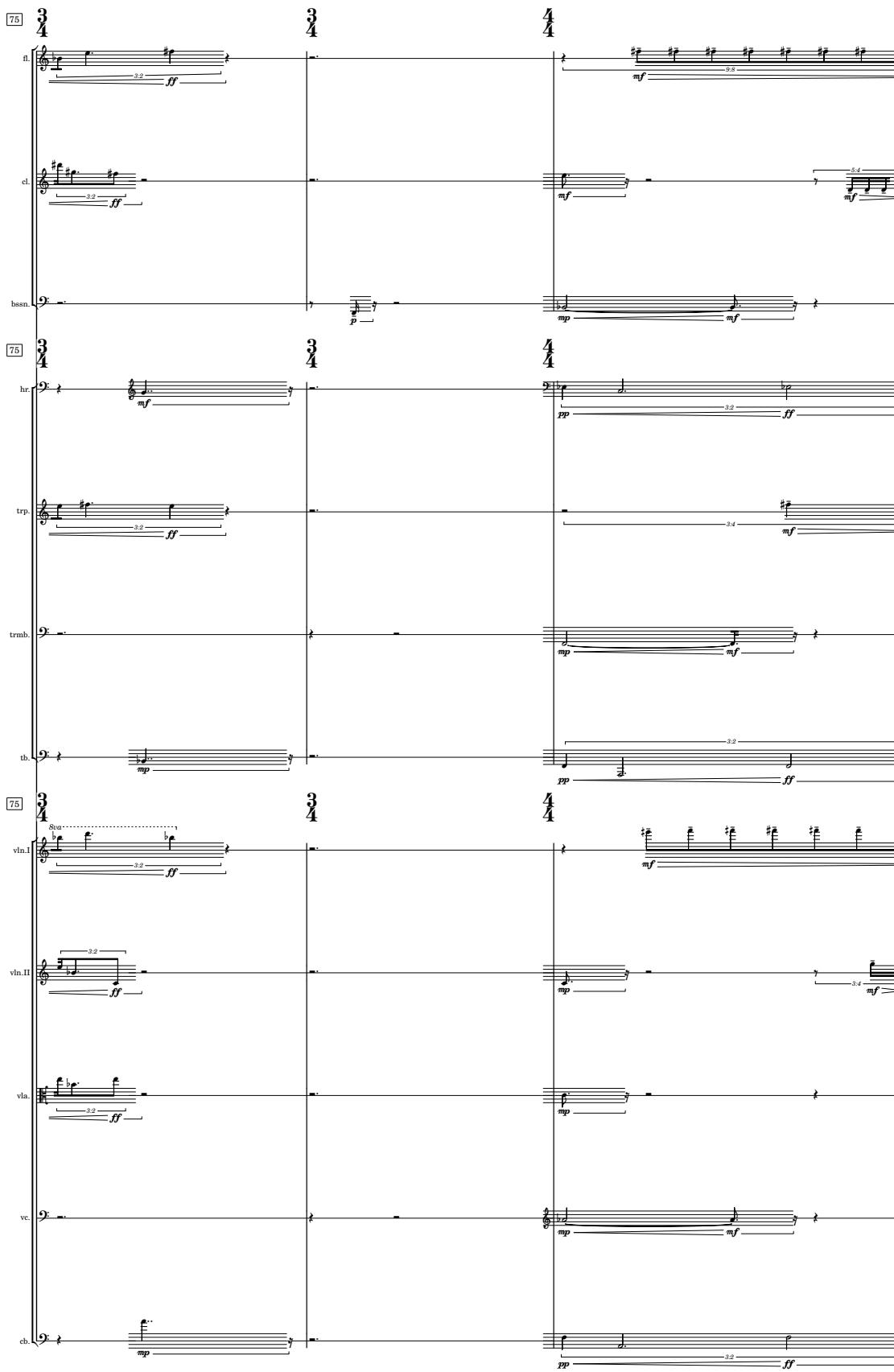
122 

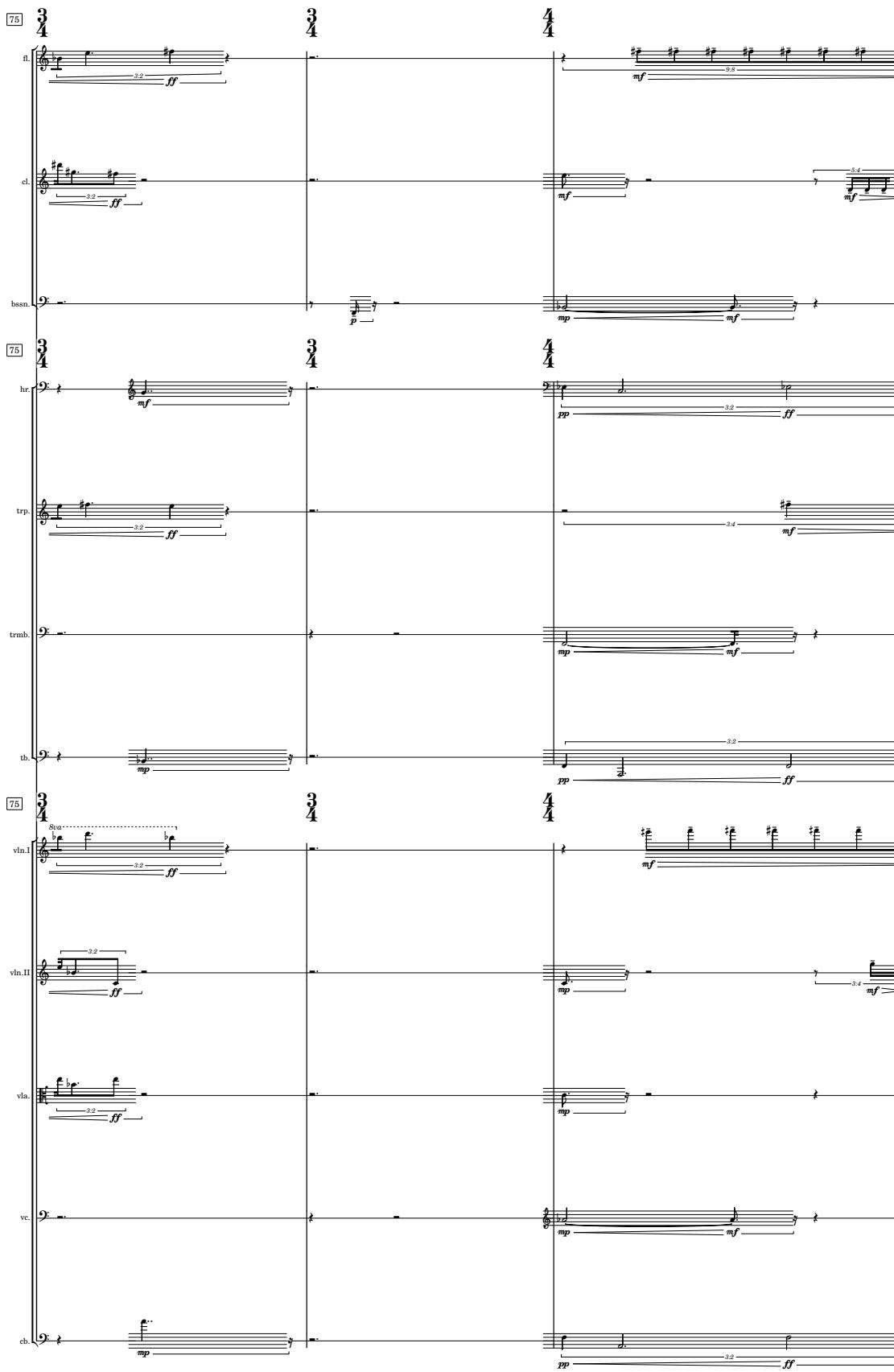
123 

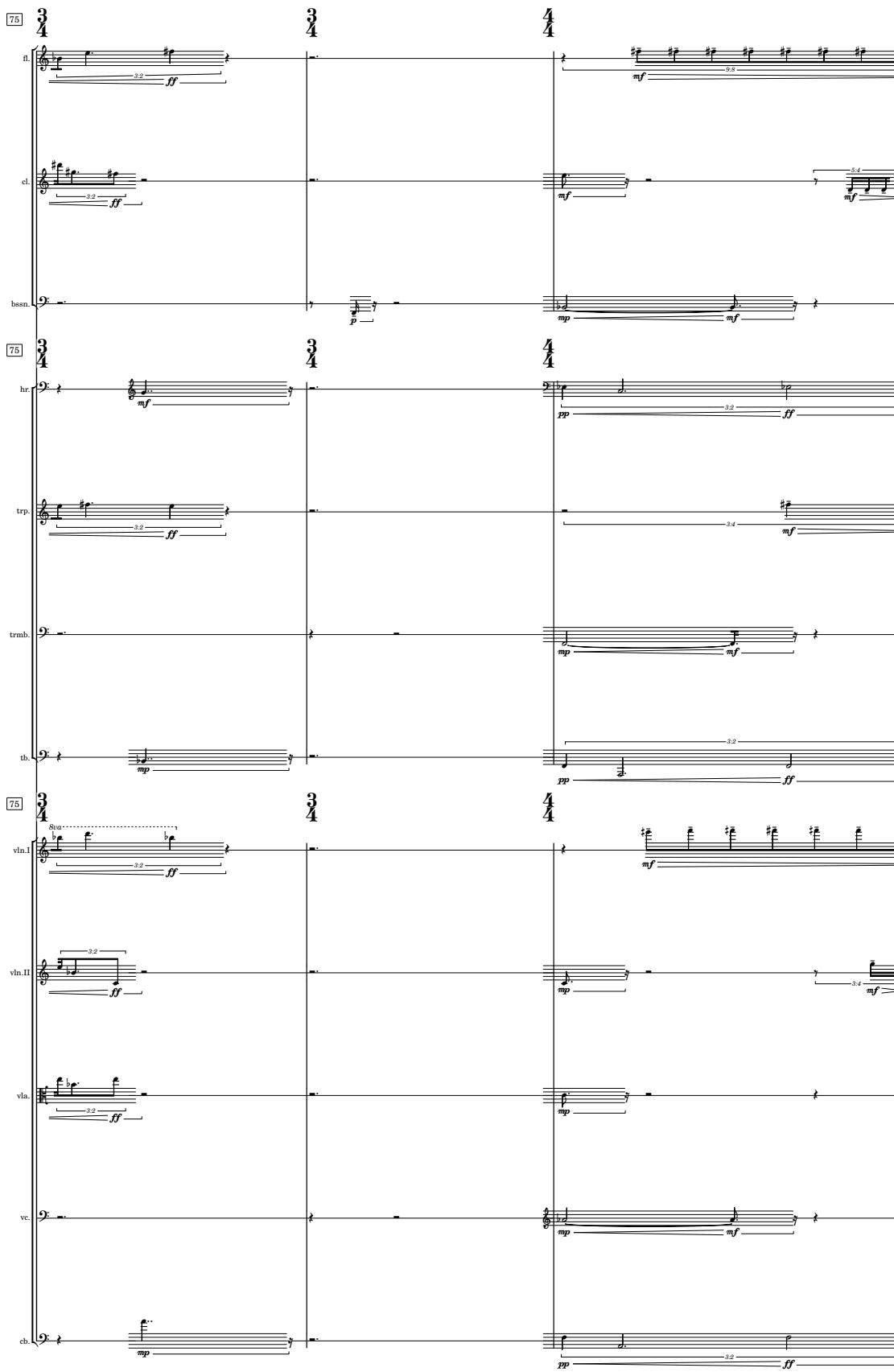
124 

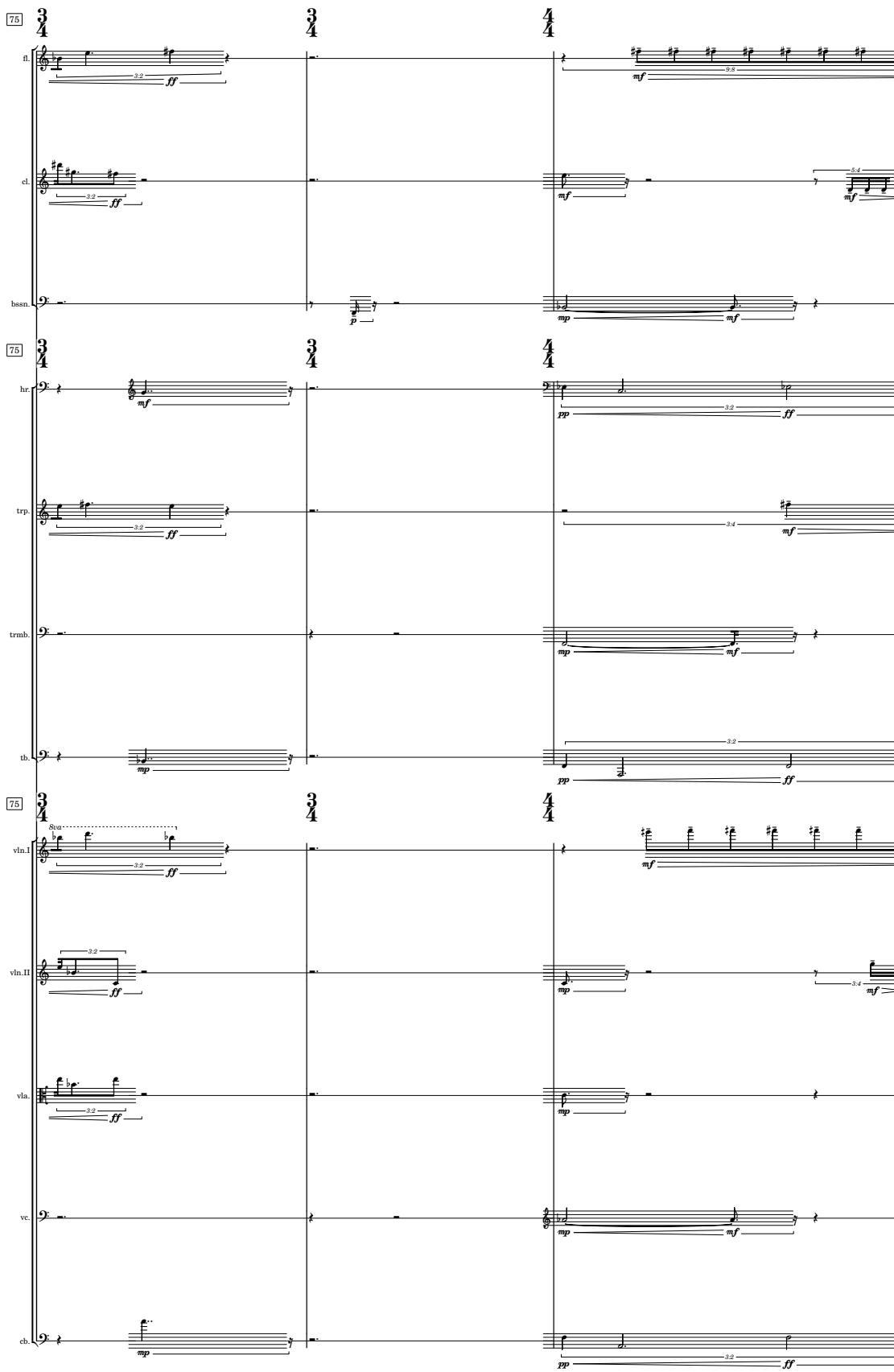
125 

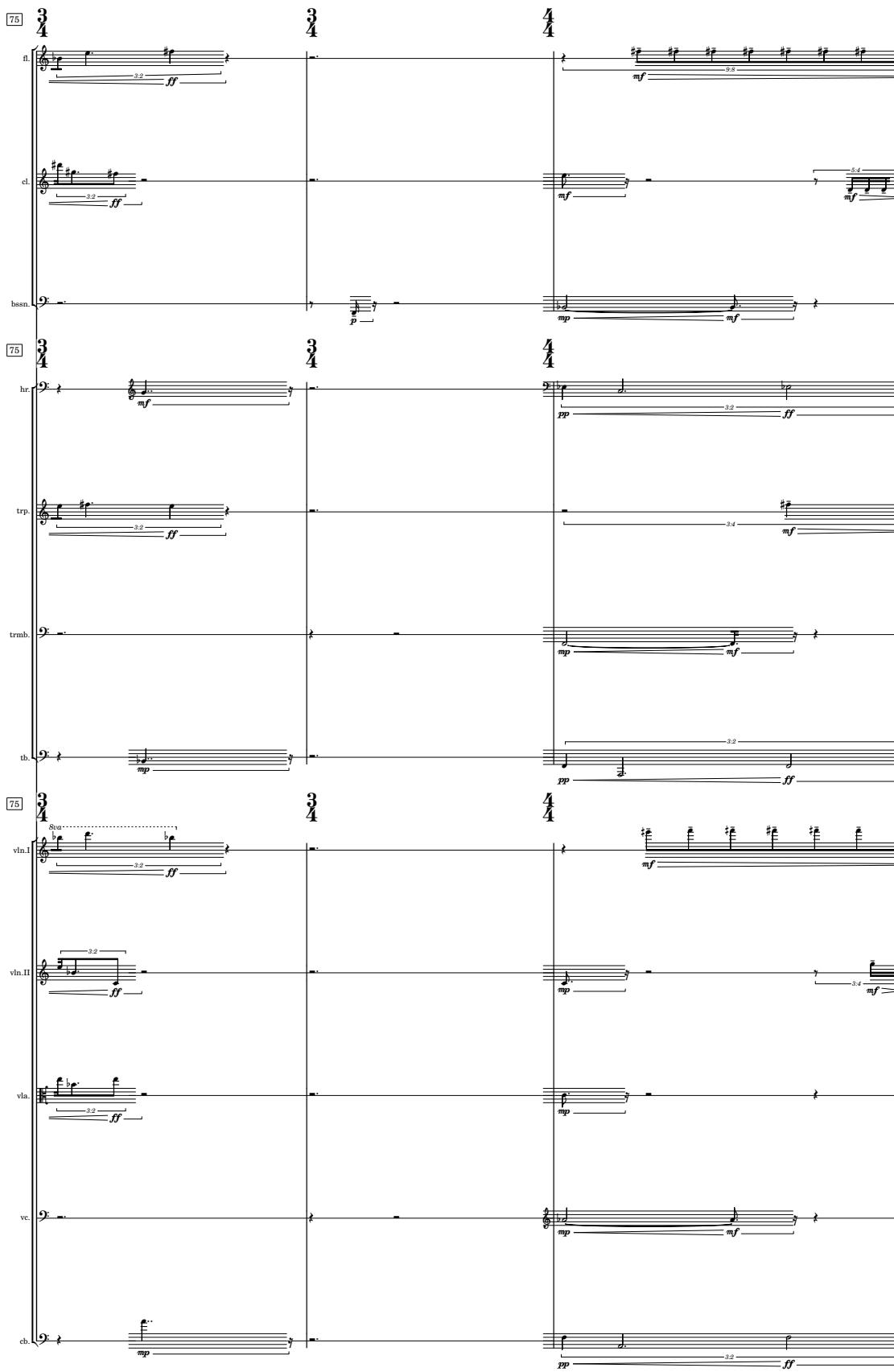
126 

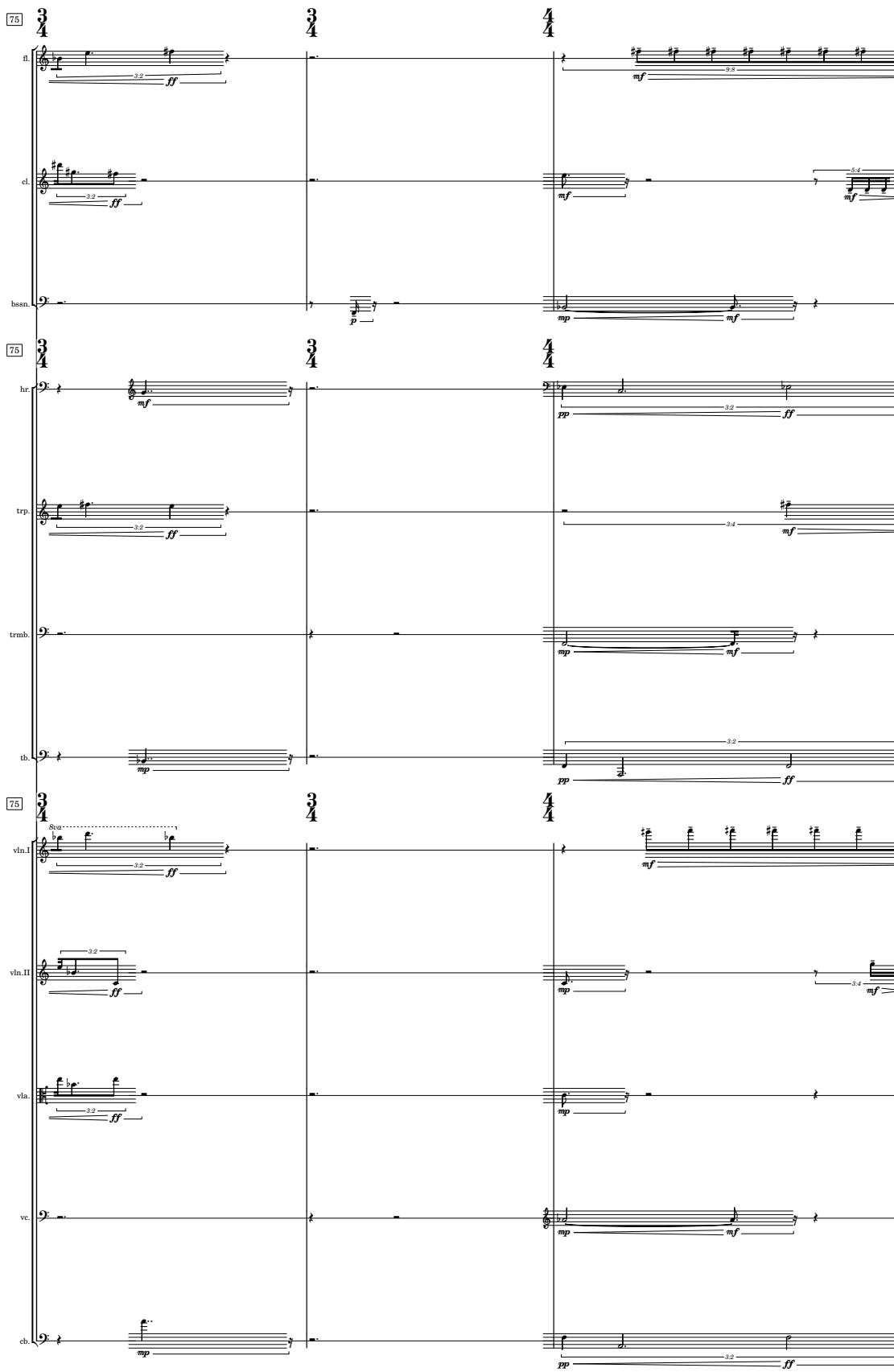
127 

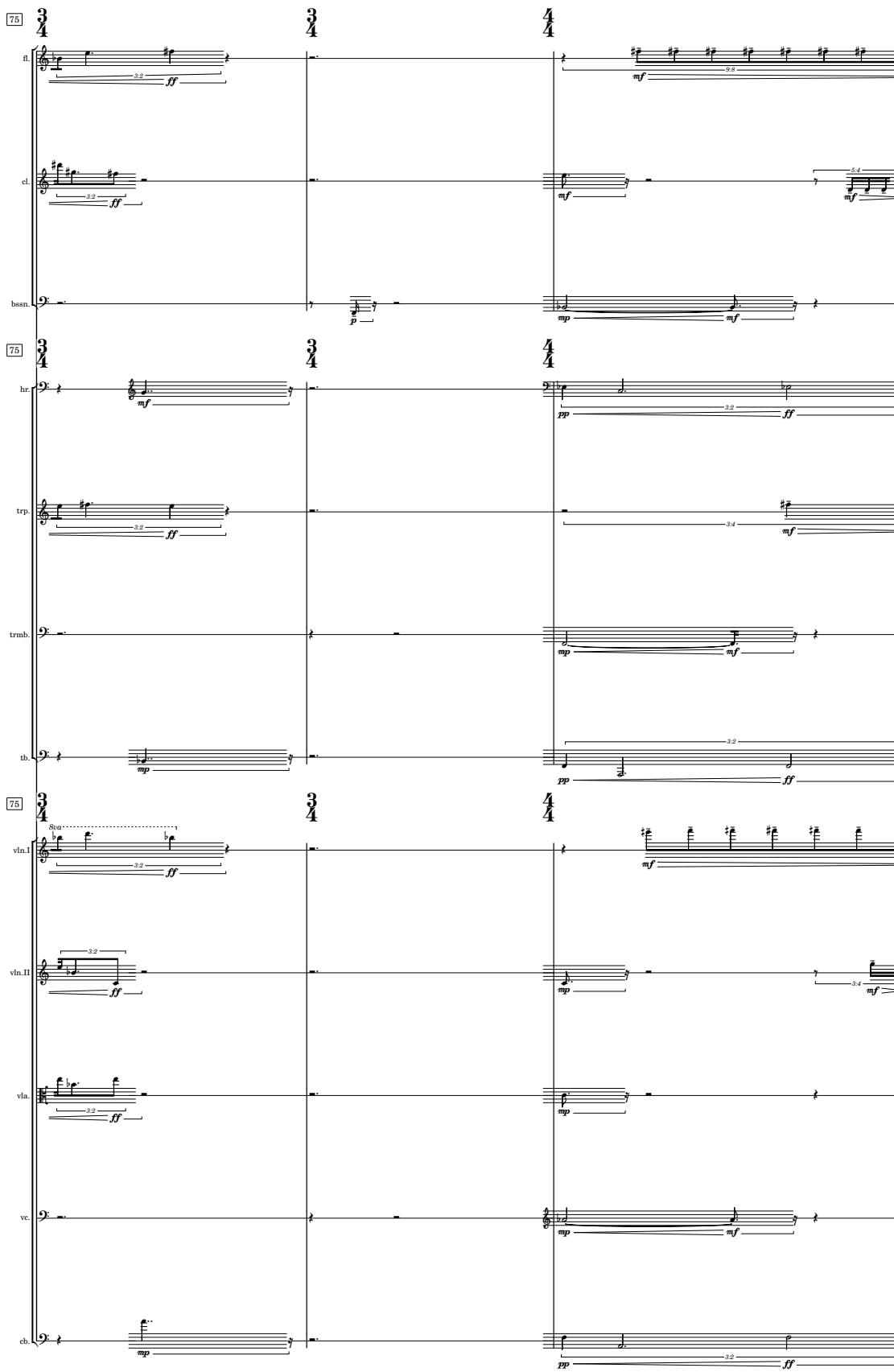
128 

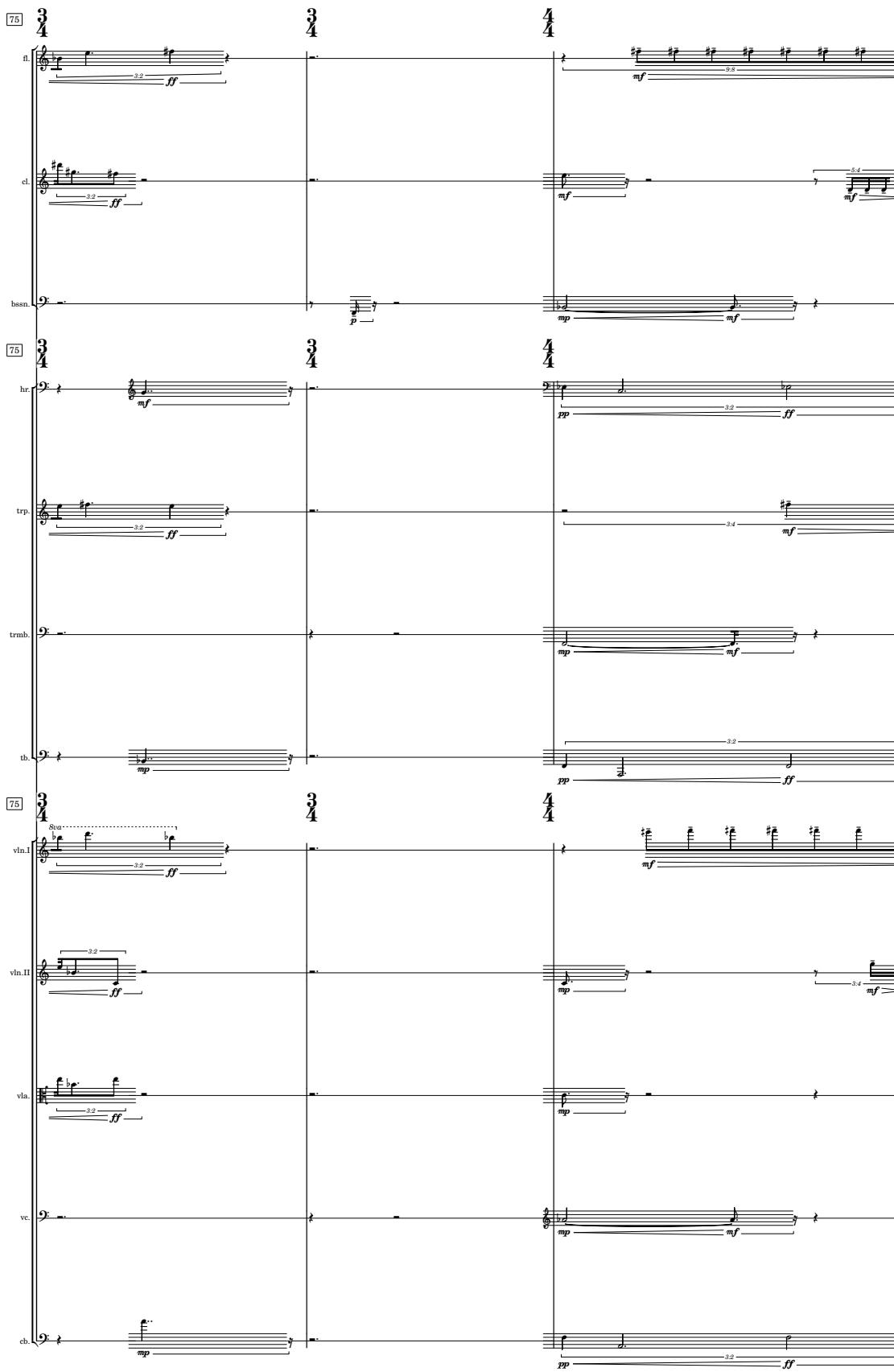
129 

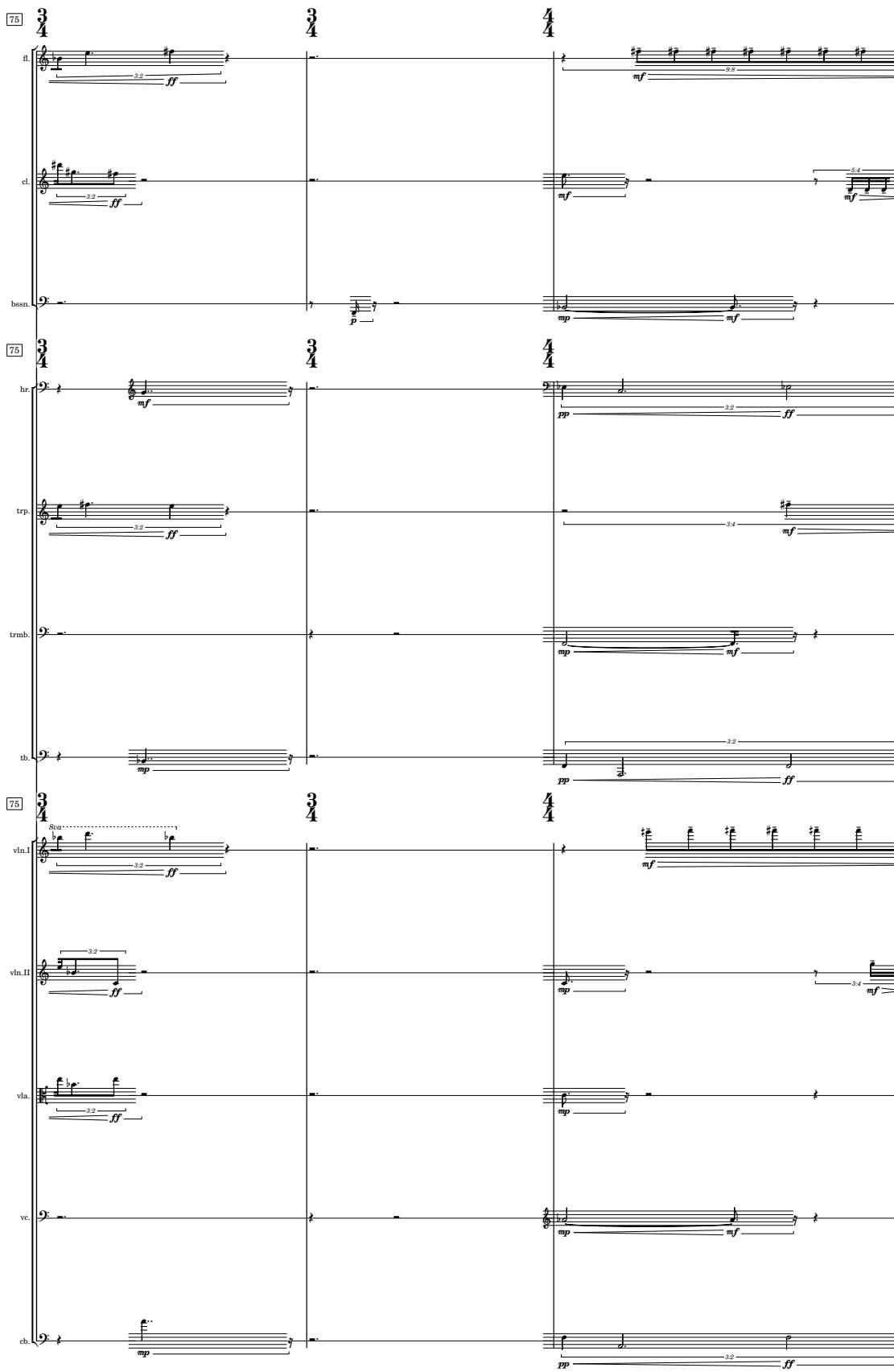
130 

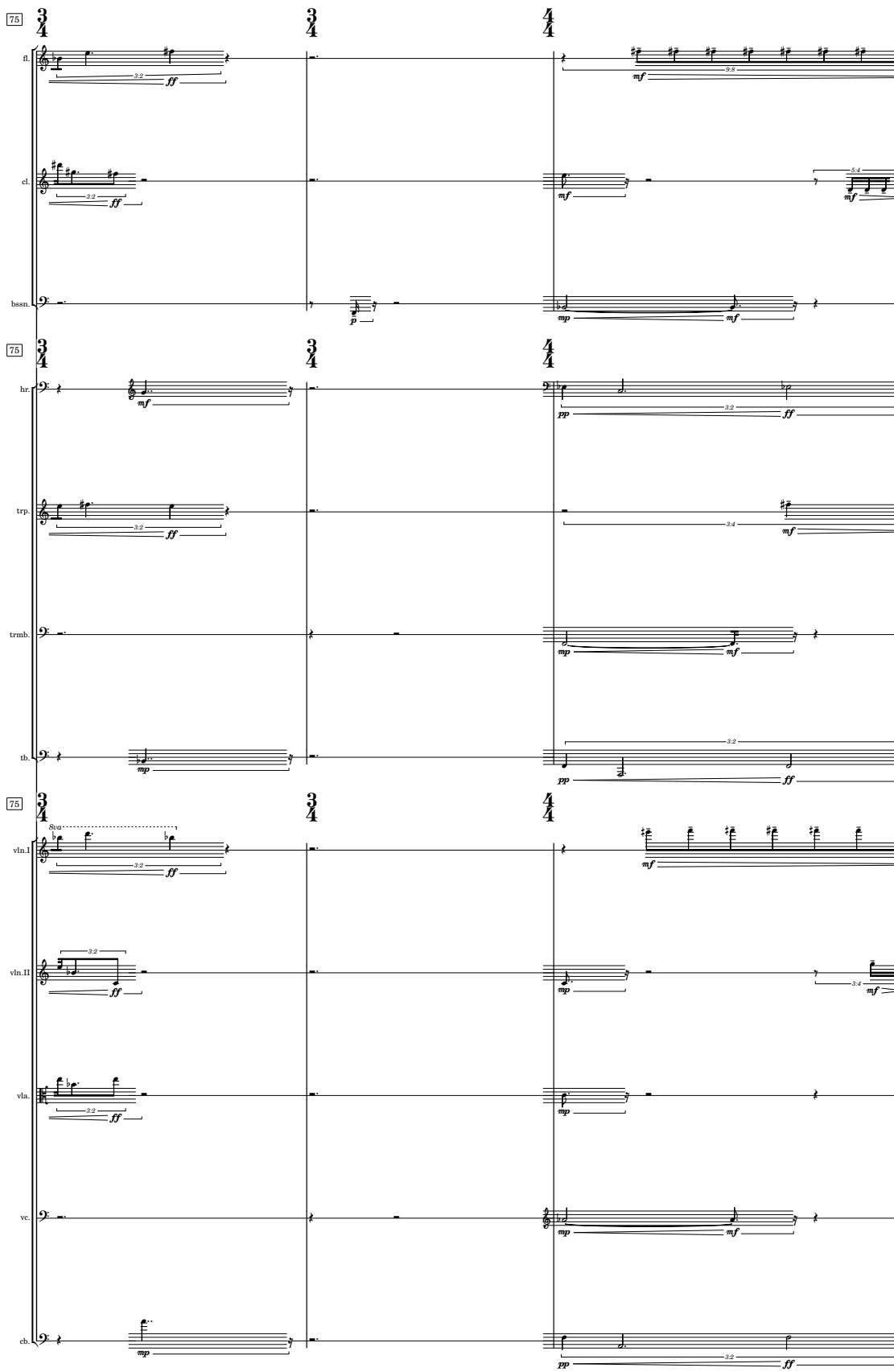
131 

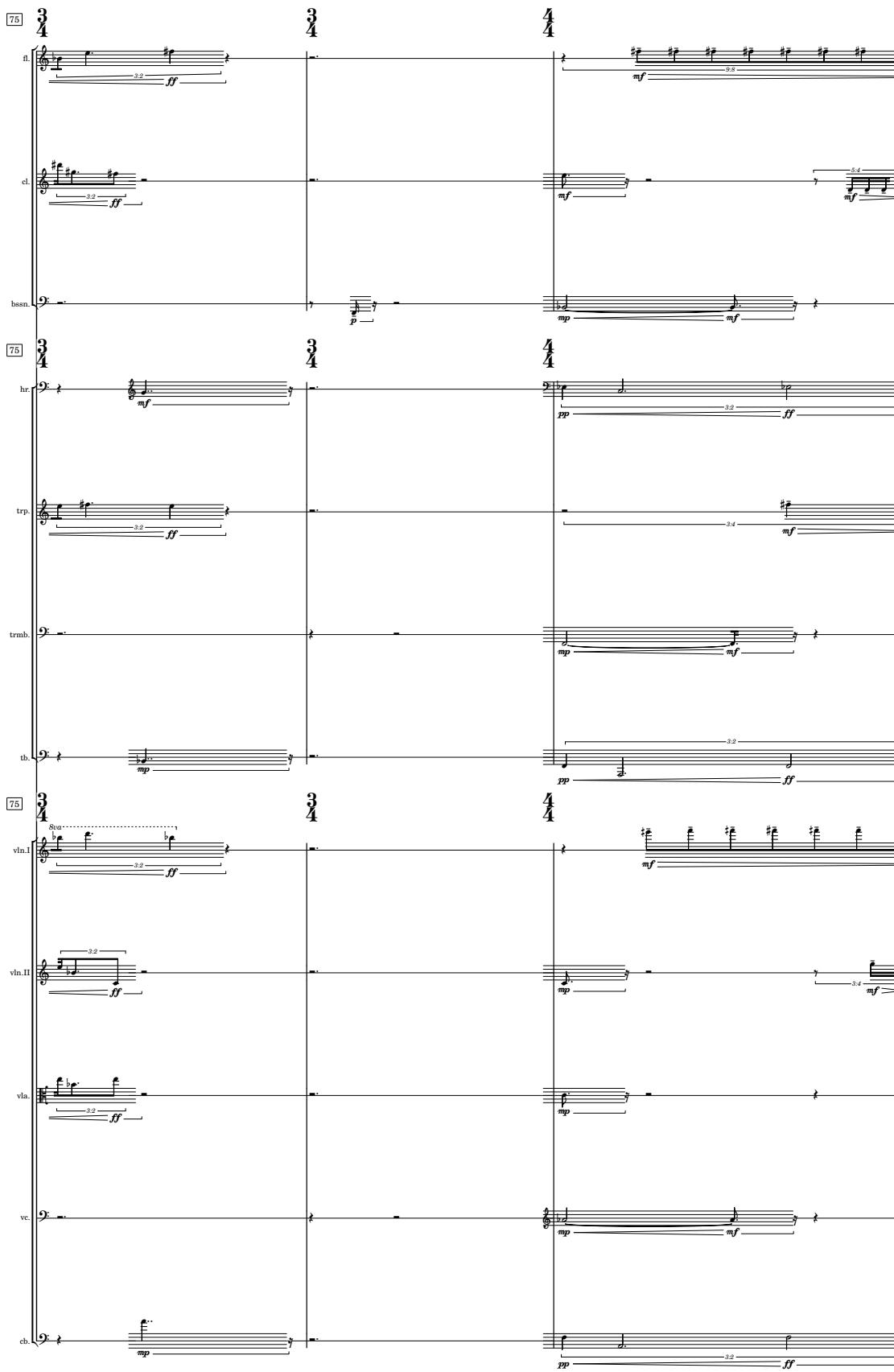
132 

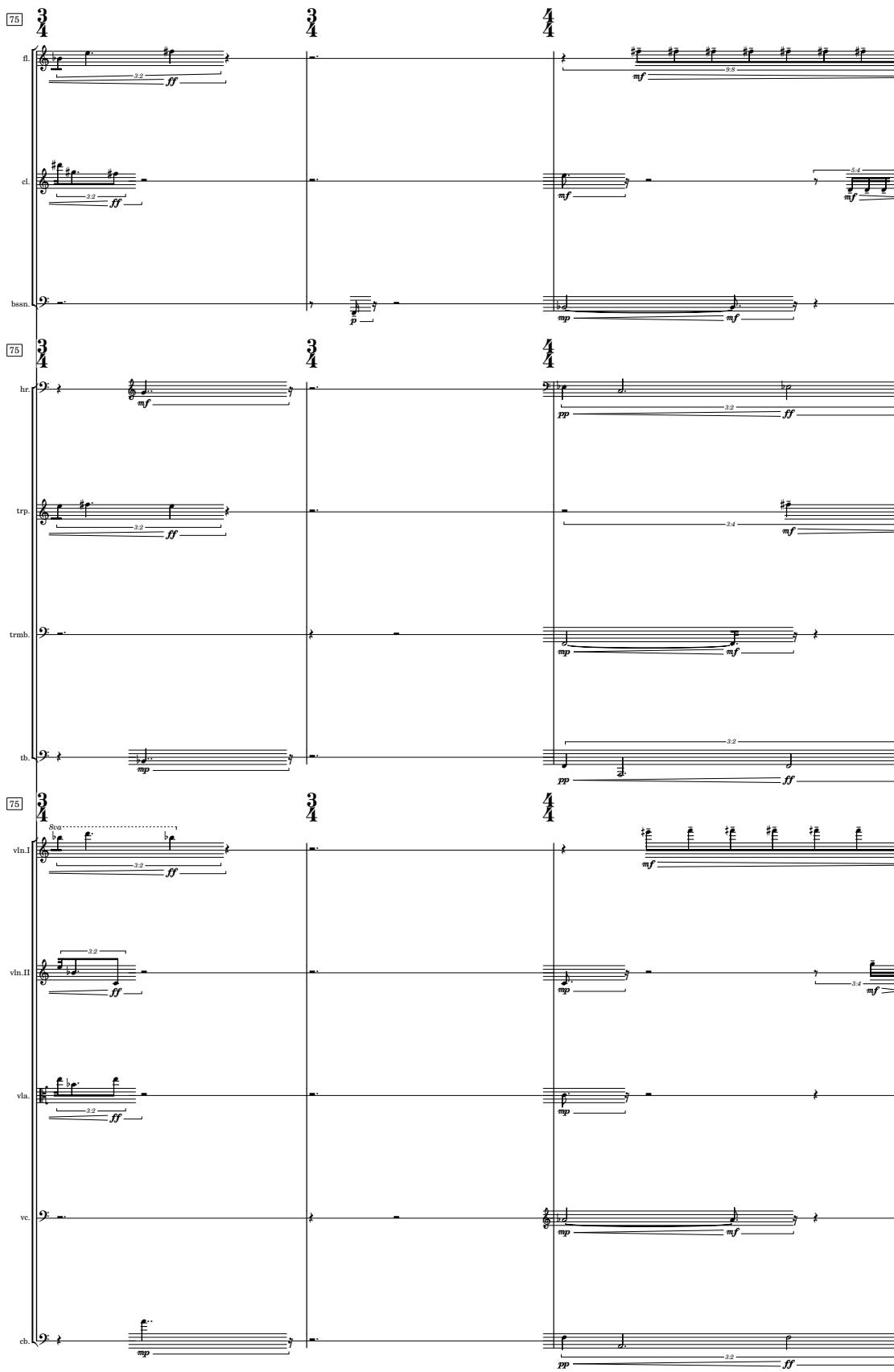
133 

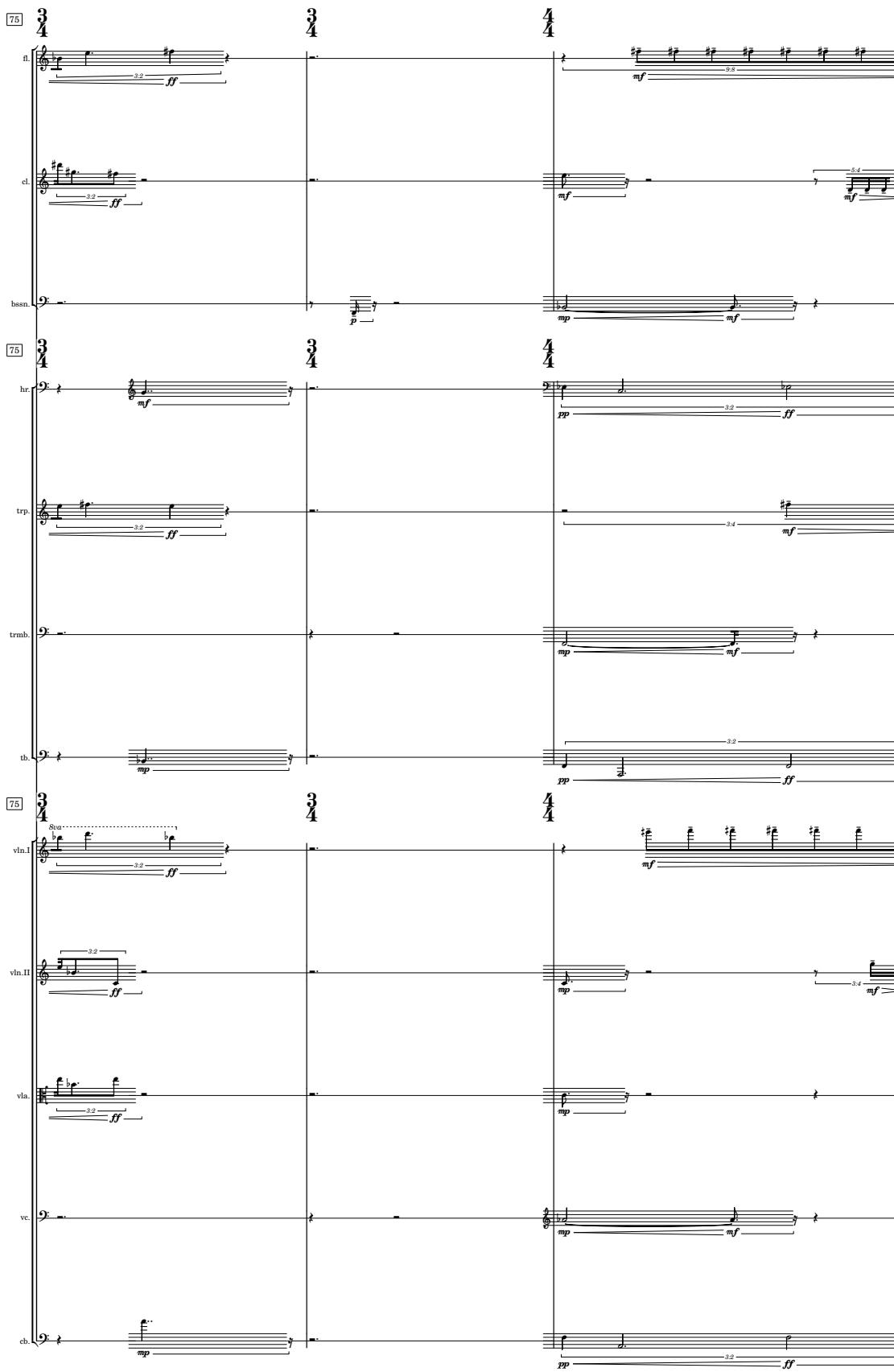
134 

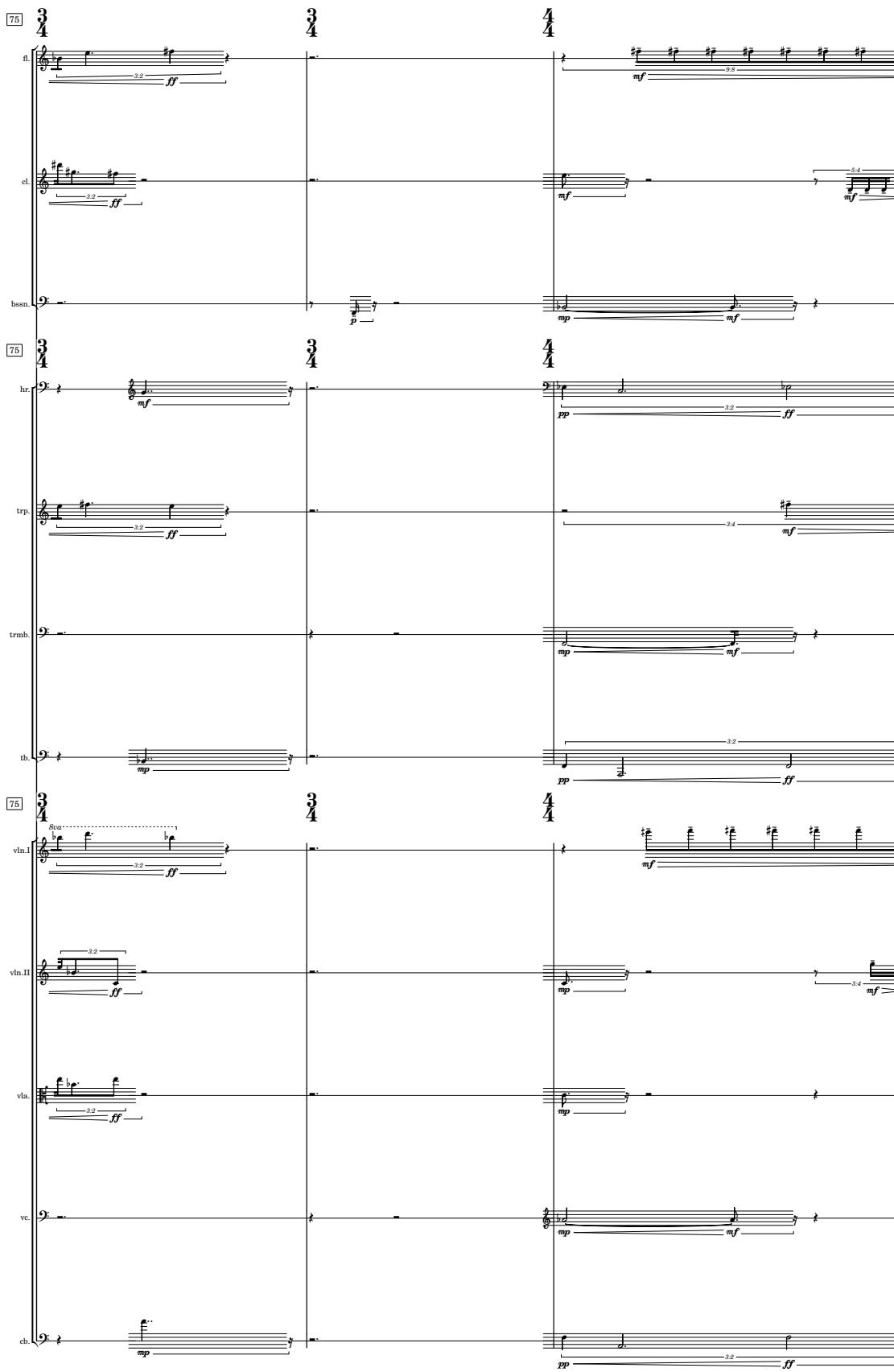
135 

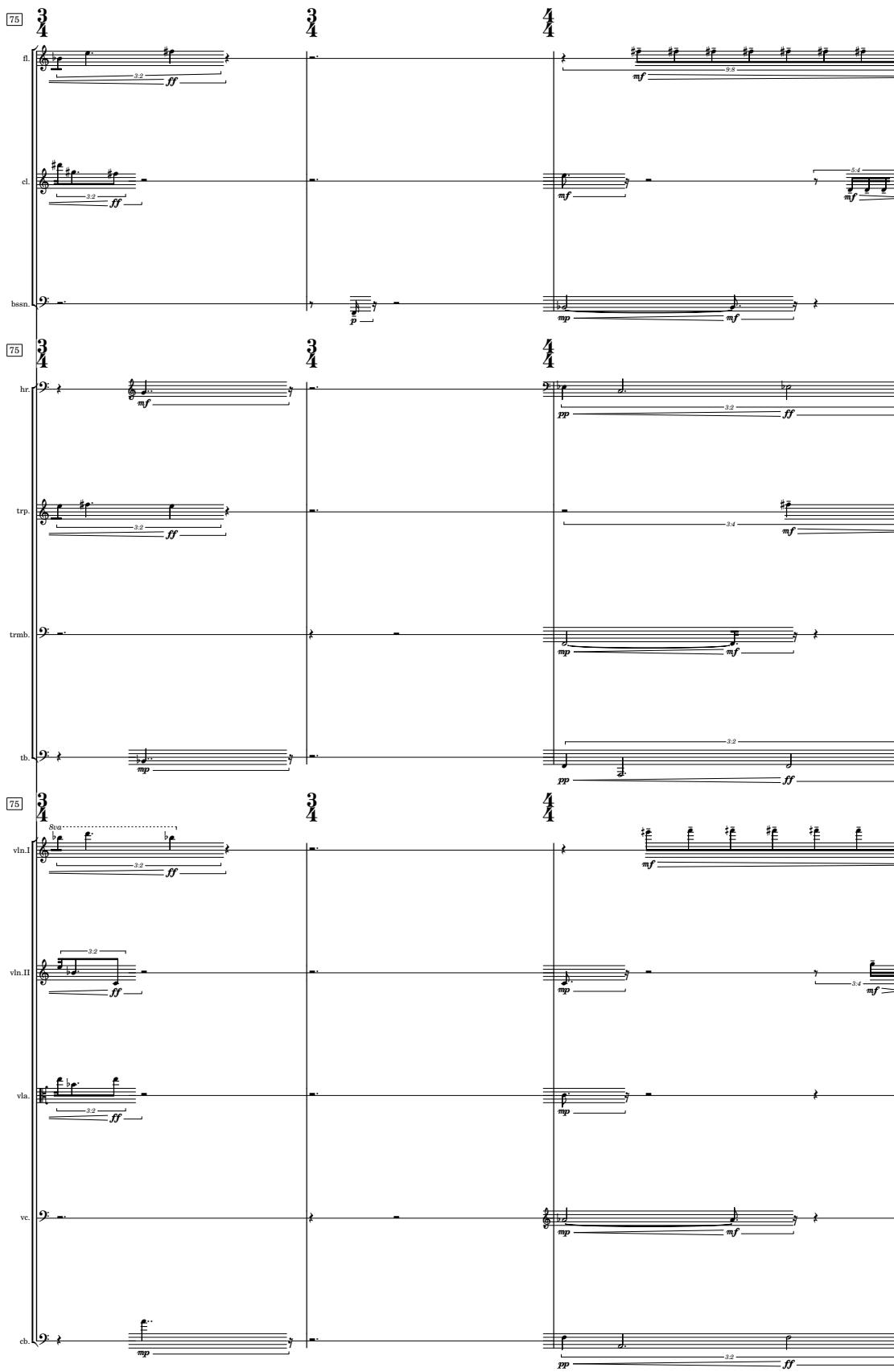
136 

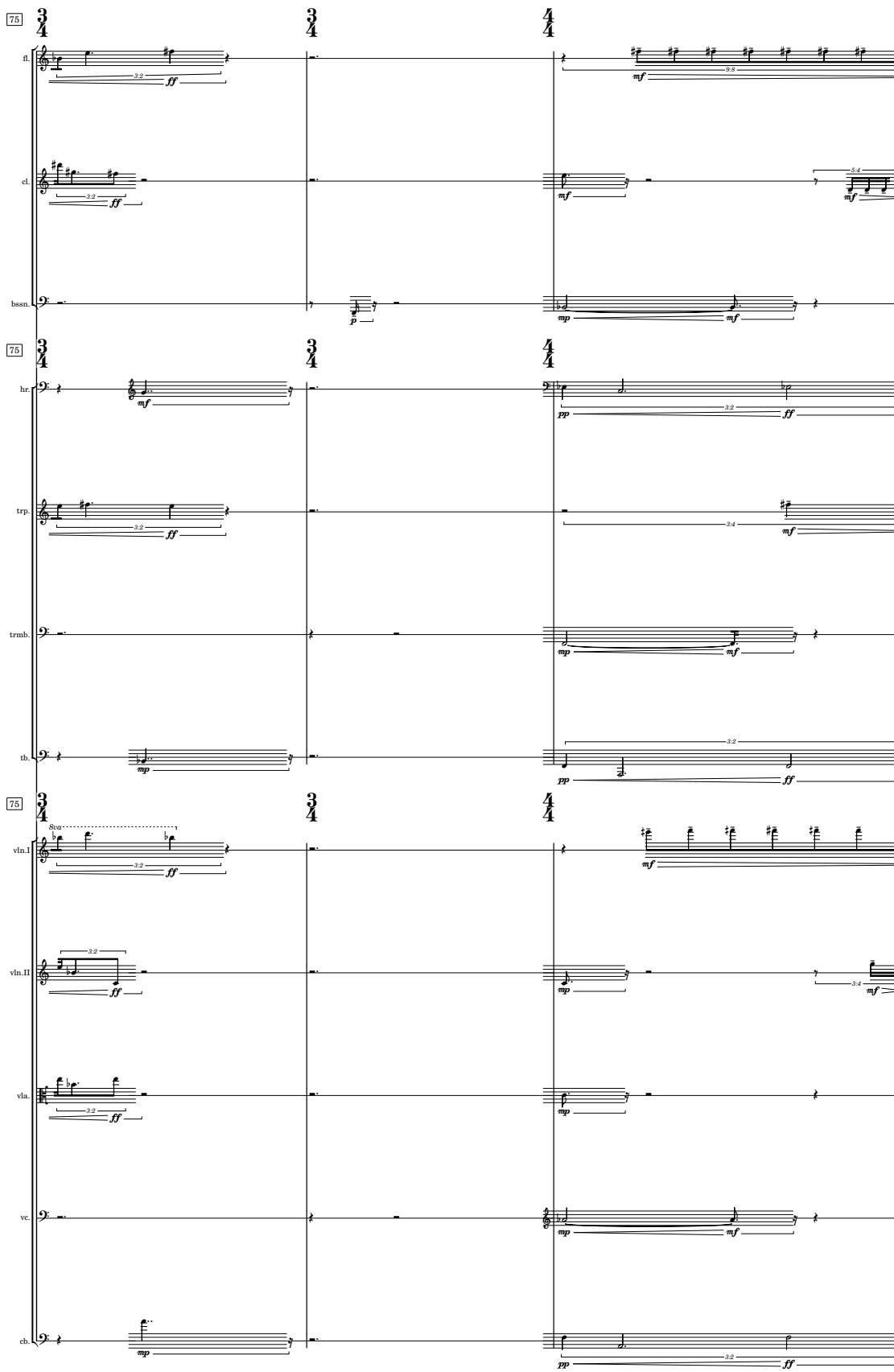
137 

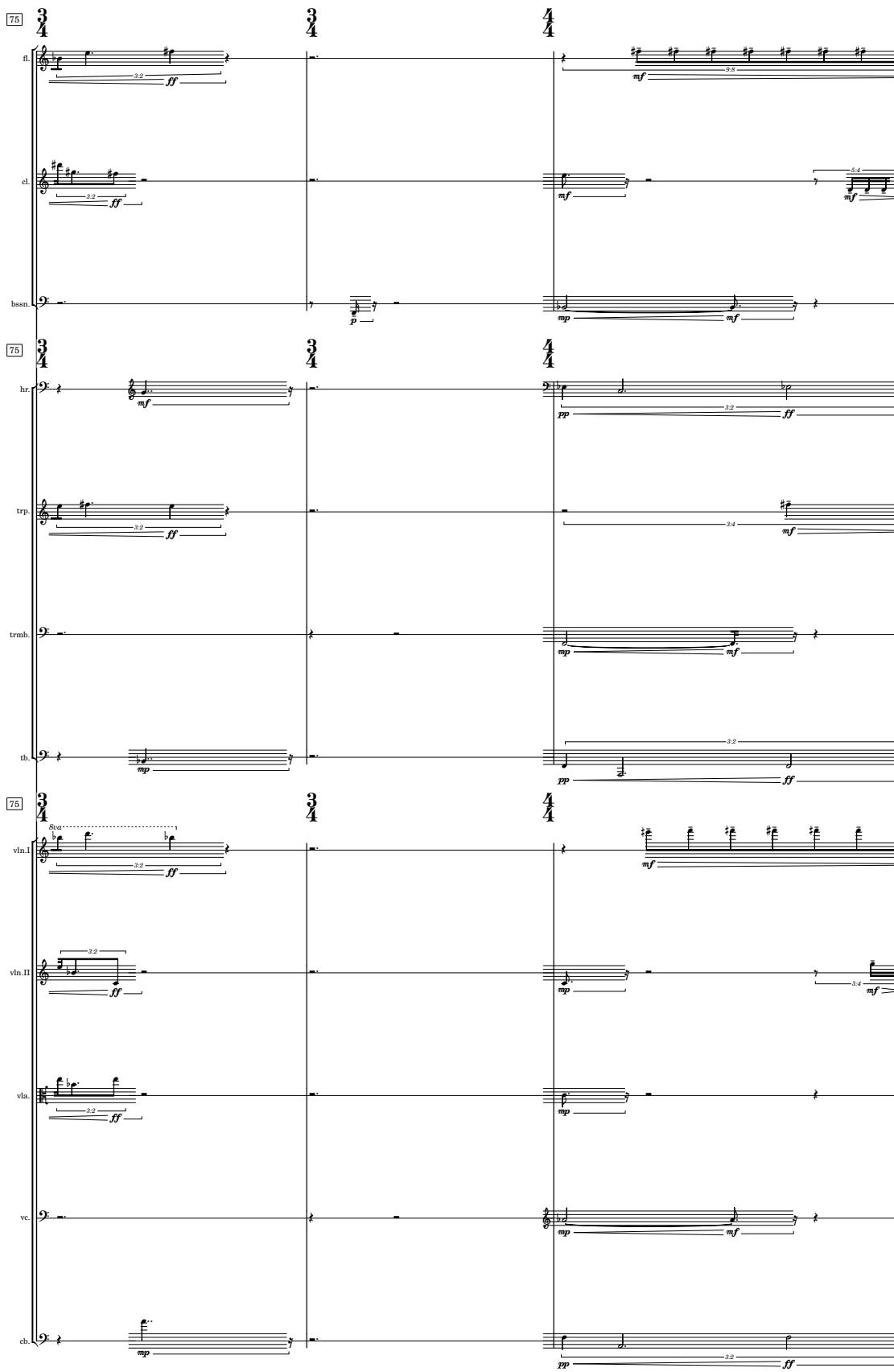
138 

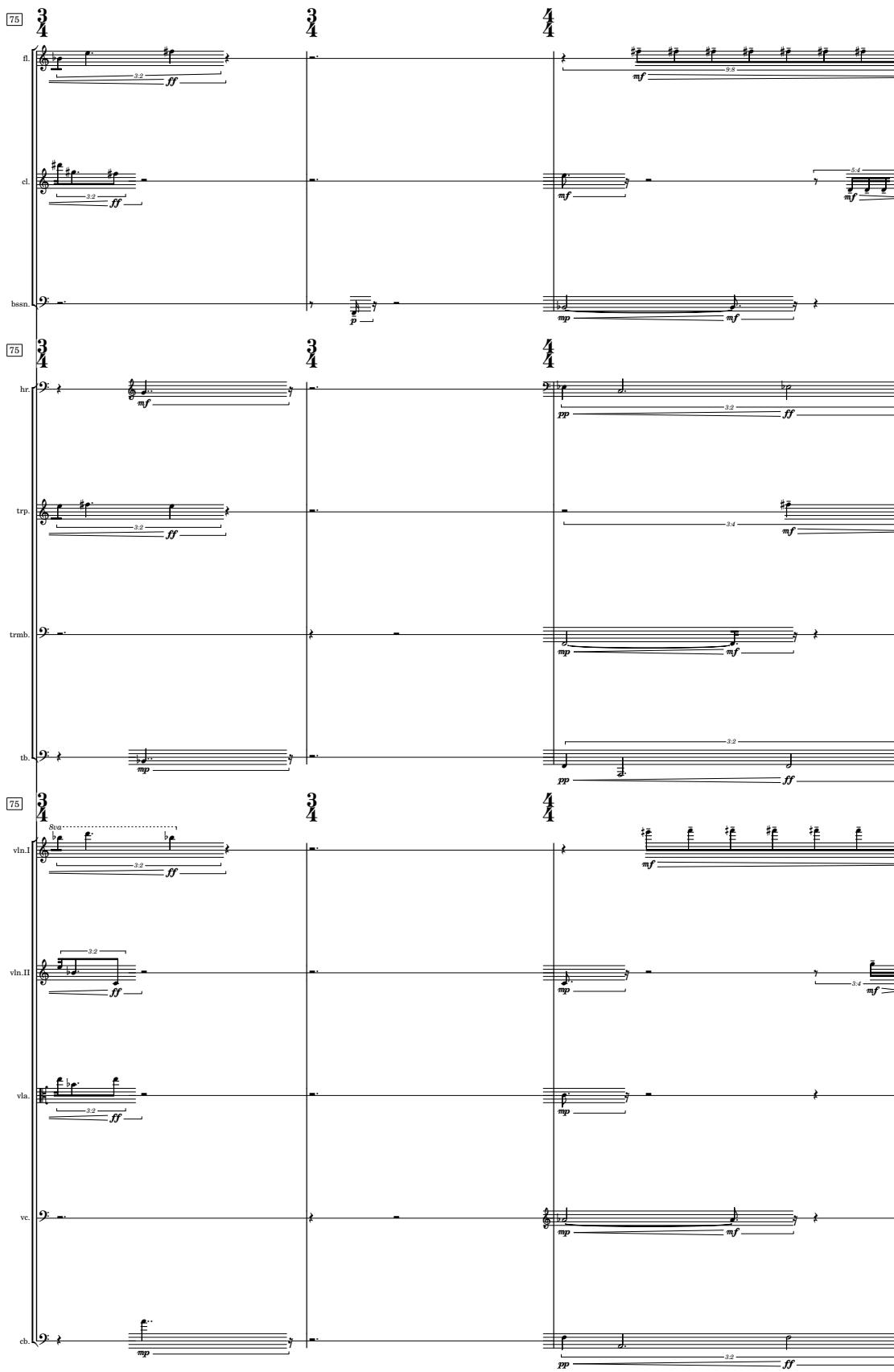
139 

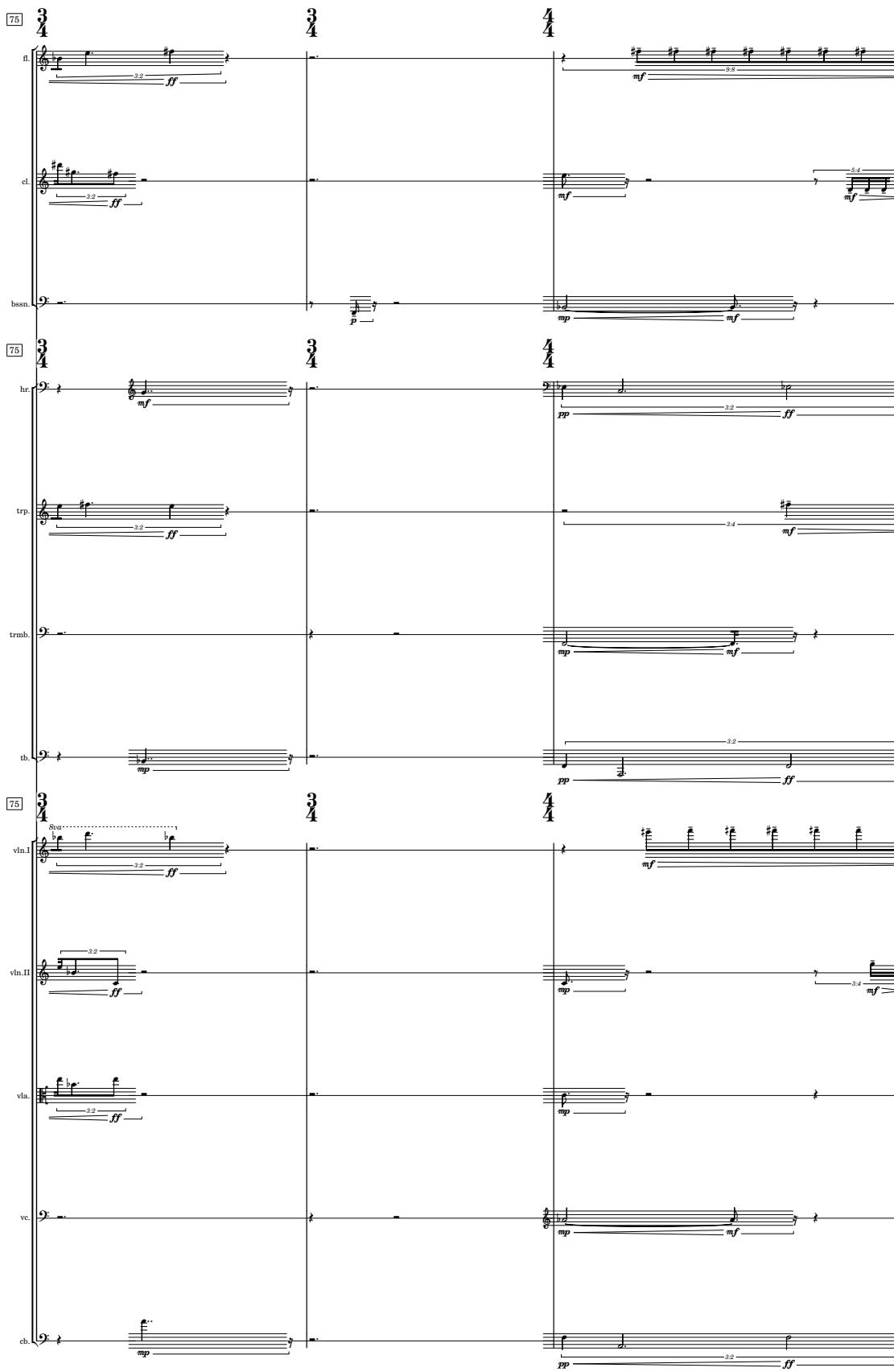
140 

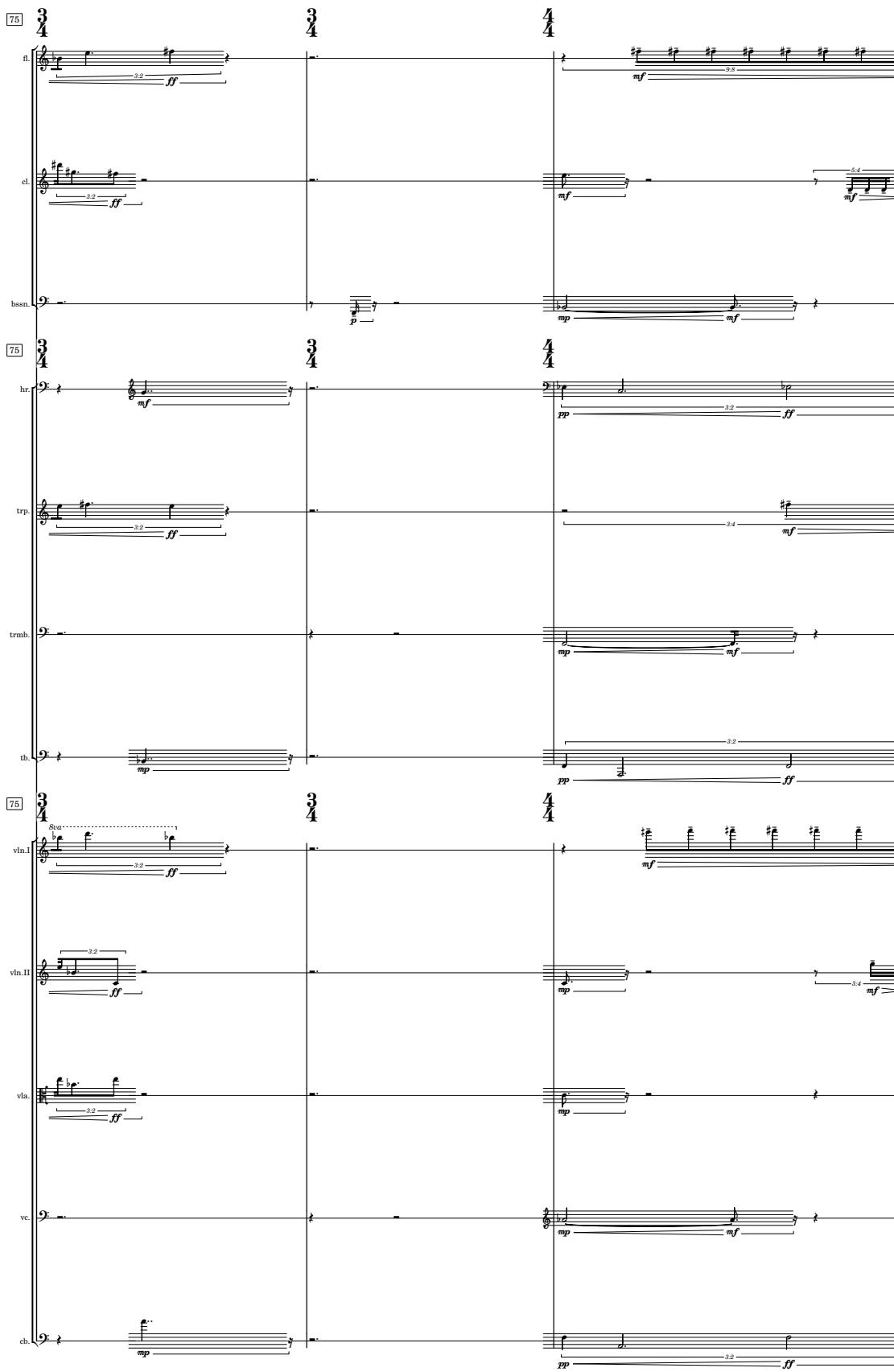
141 

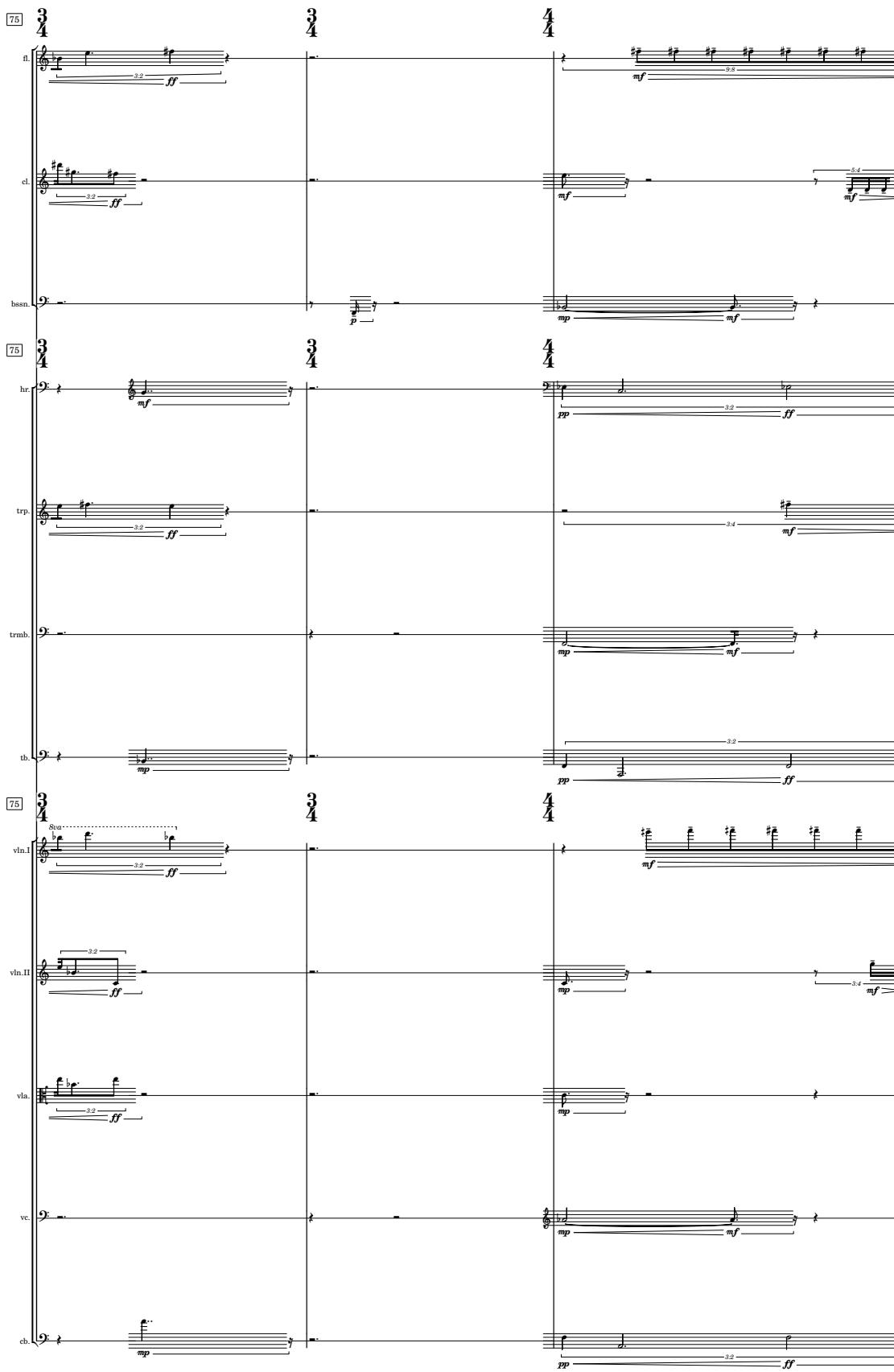
142 

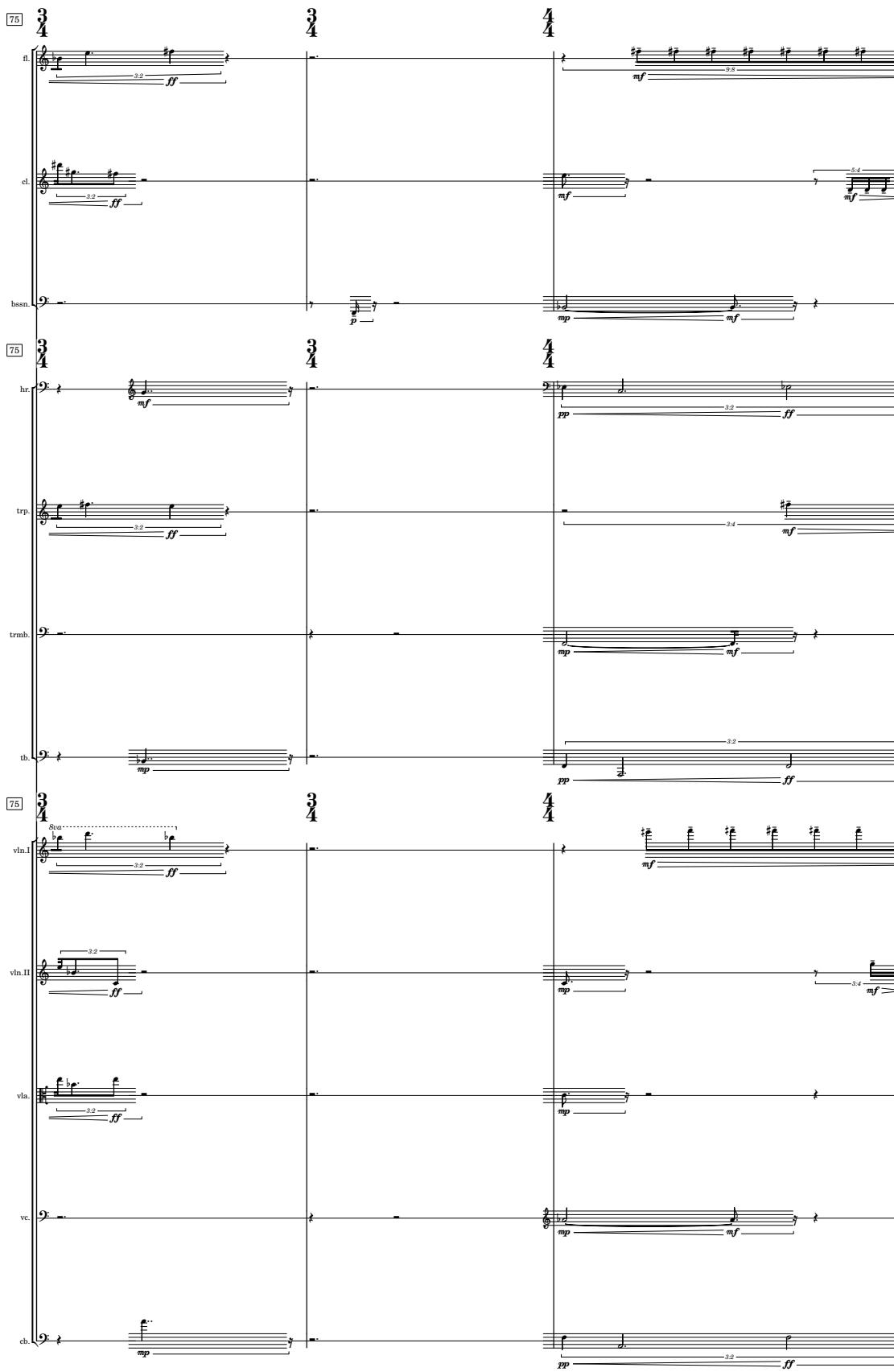
143 

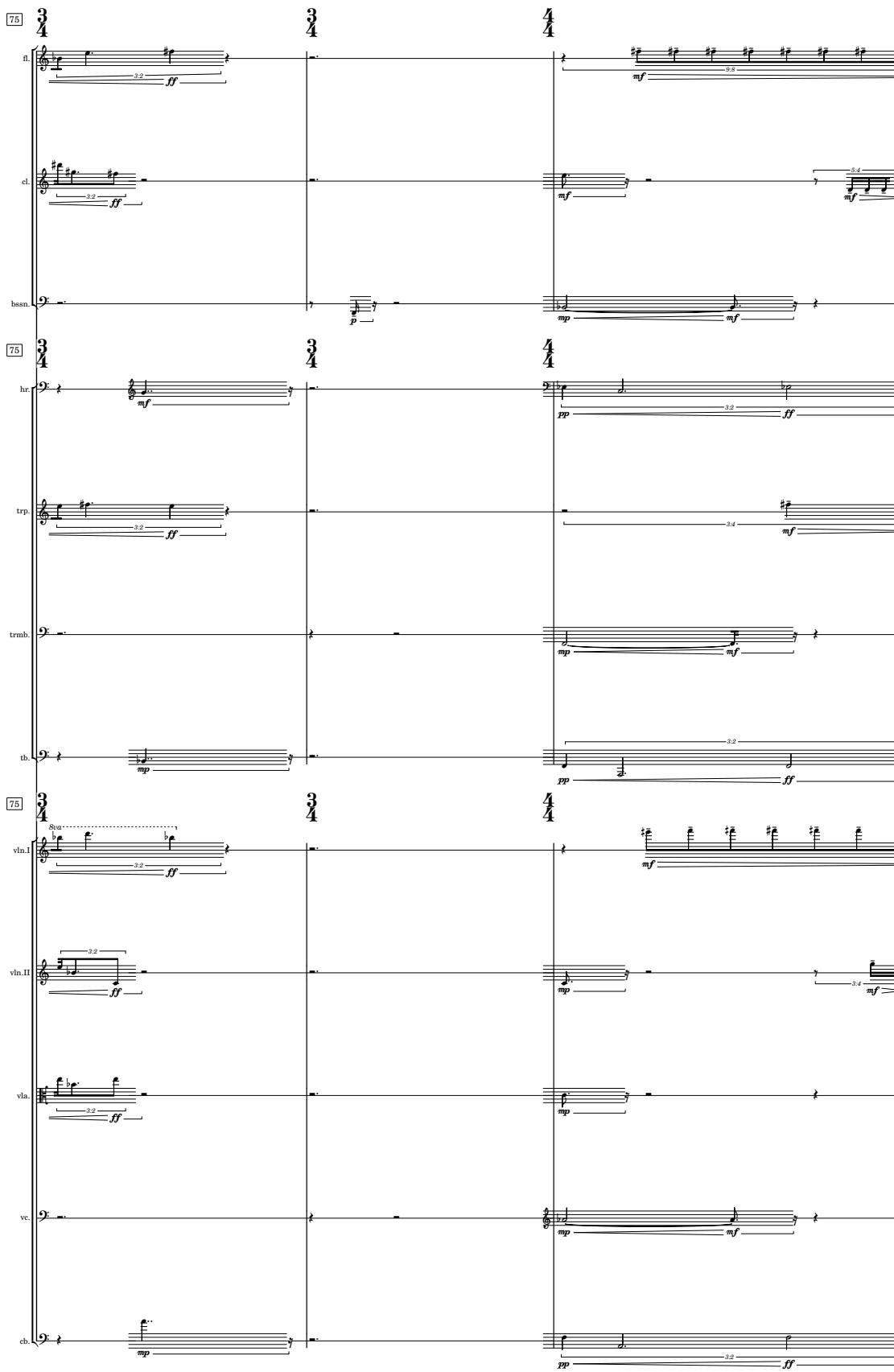
144 

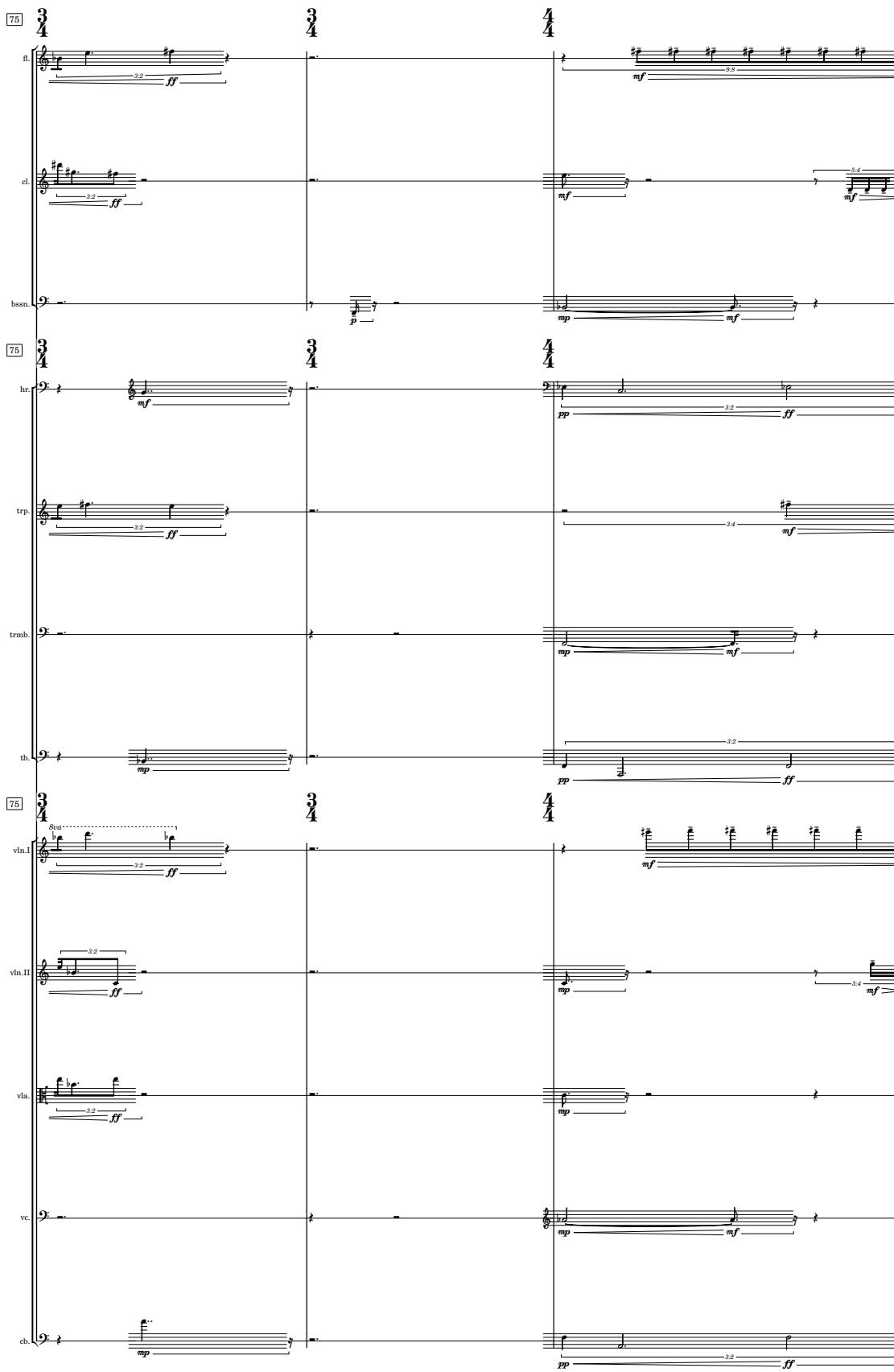
145 

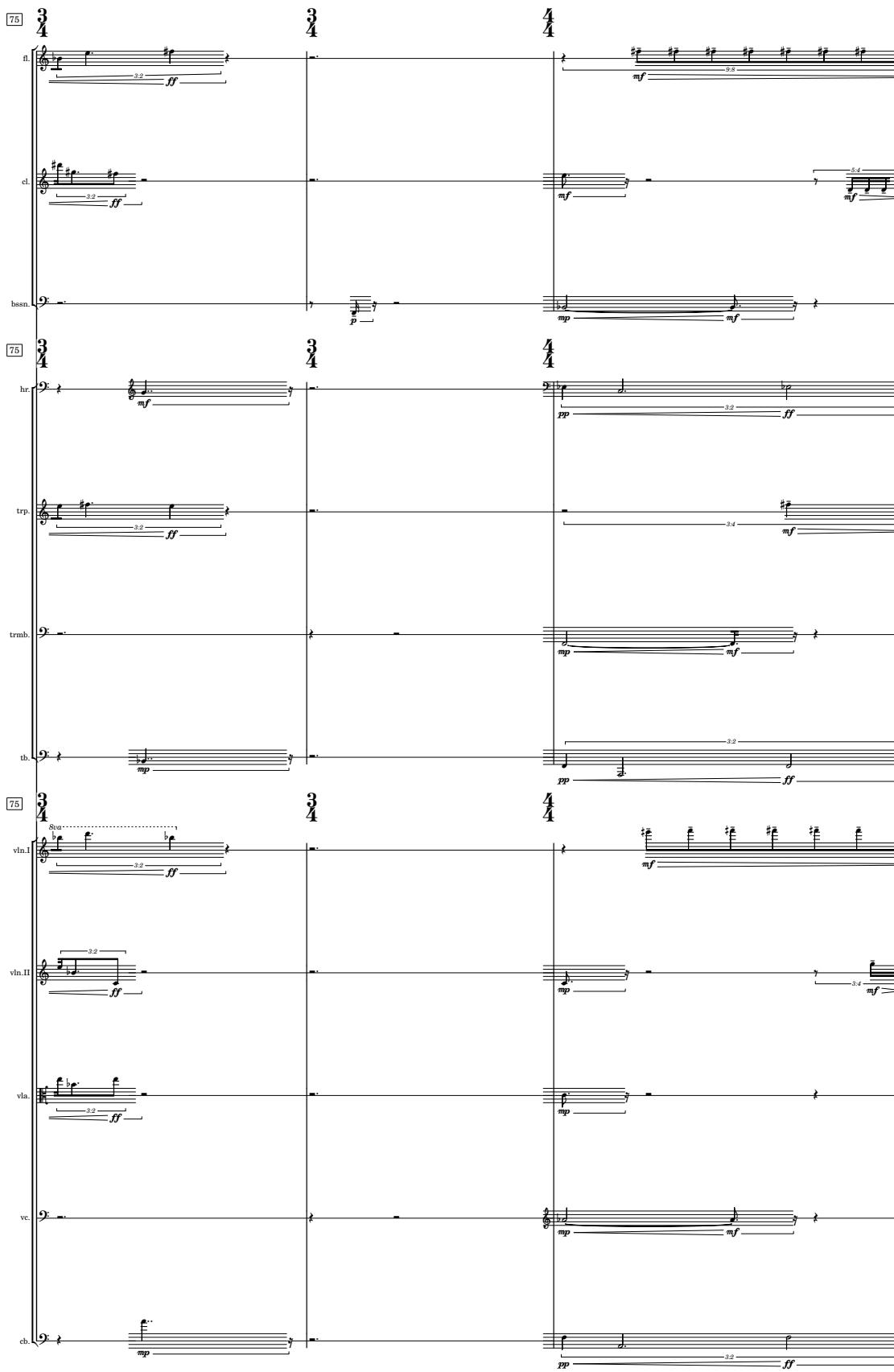
146 

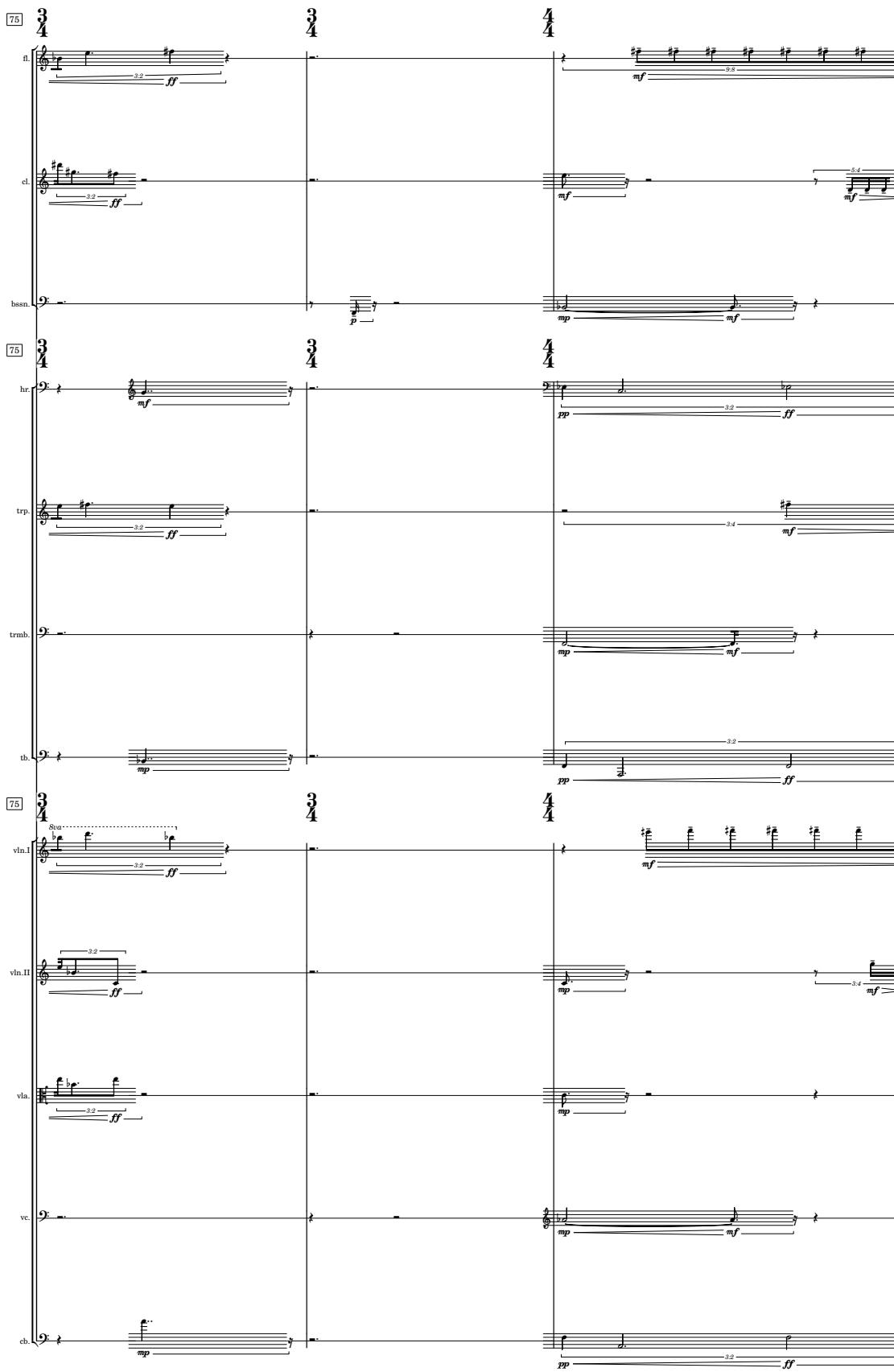
147 

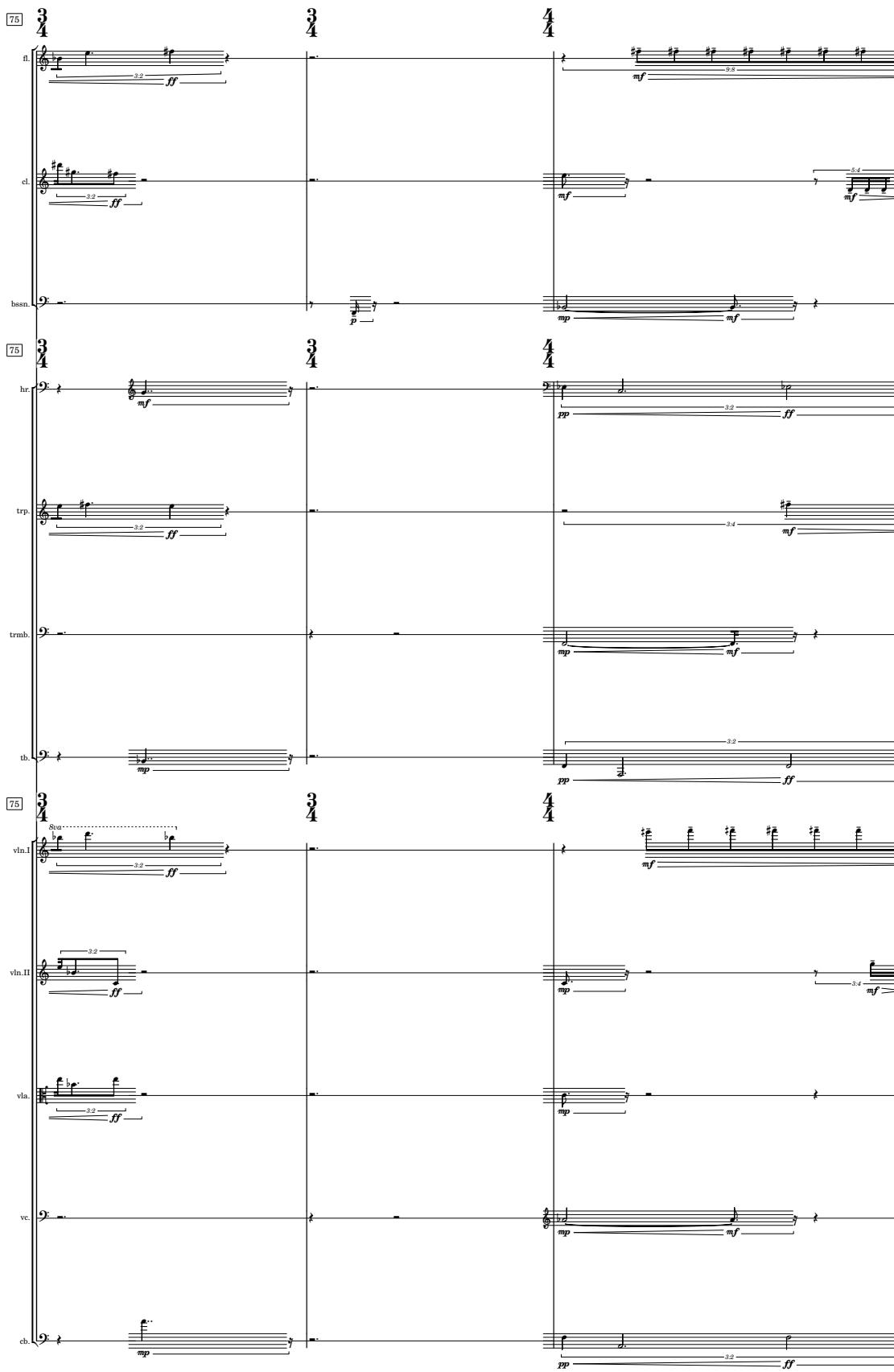
148 

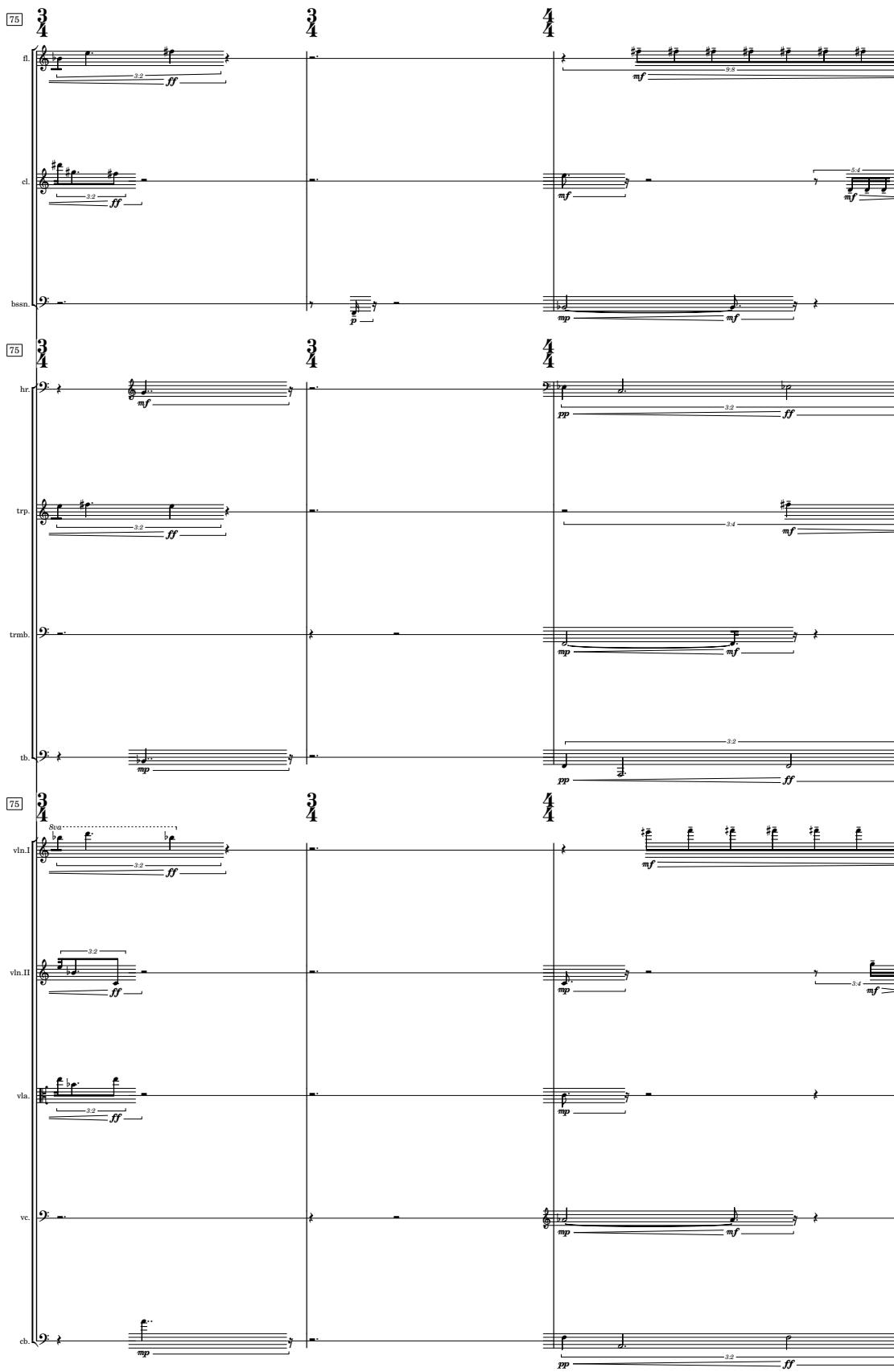
149 

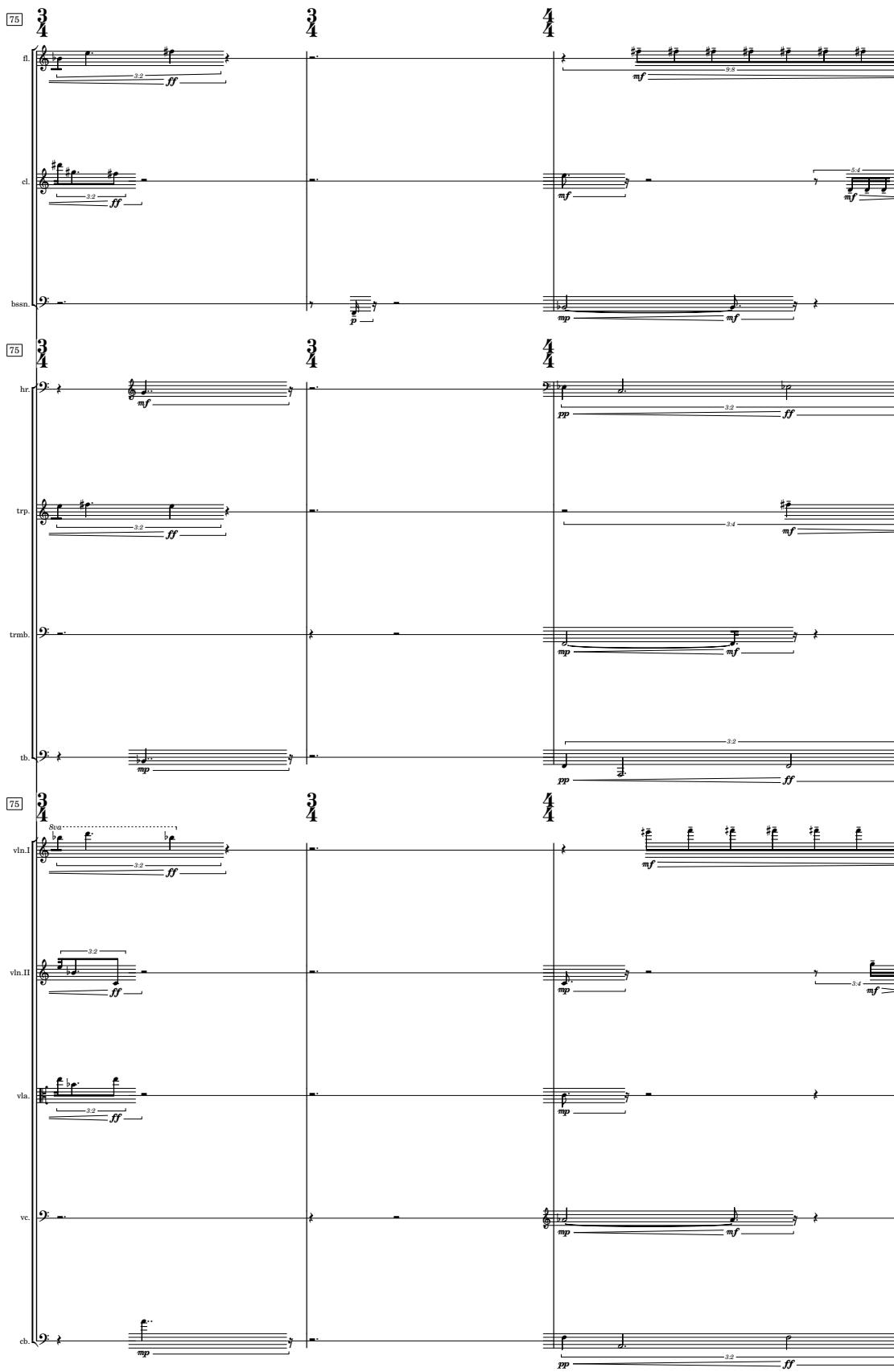
150 

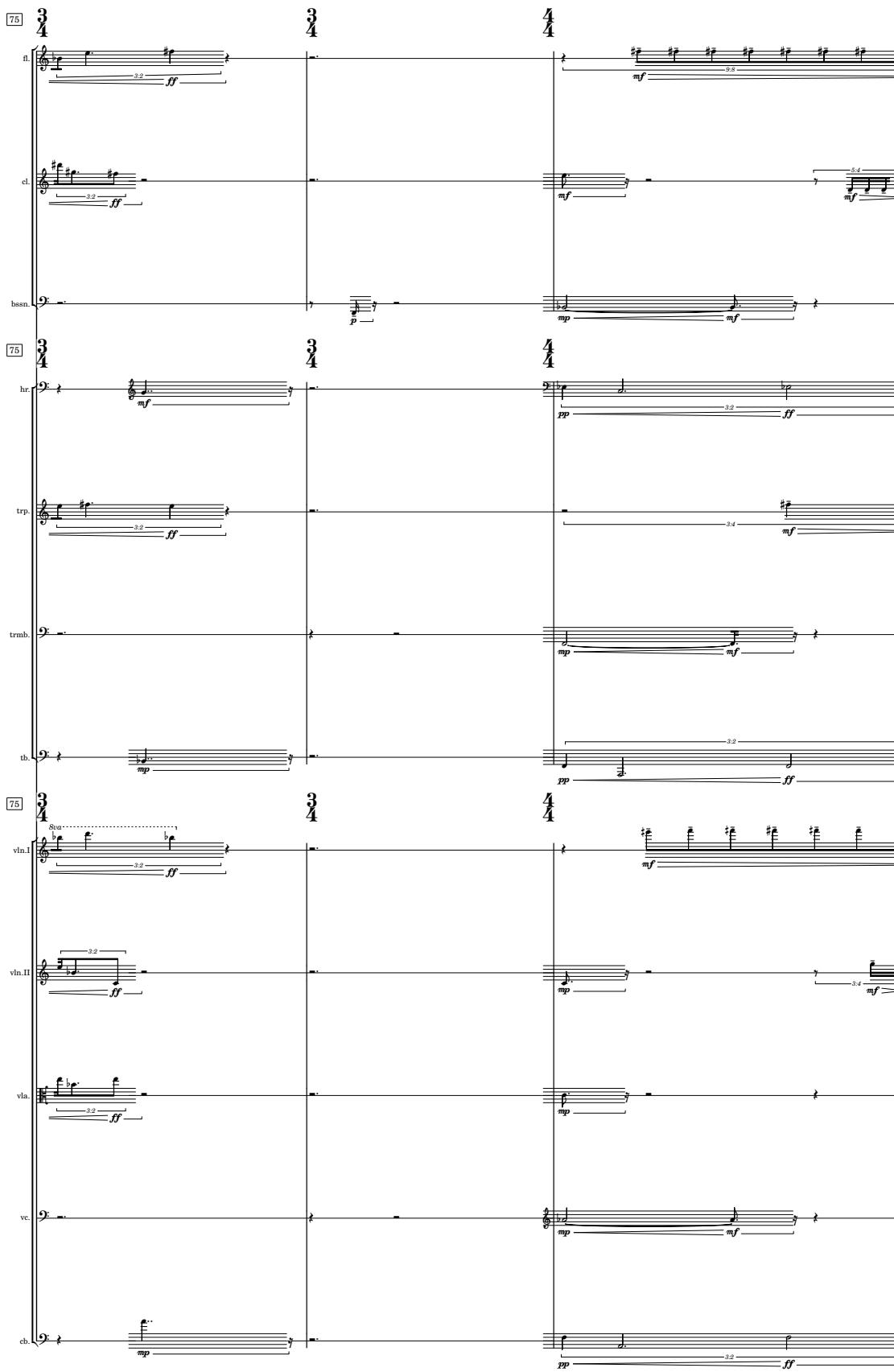
151 

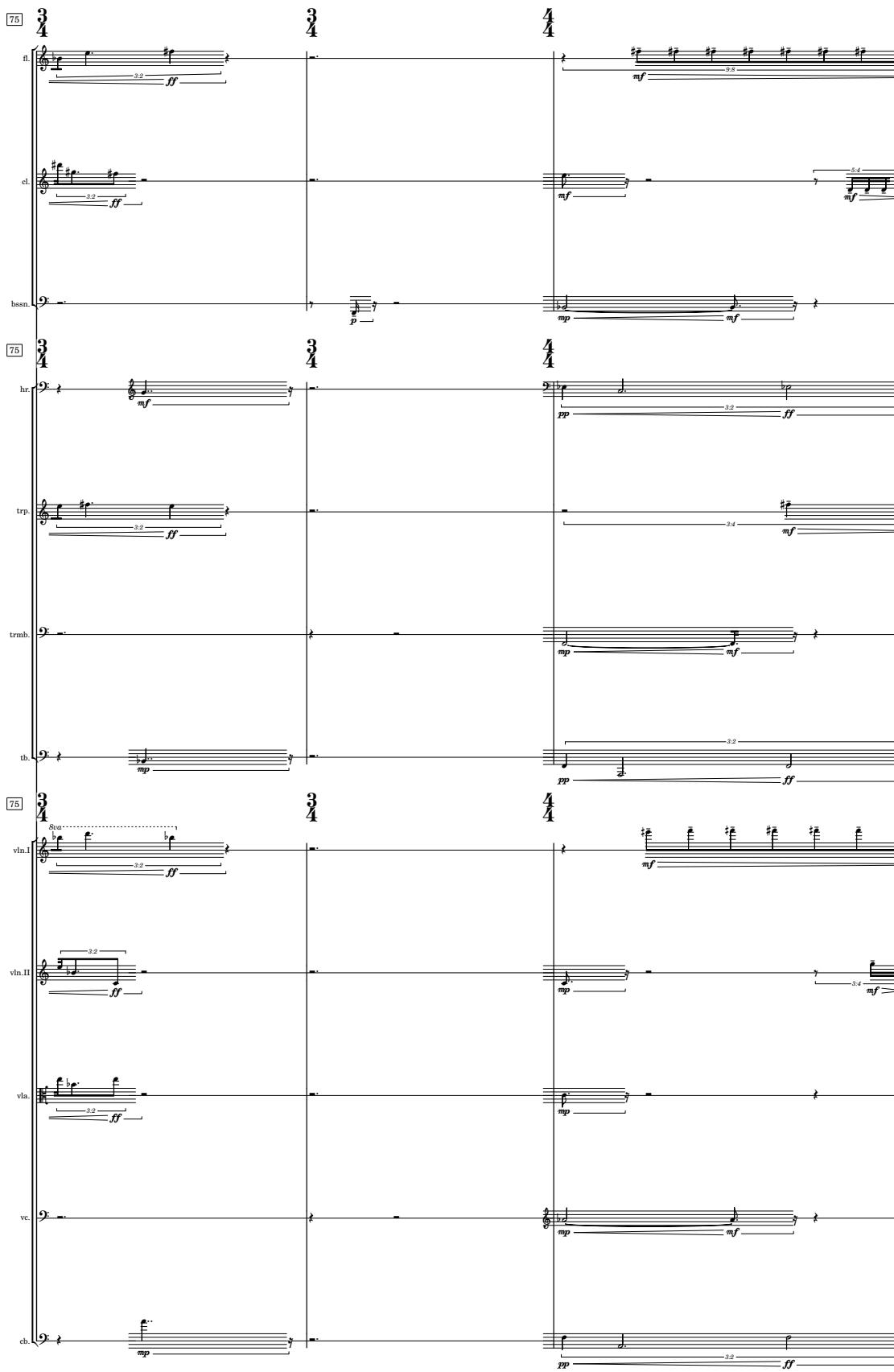
152 

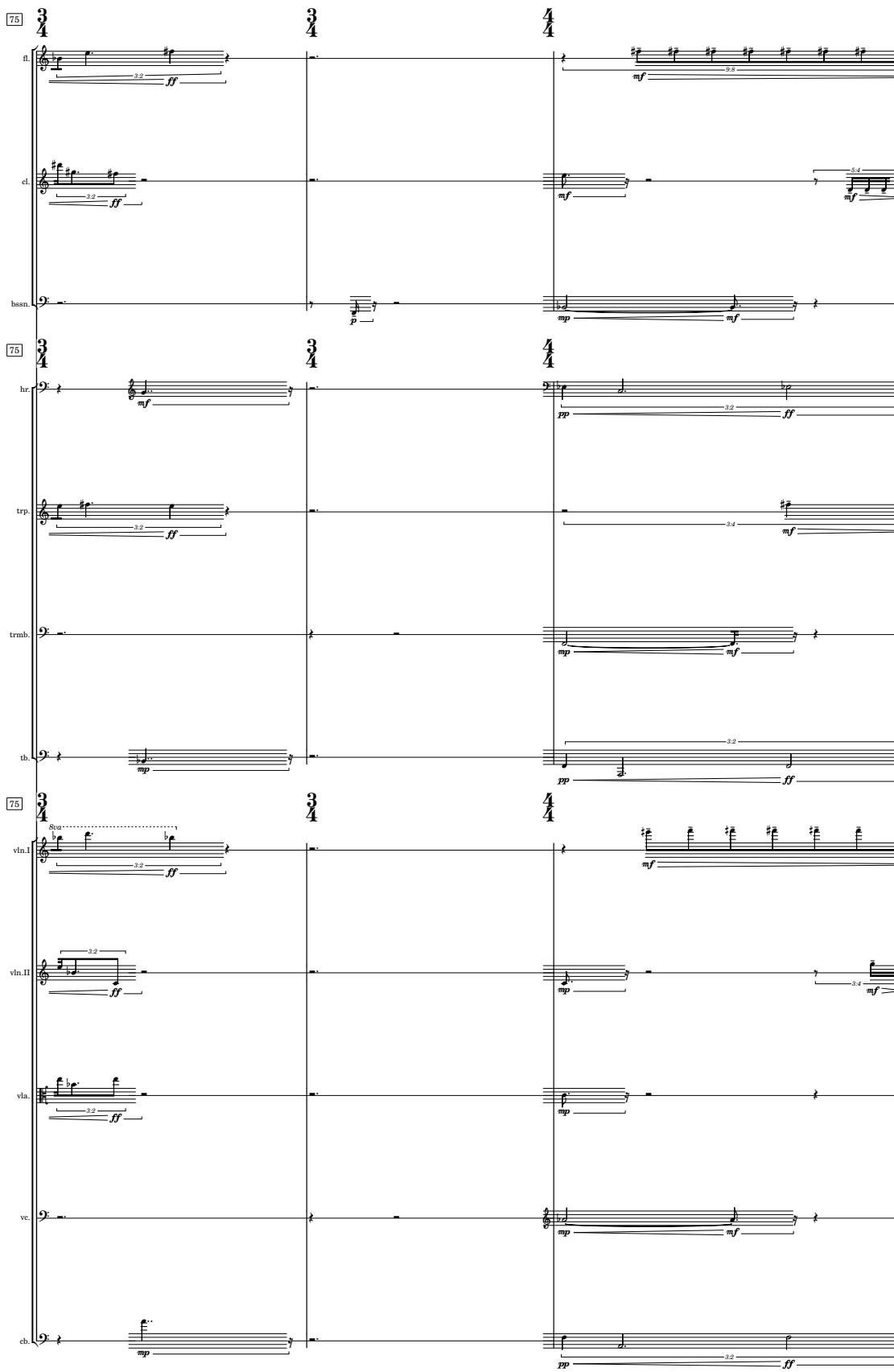
153 

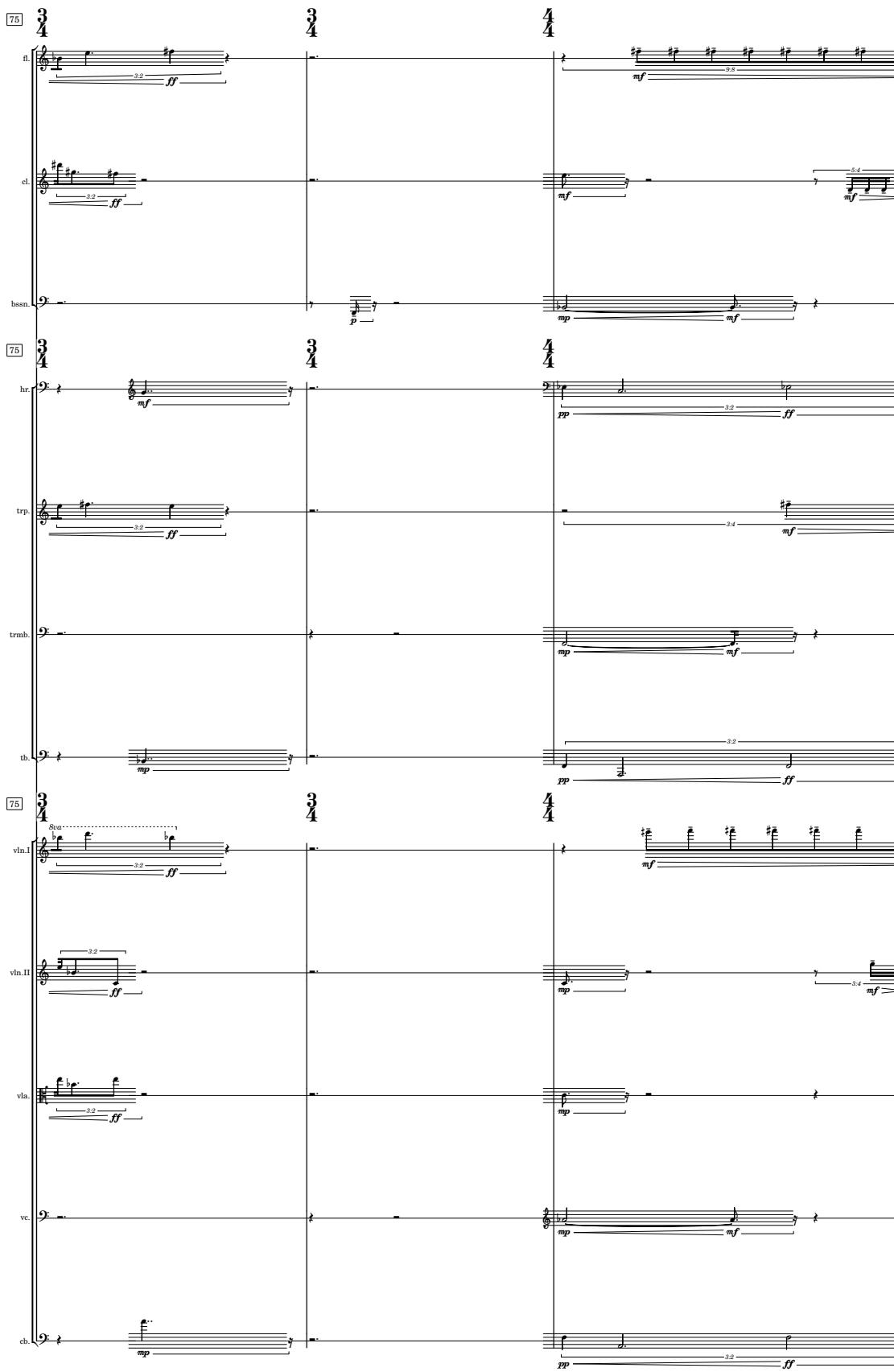
154 

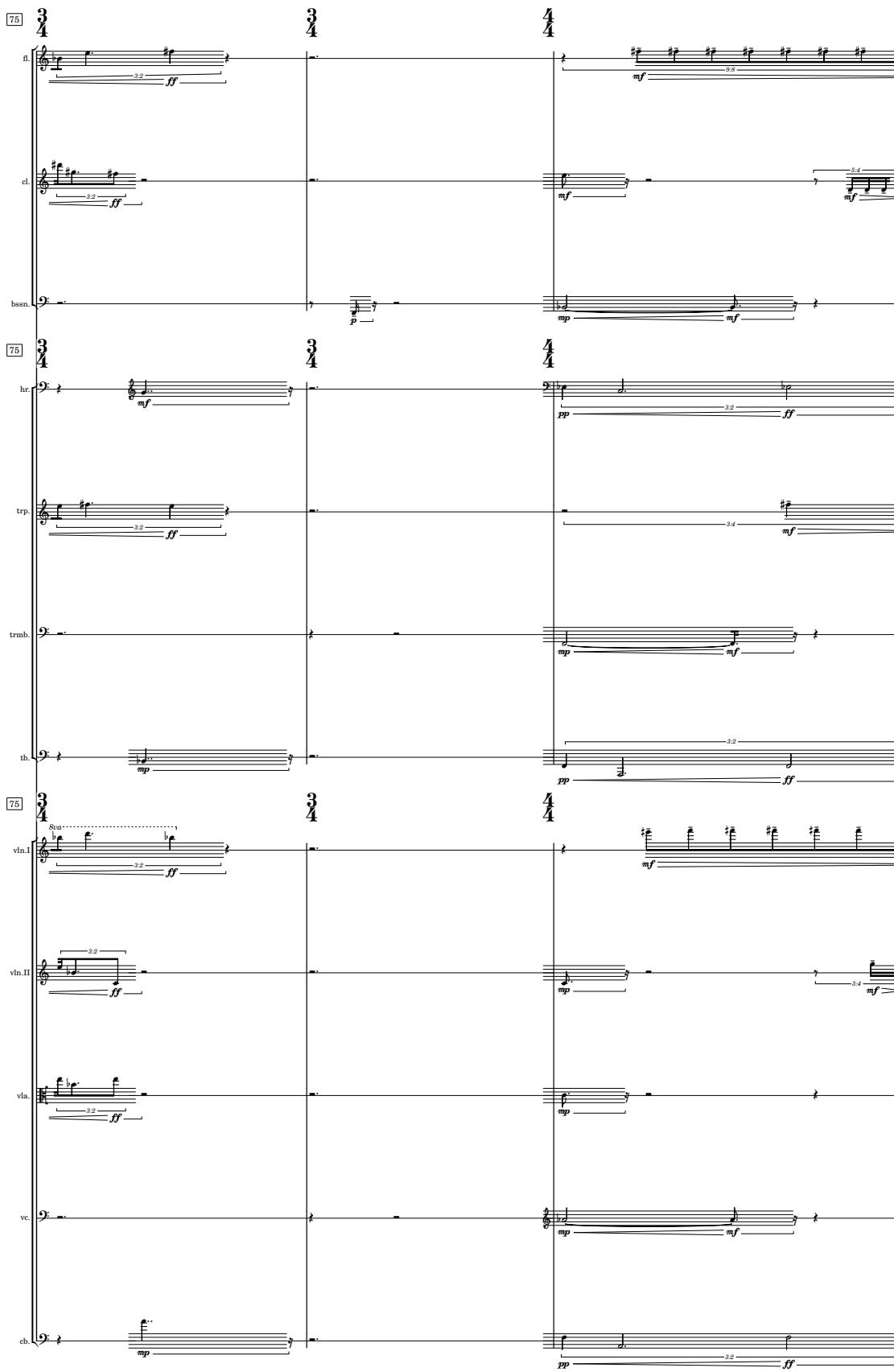
155 

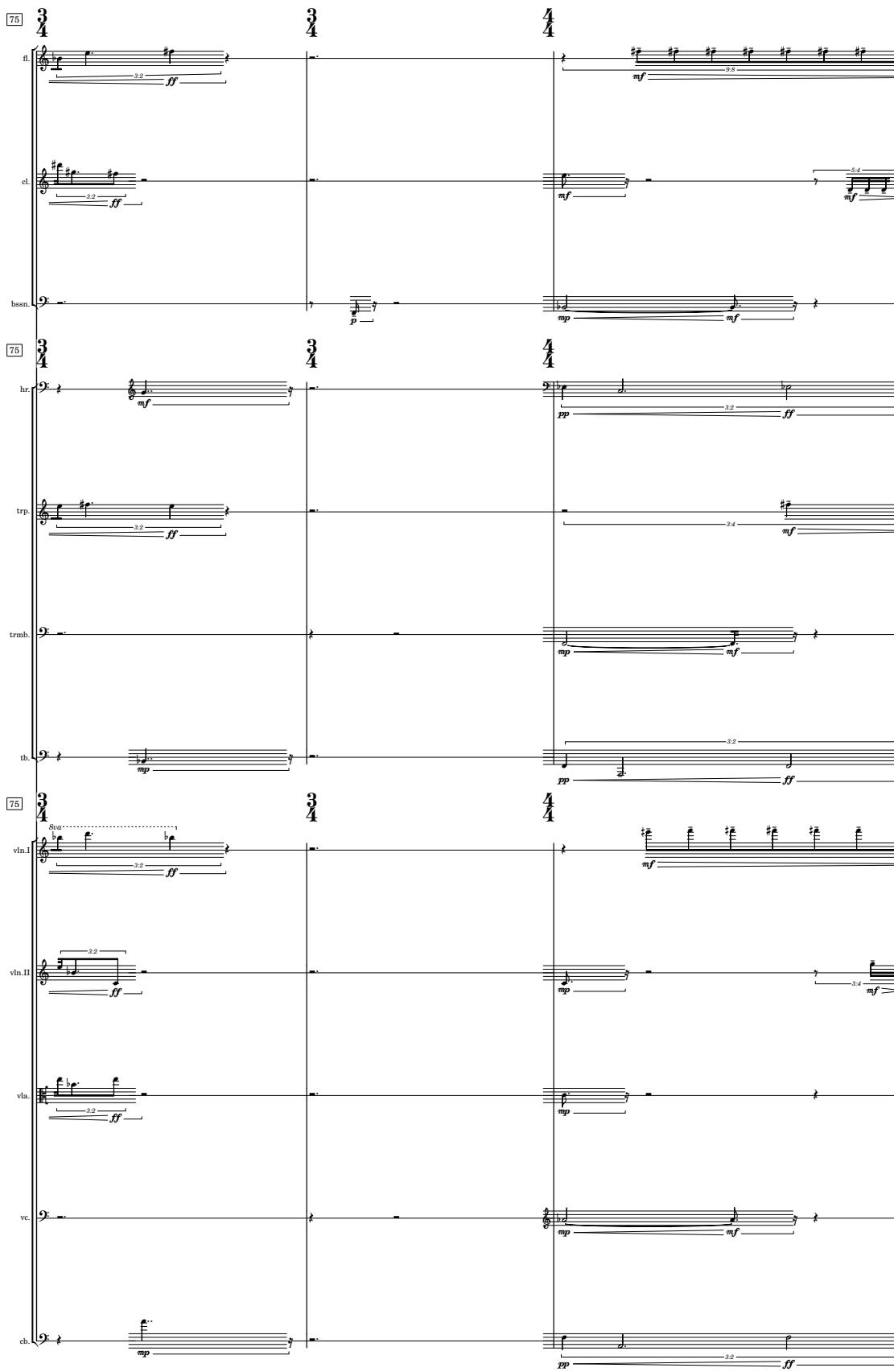
156 

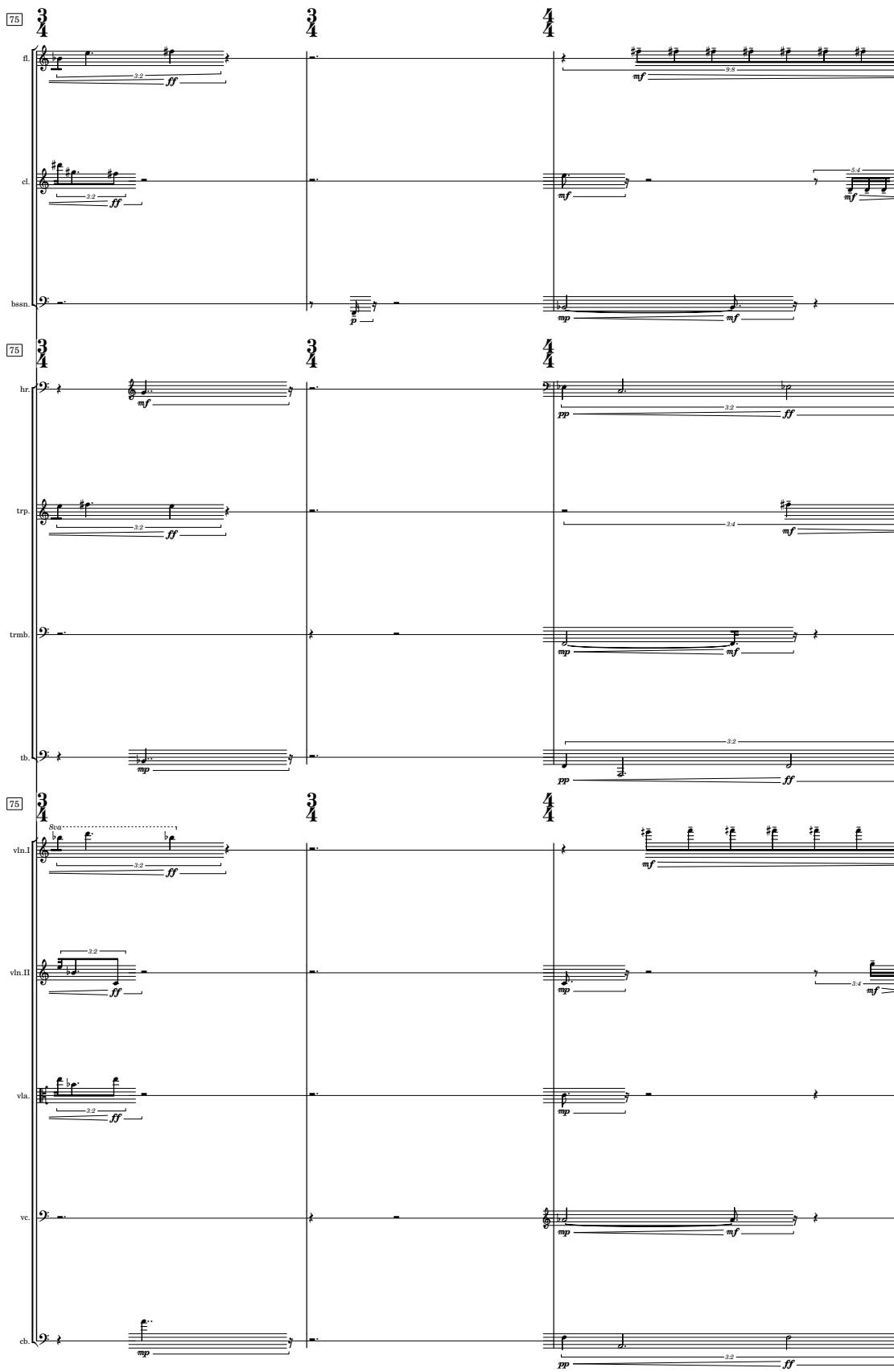
157 

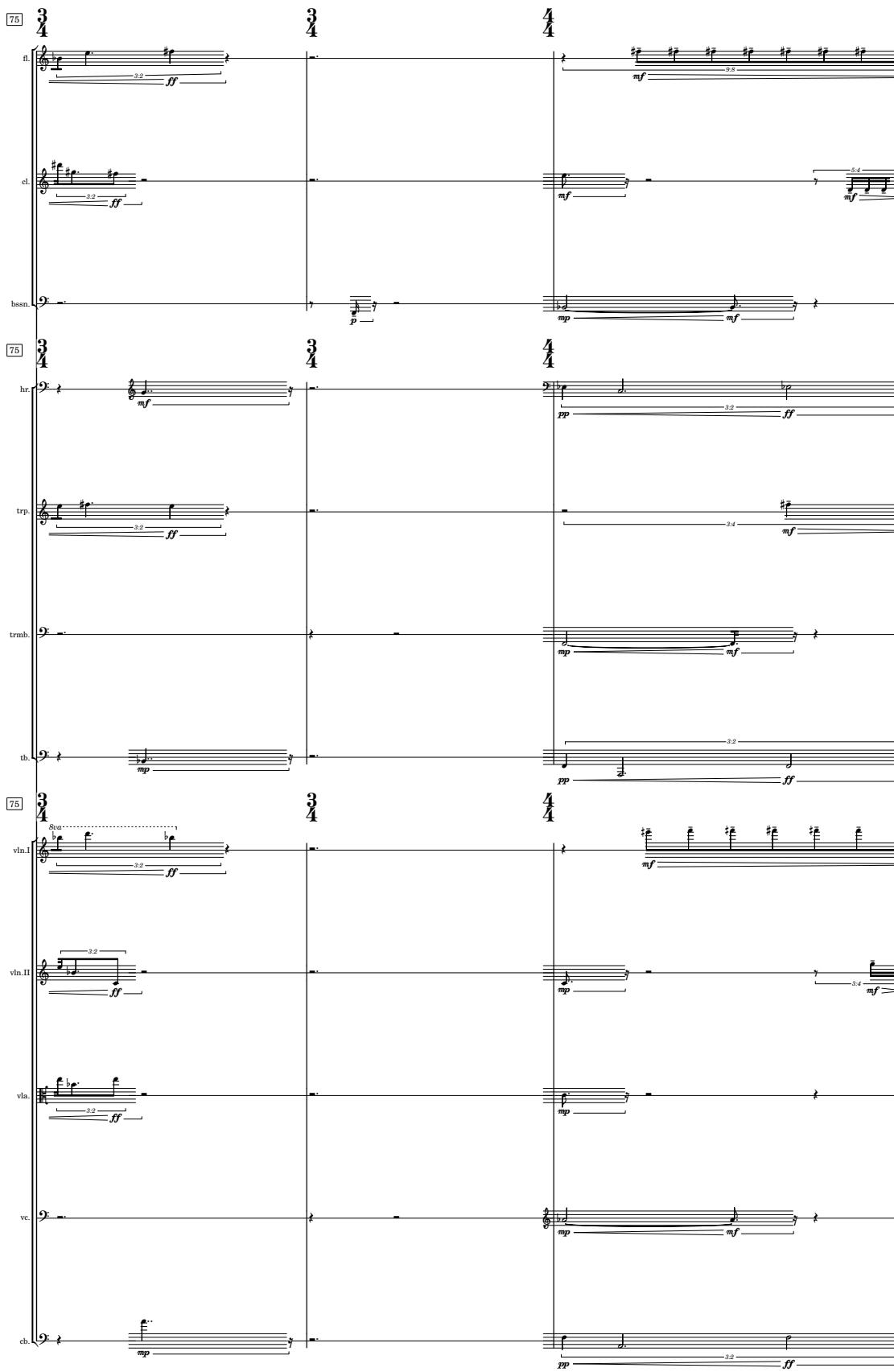
158 

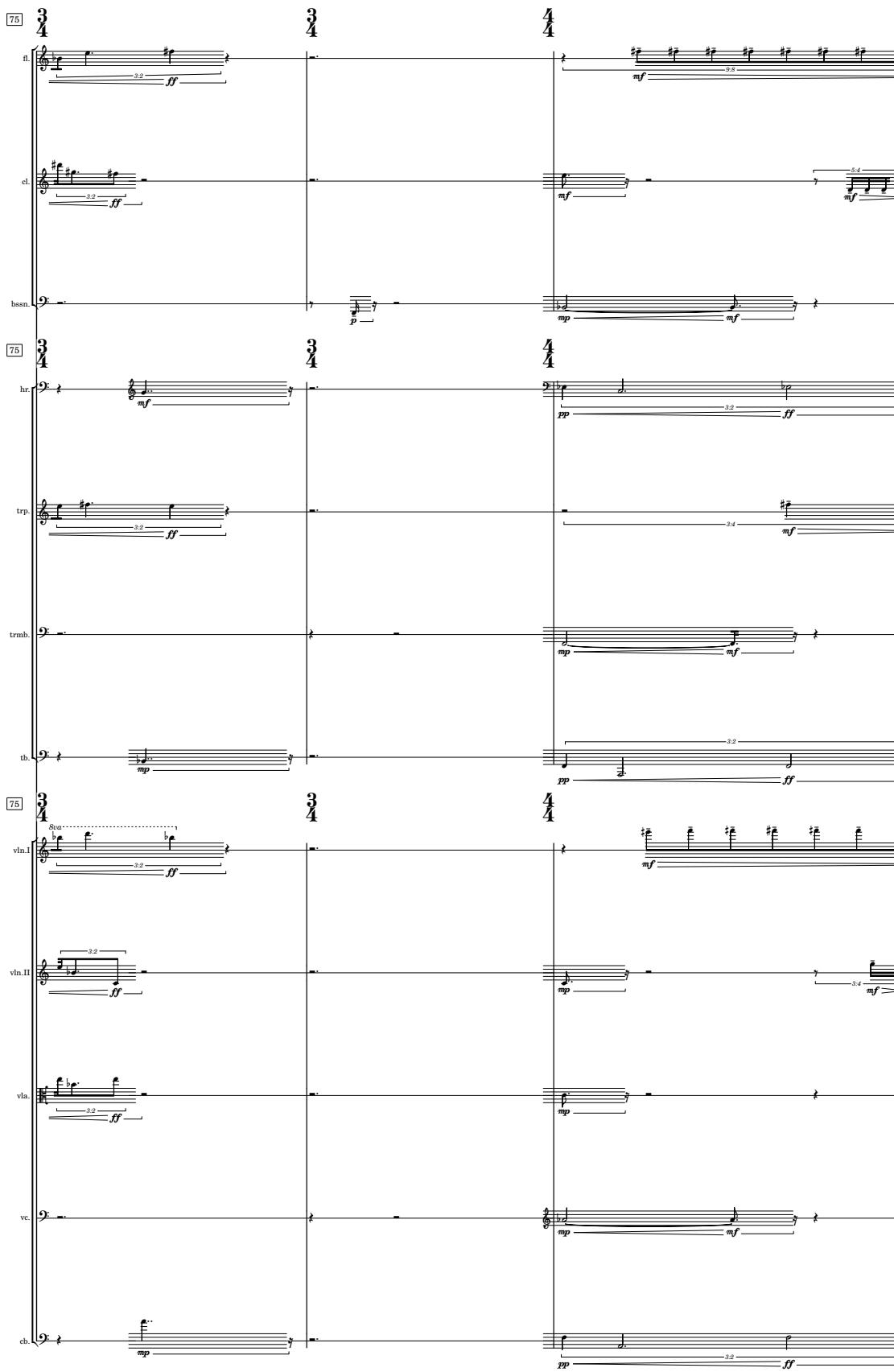
159 

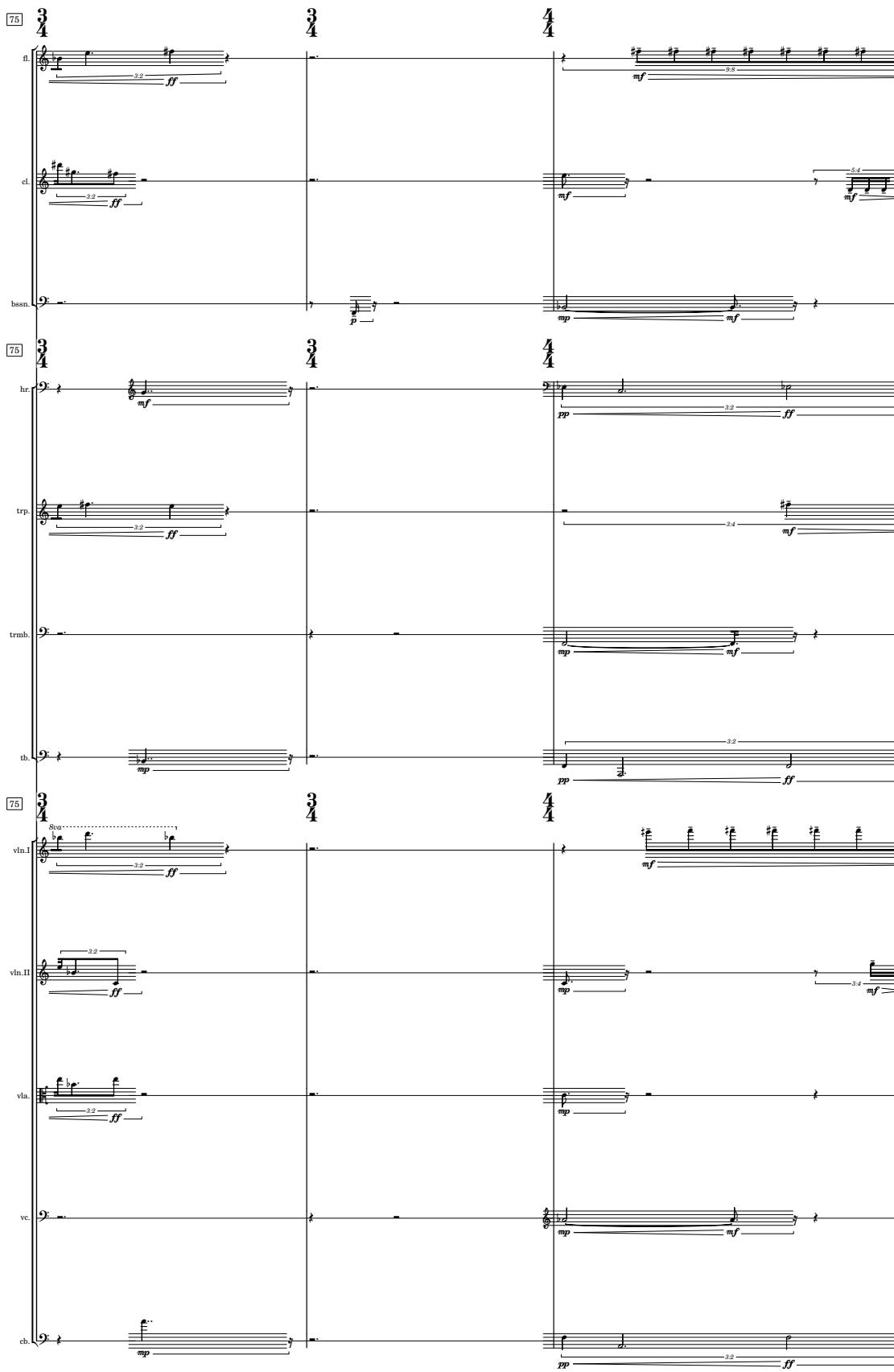
160 

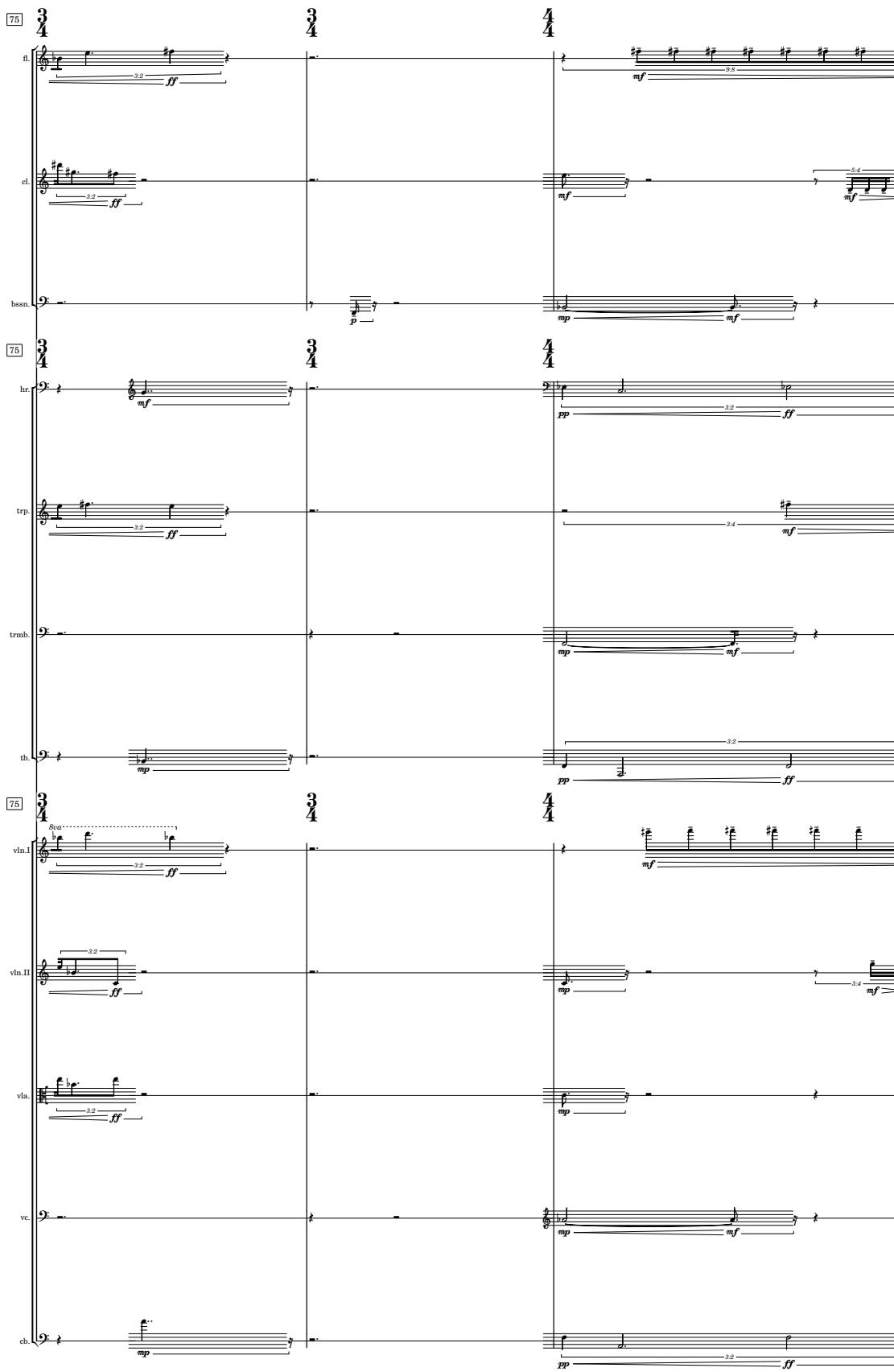
161 

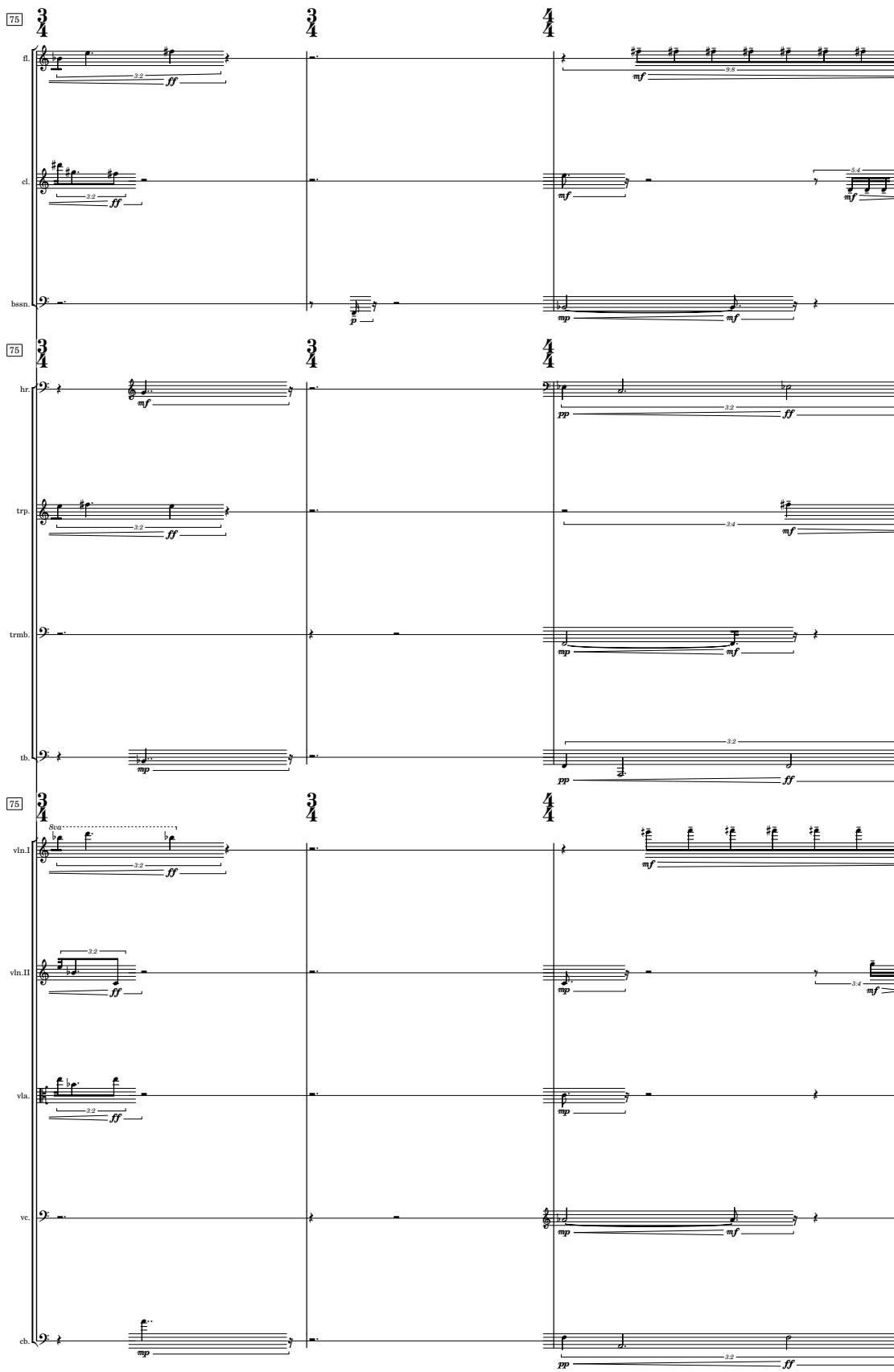
162 

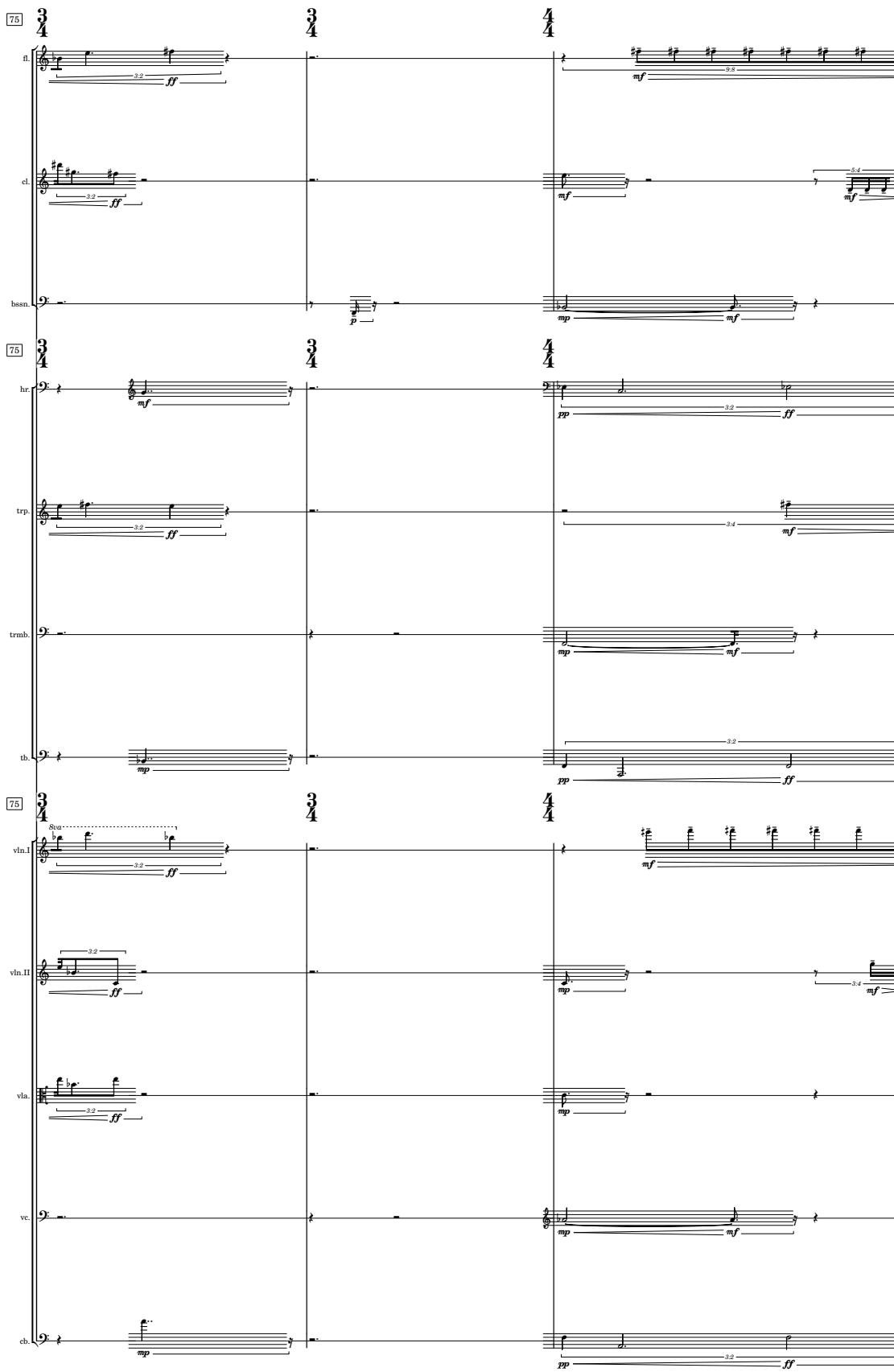
163 

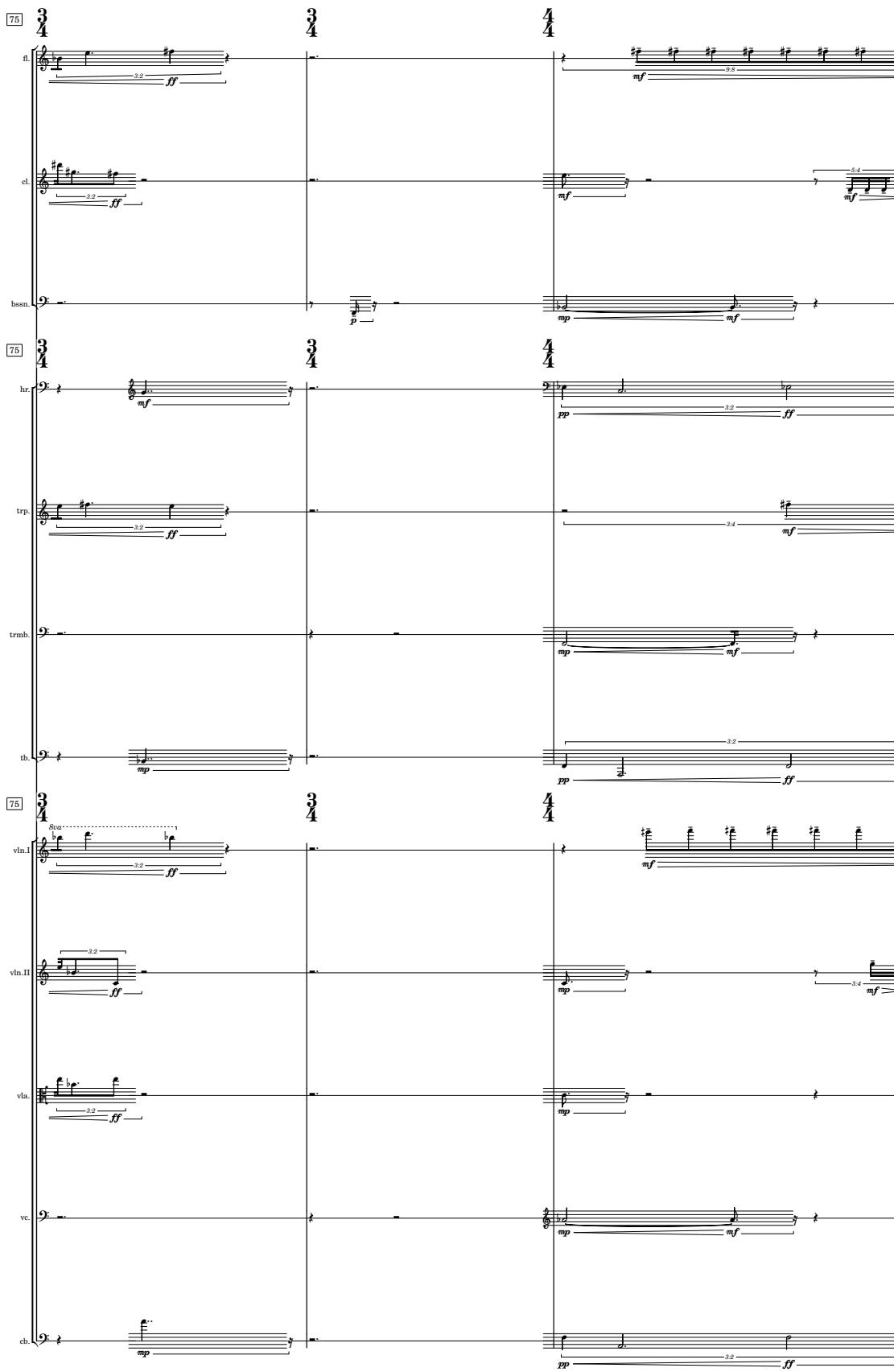
164 

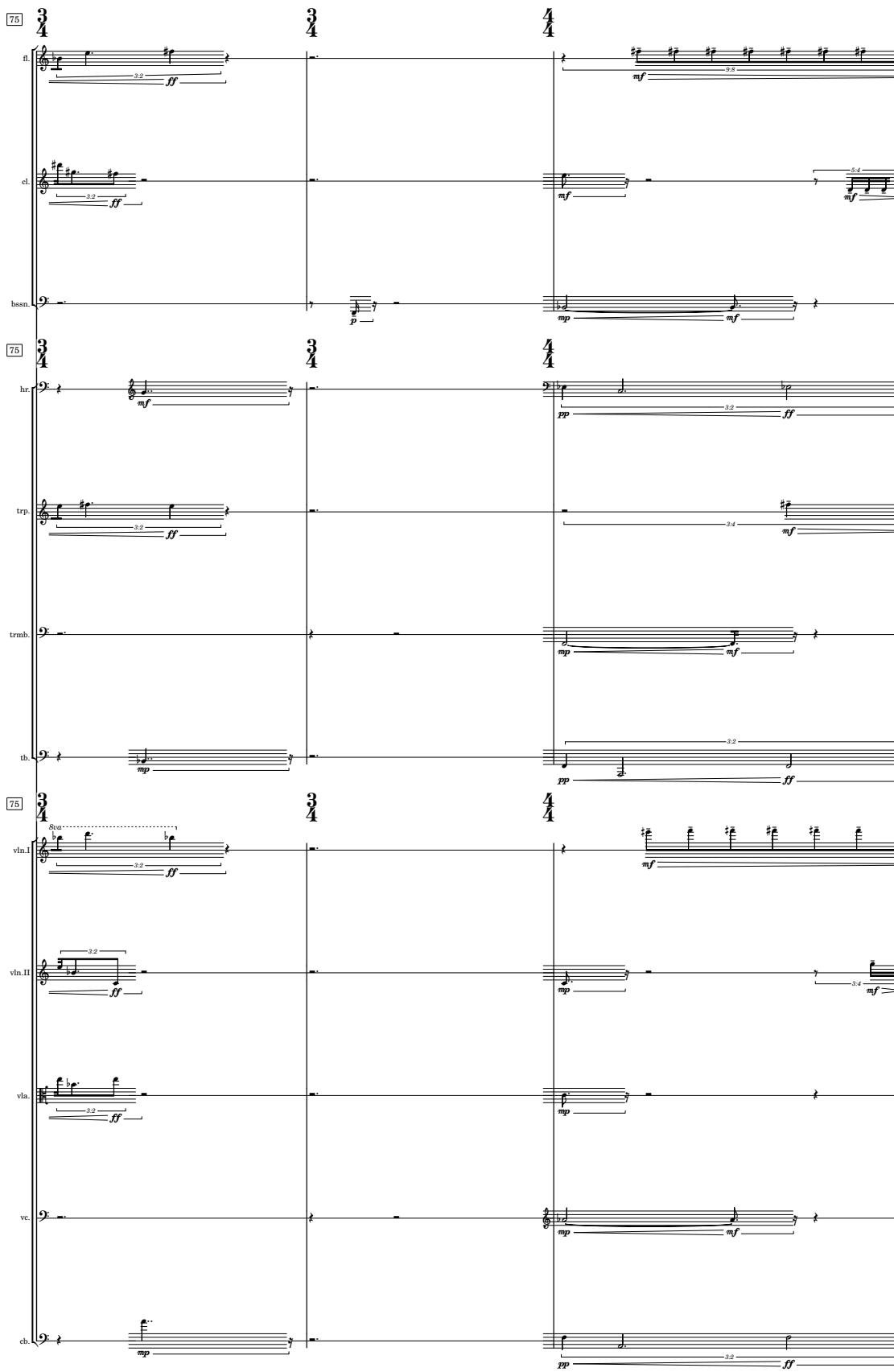
165 

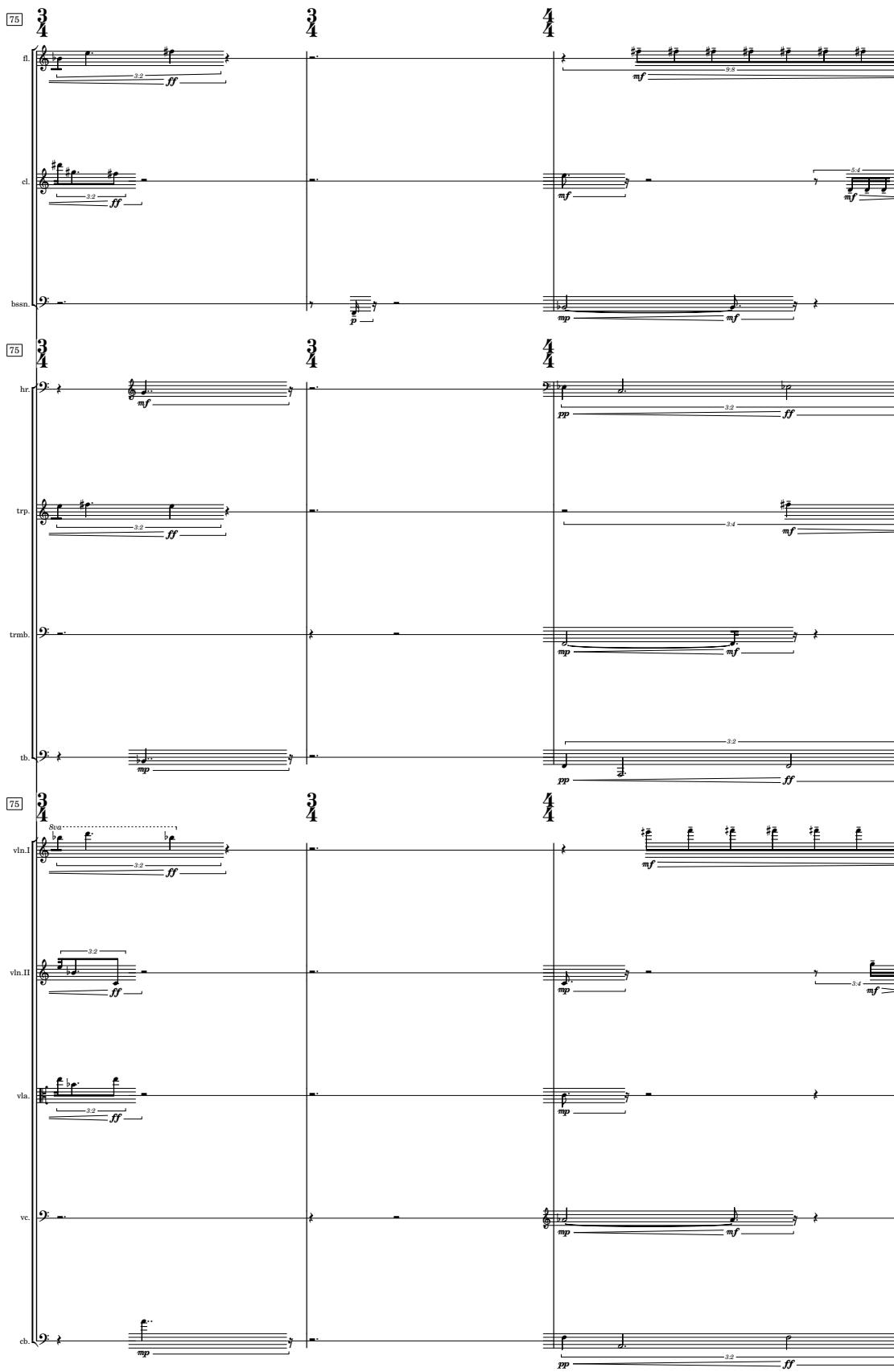
166 

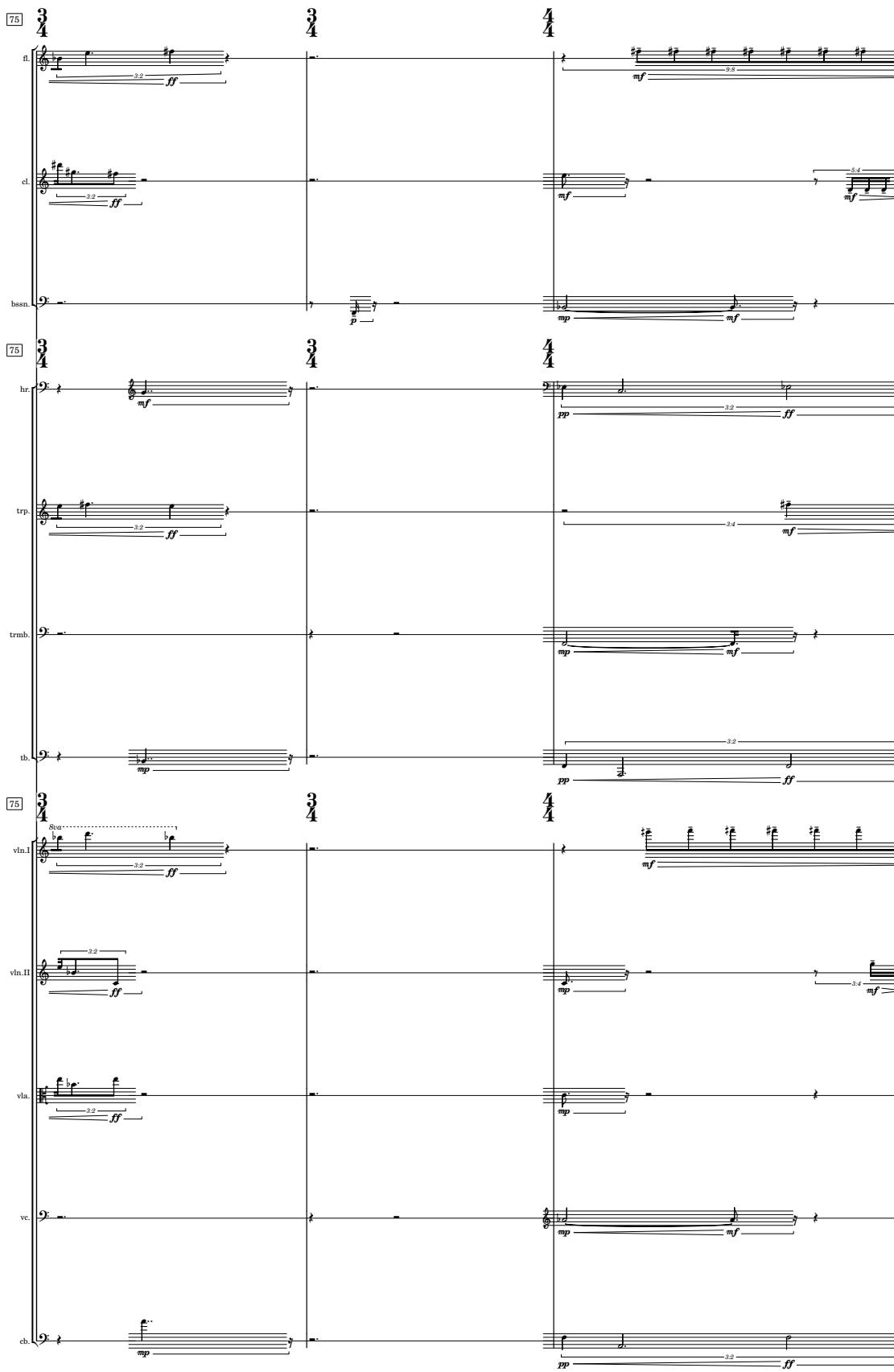
167 

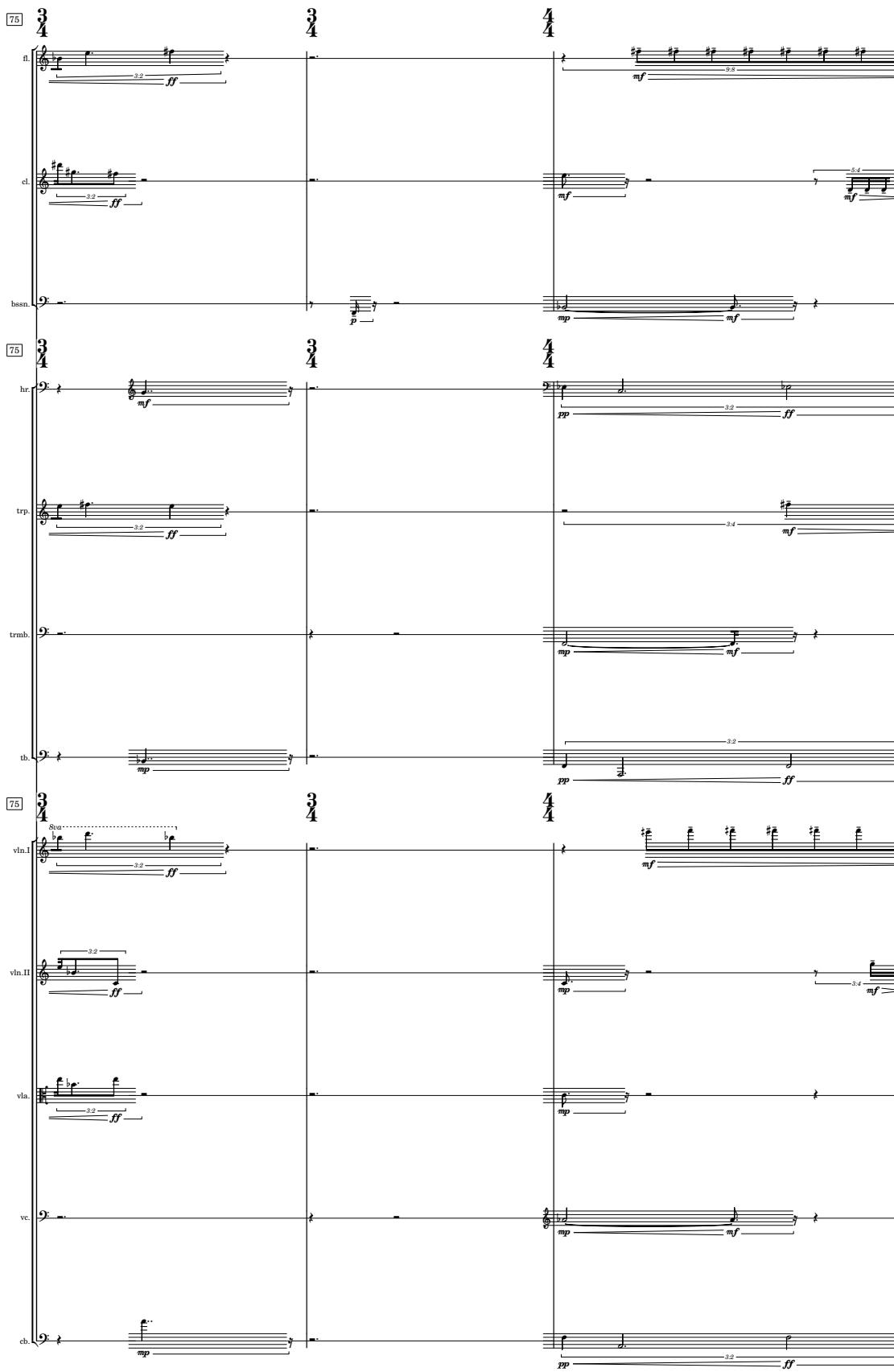
168 

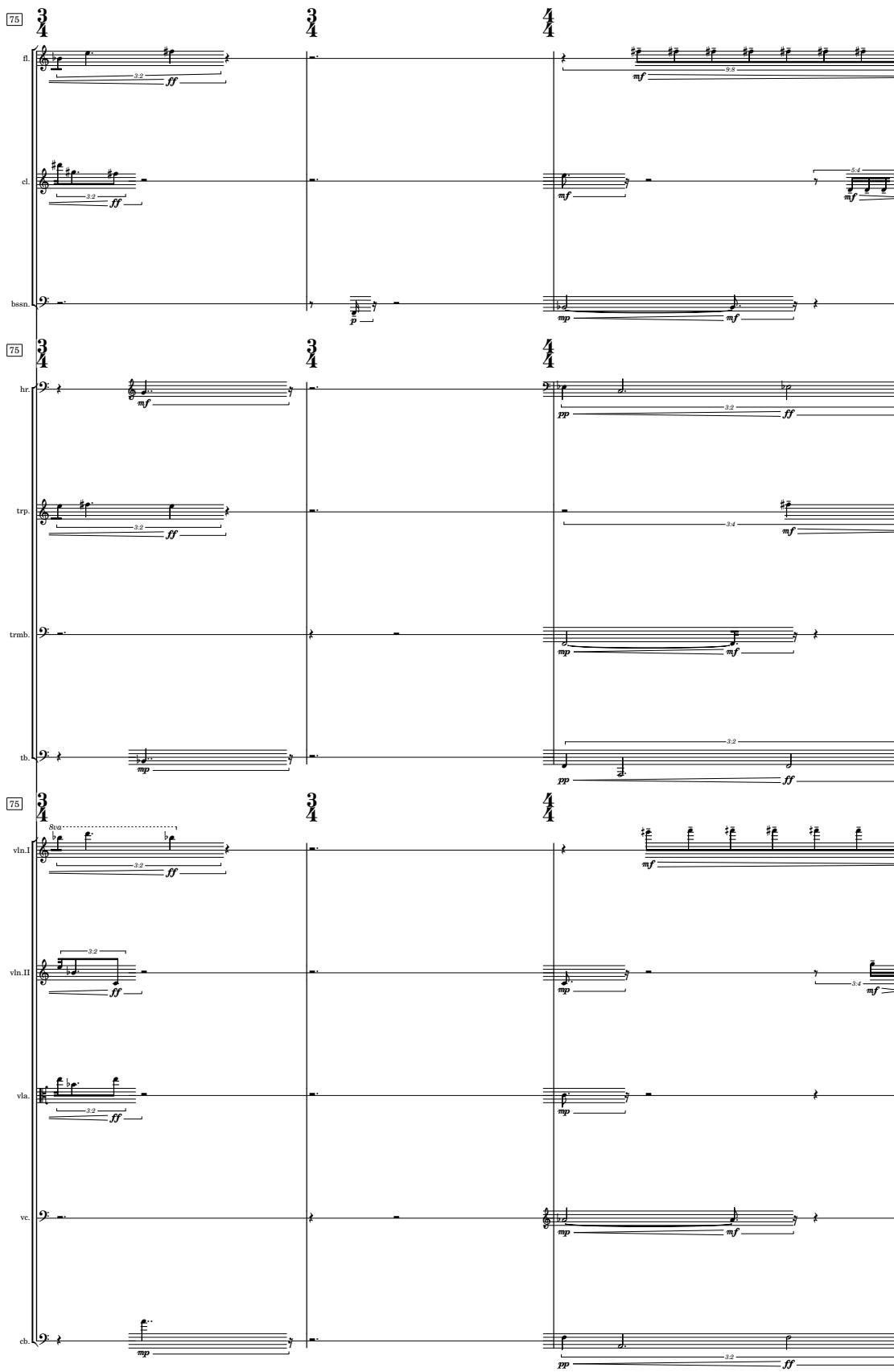
169 

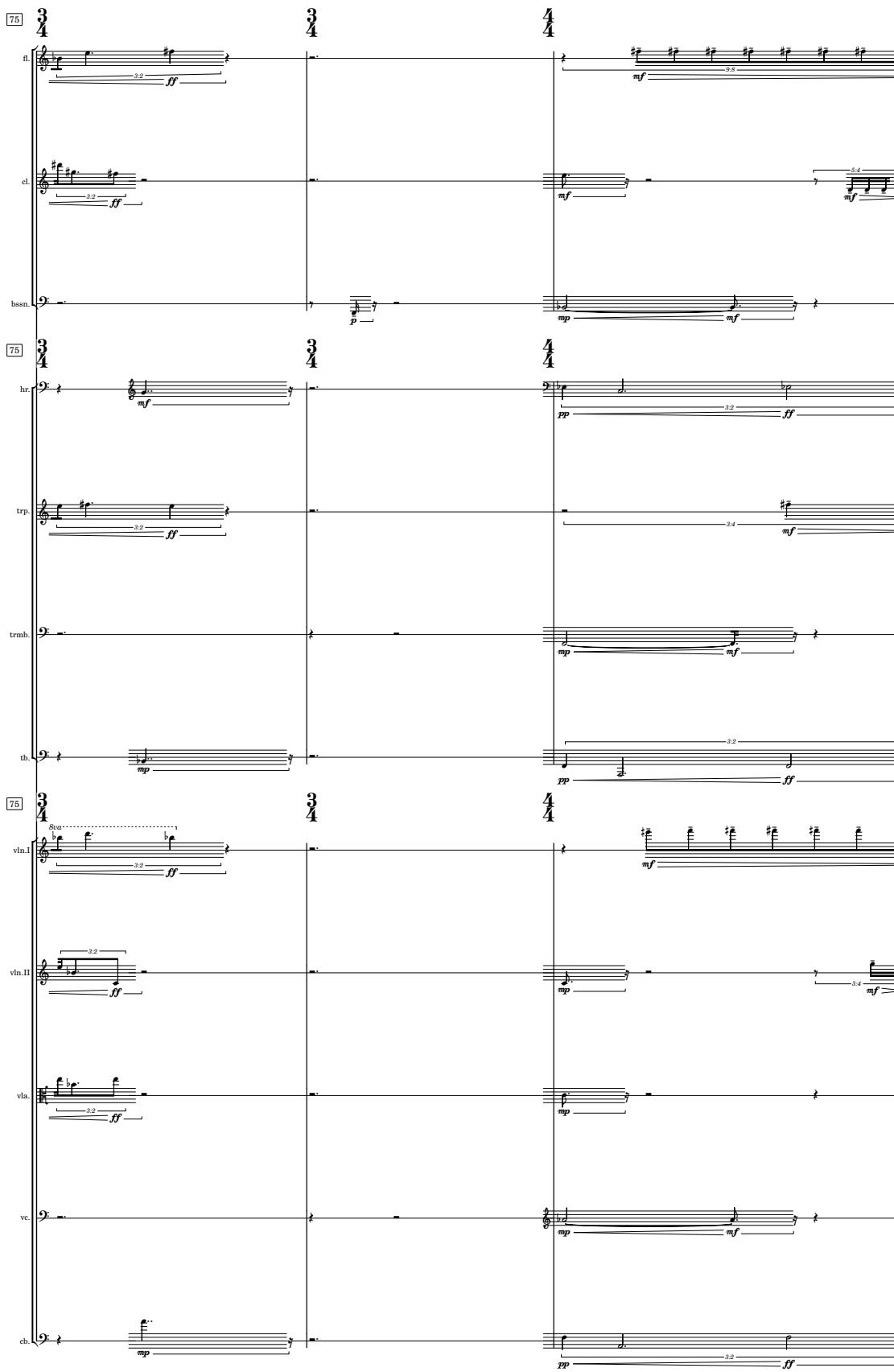
170 

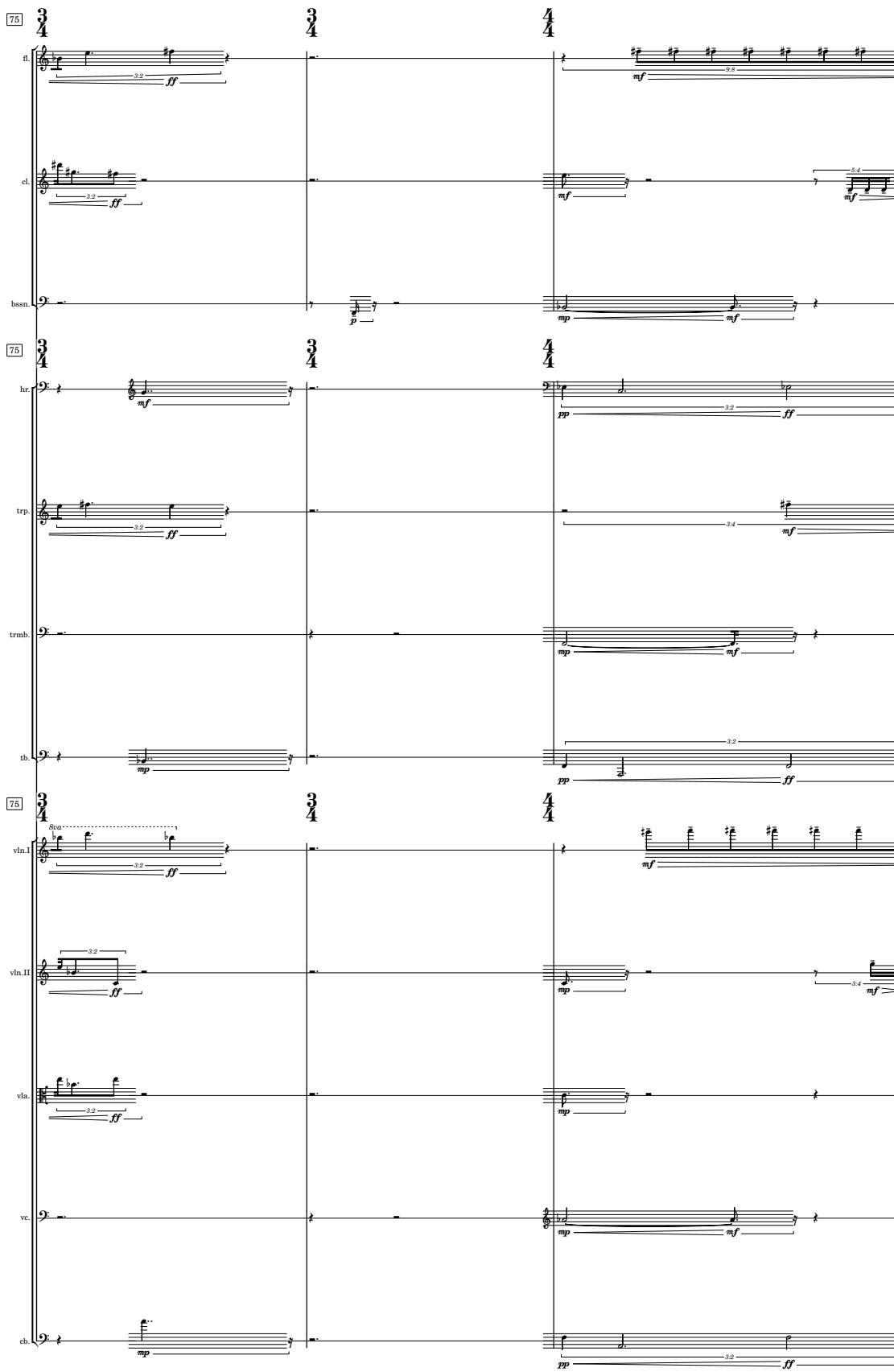
171 

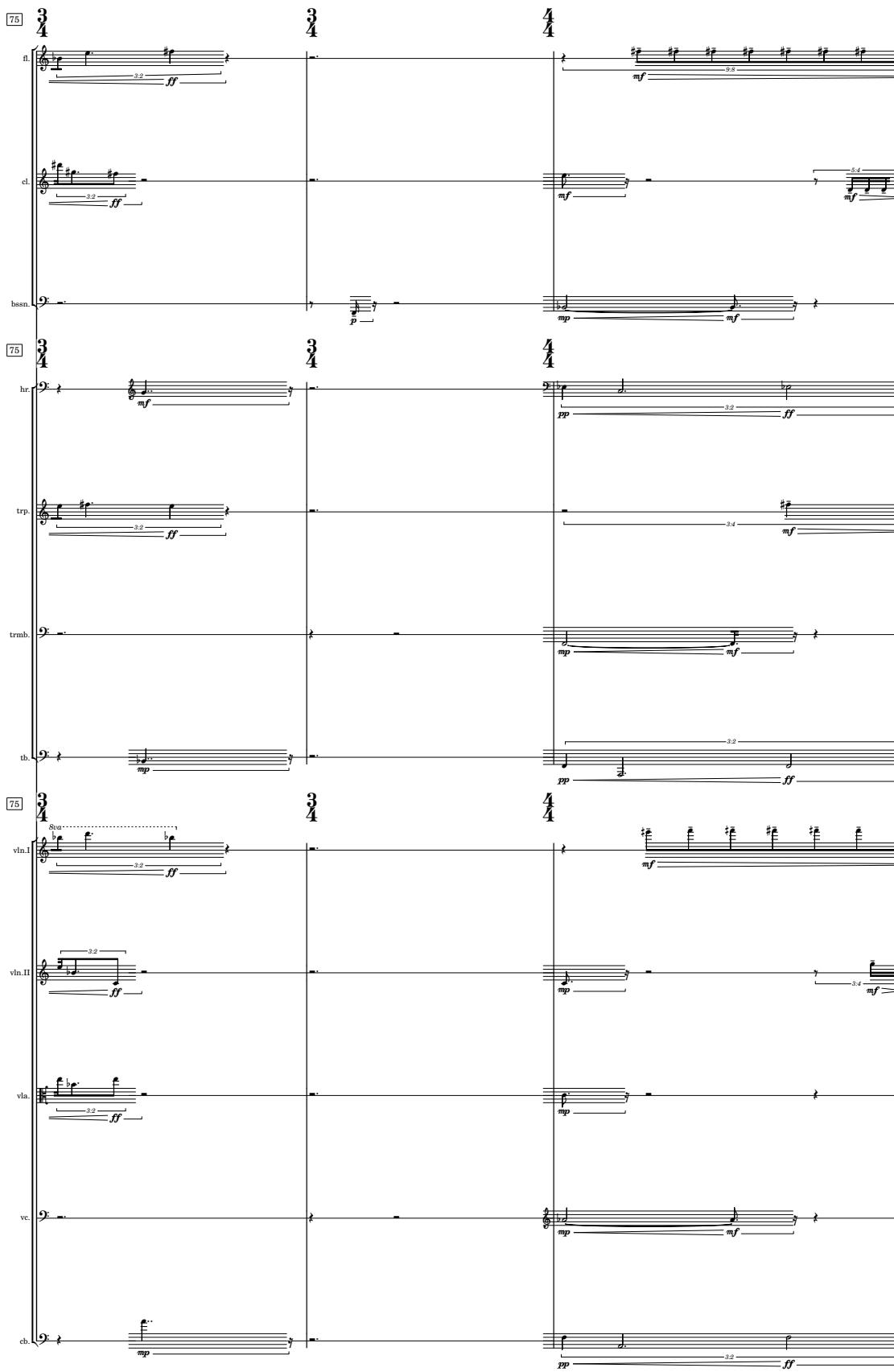
172 

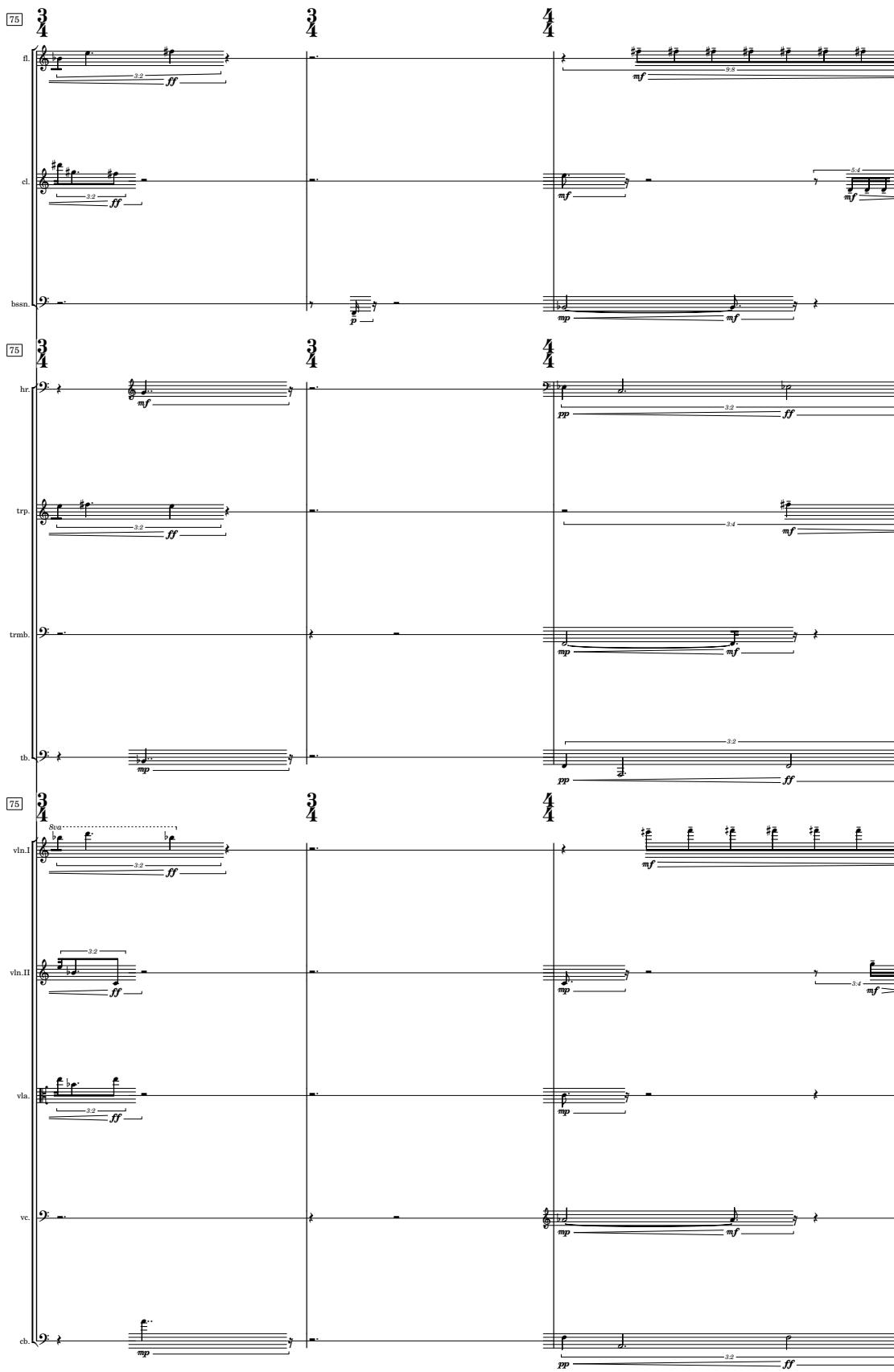
173 

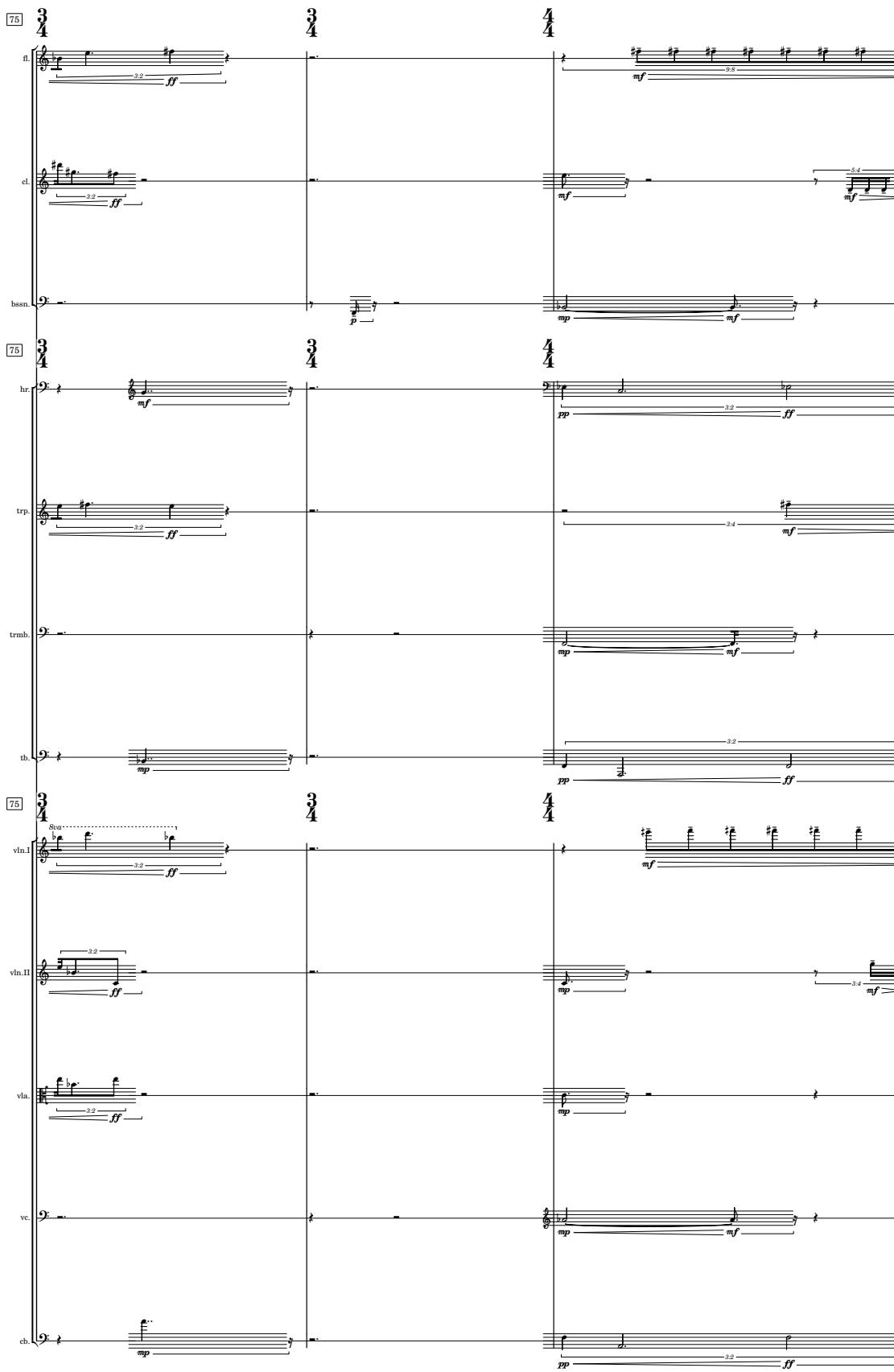
174 

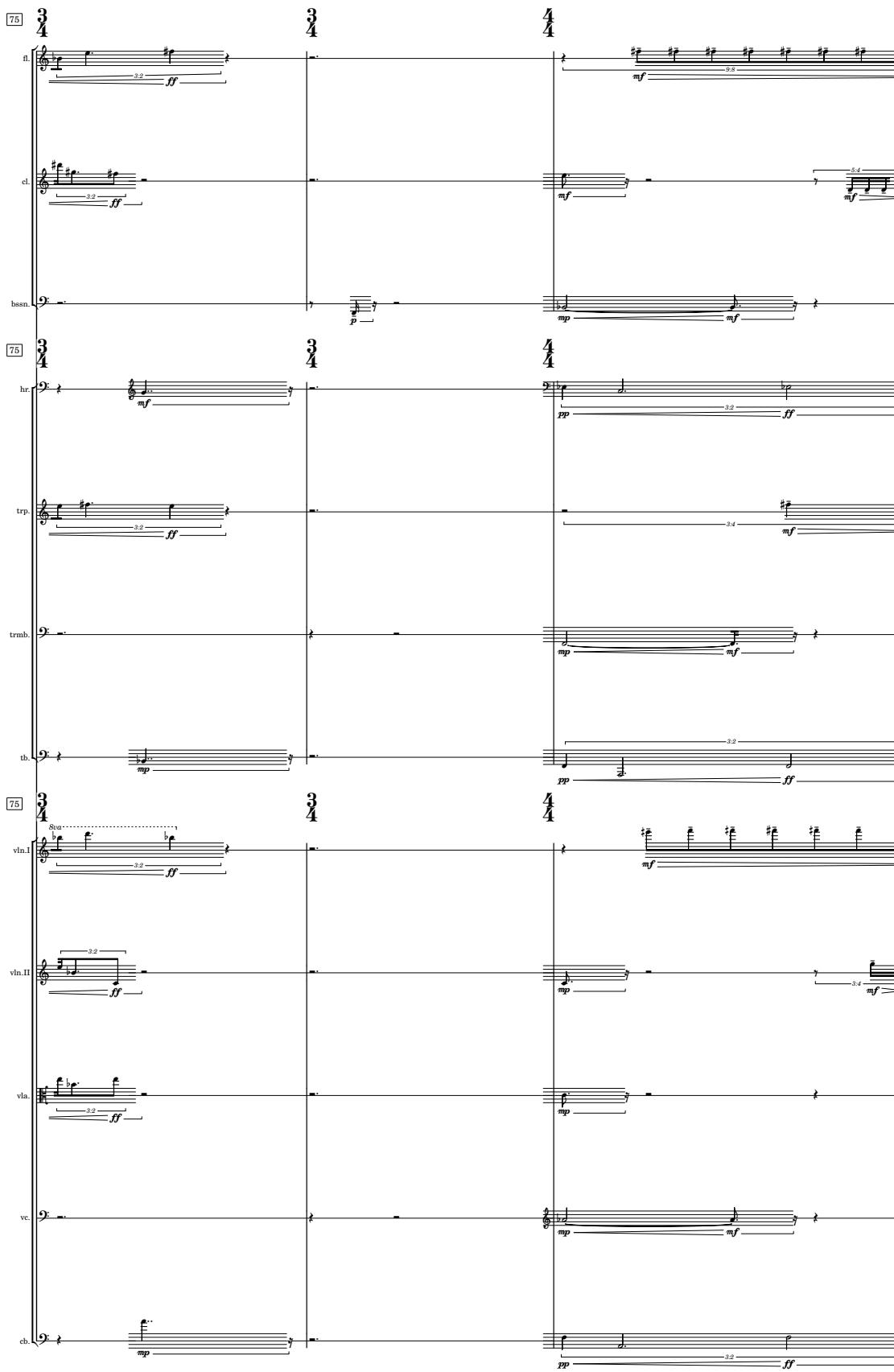
175 

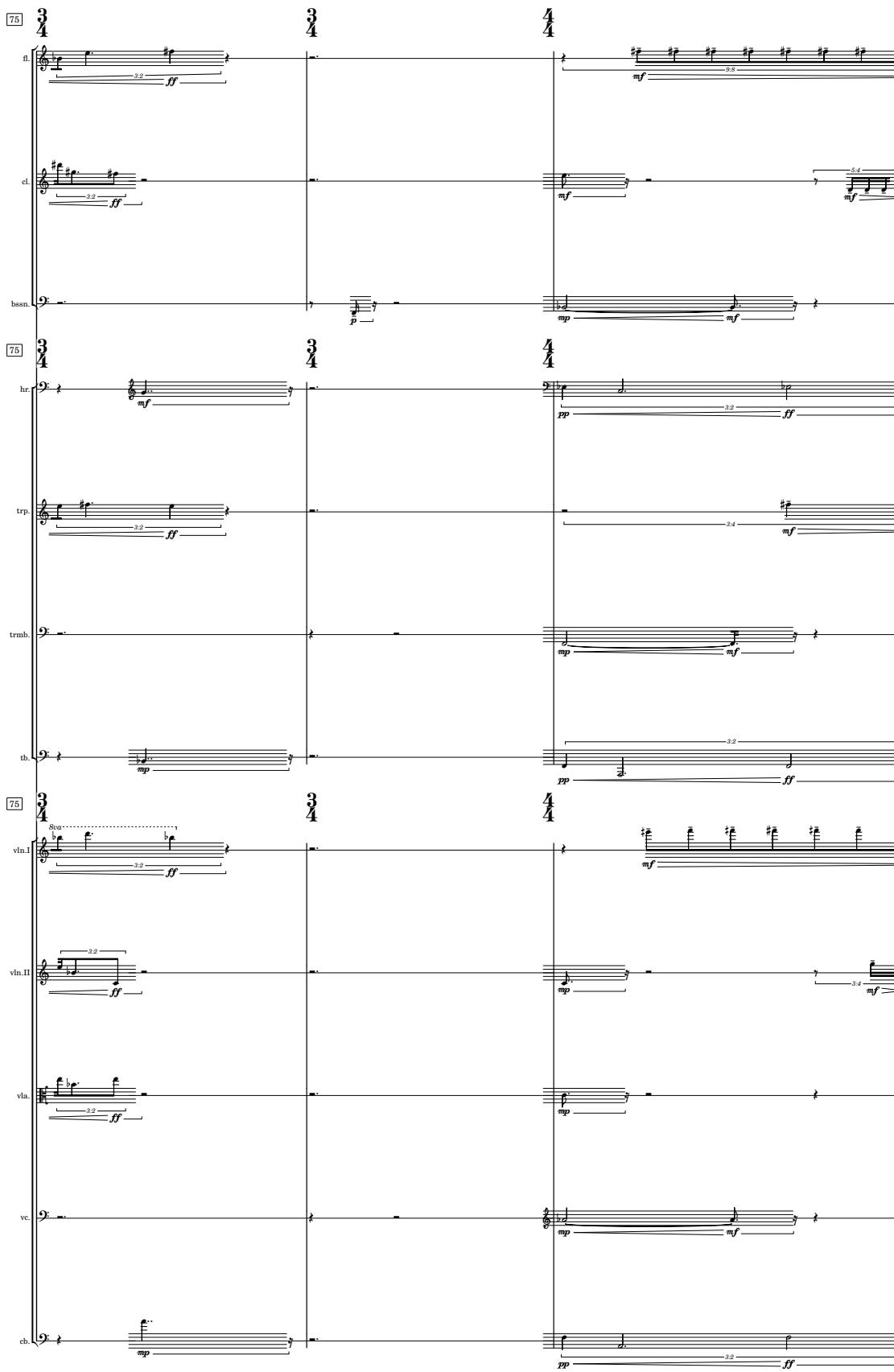
176 

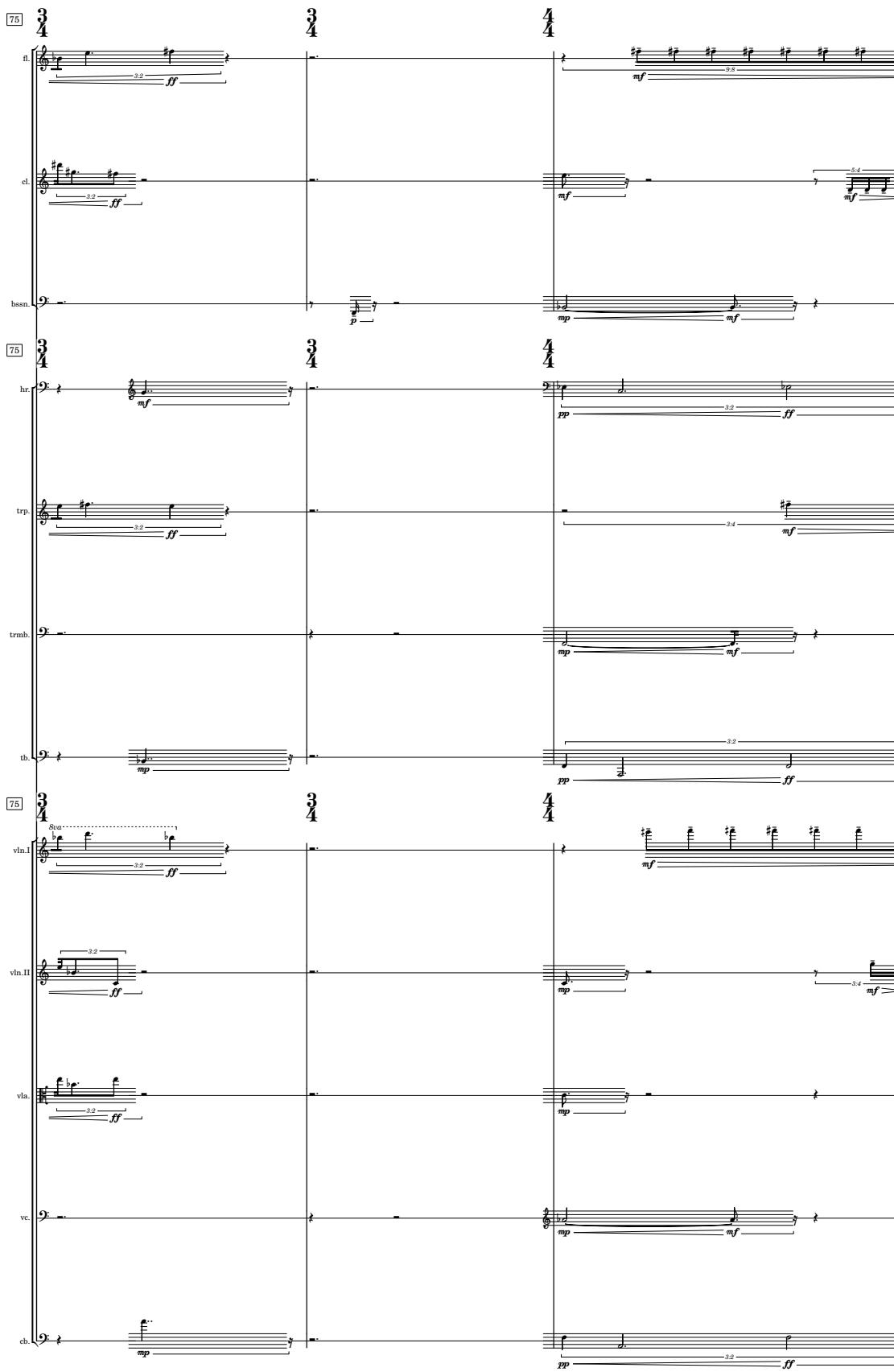
177 

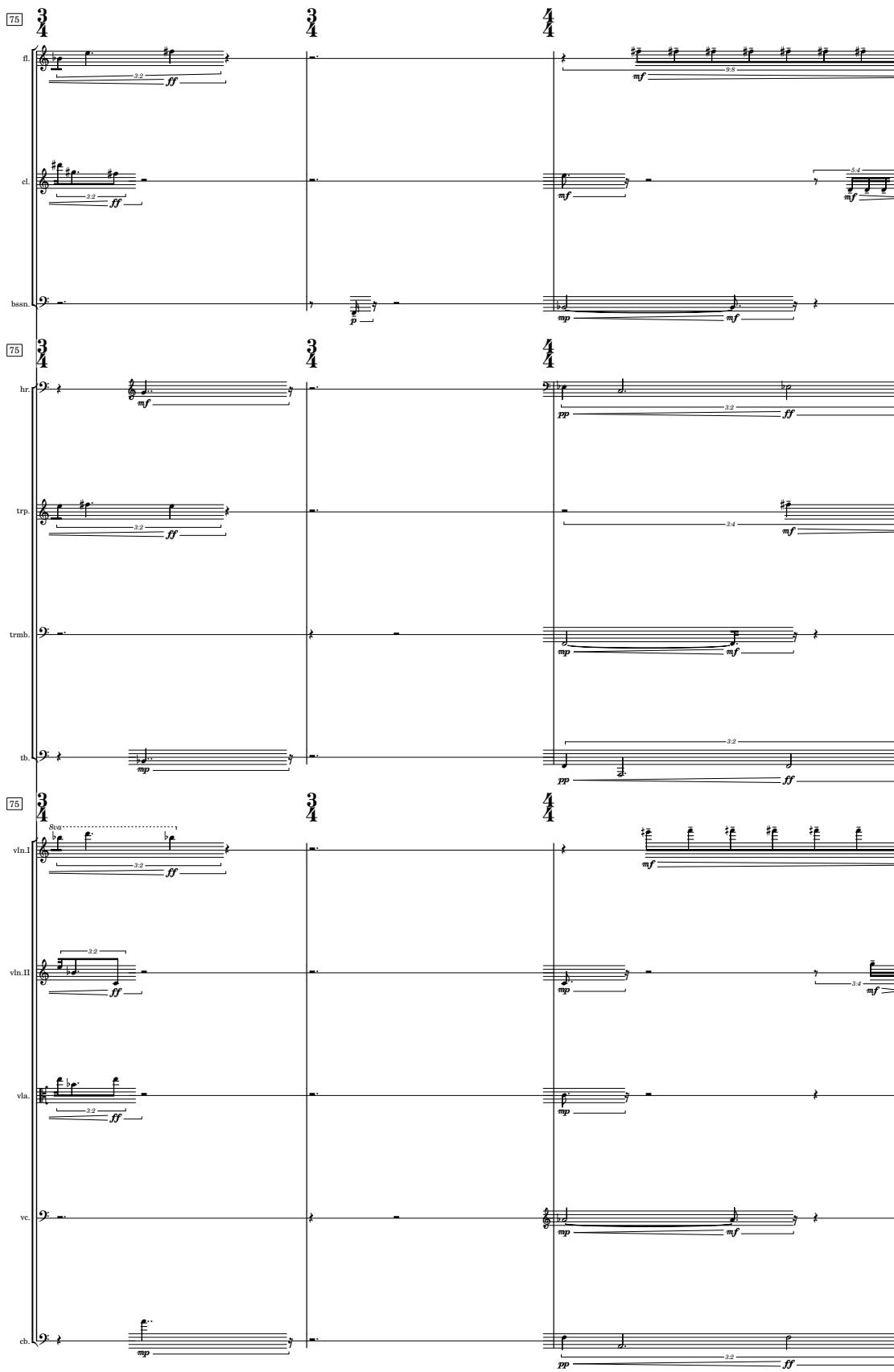
178 

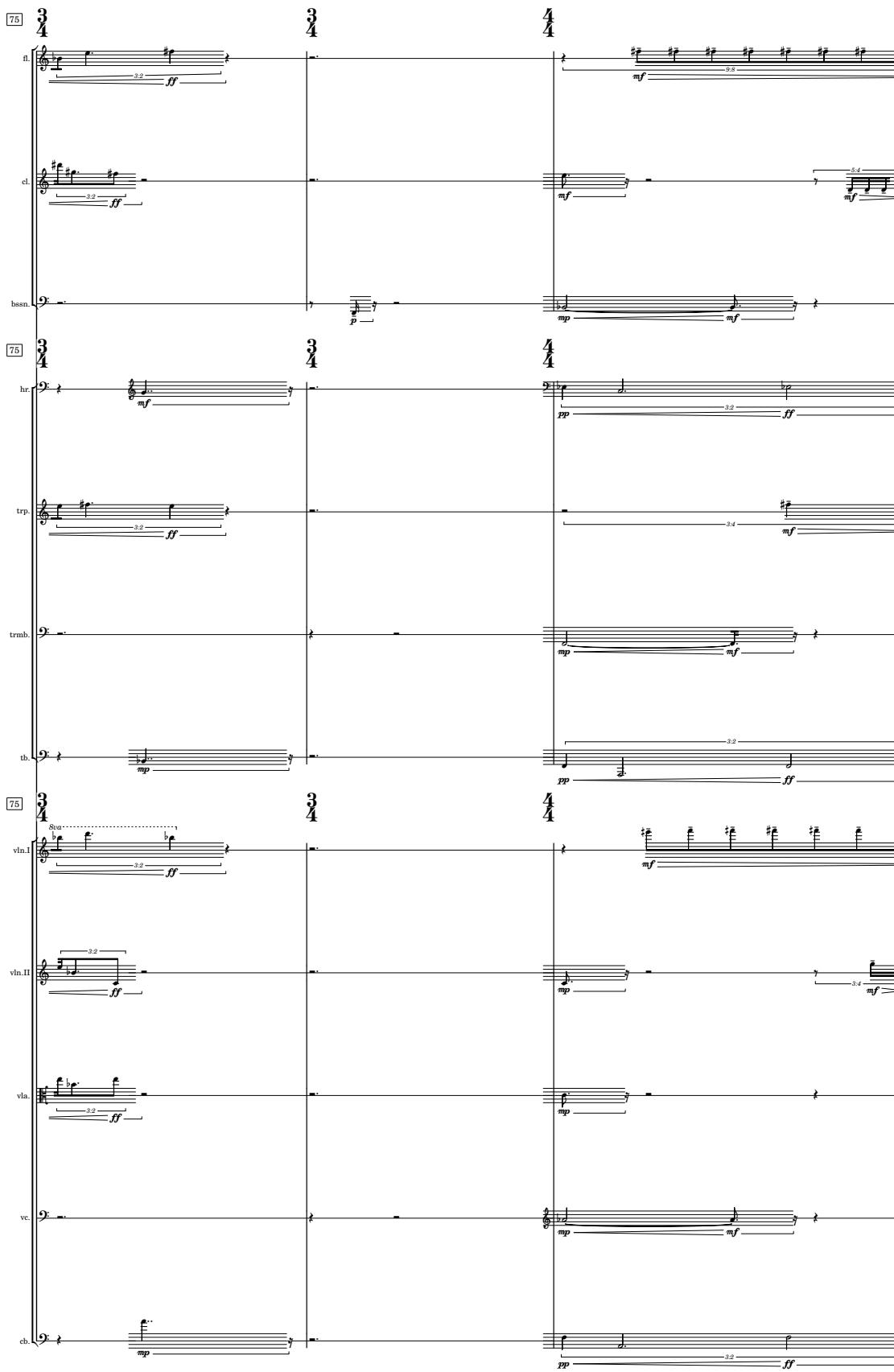
179 

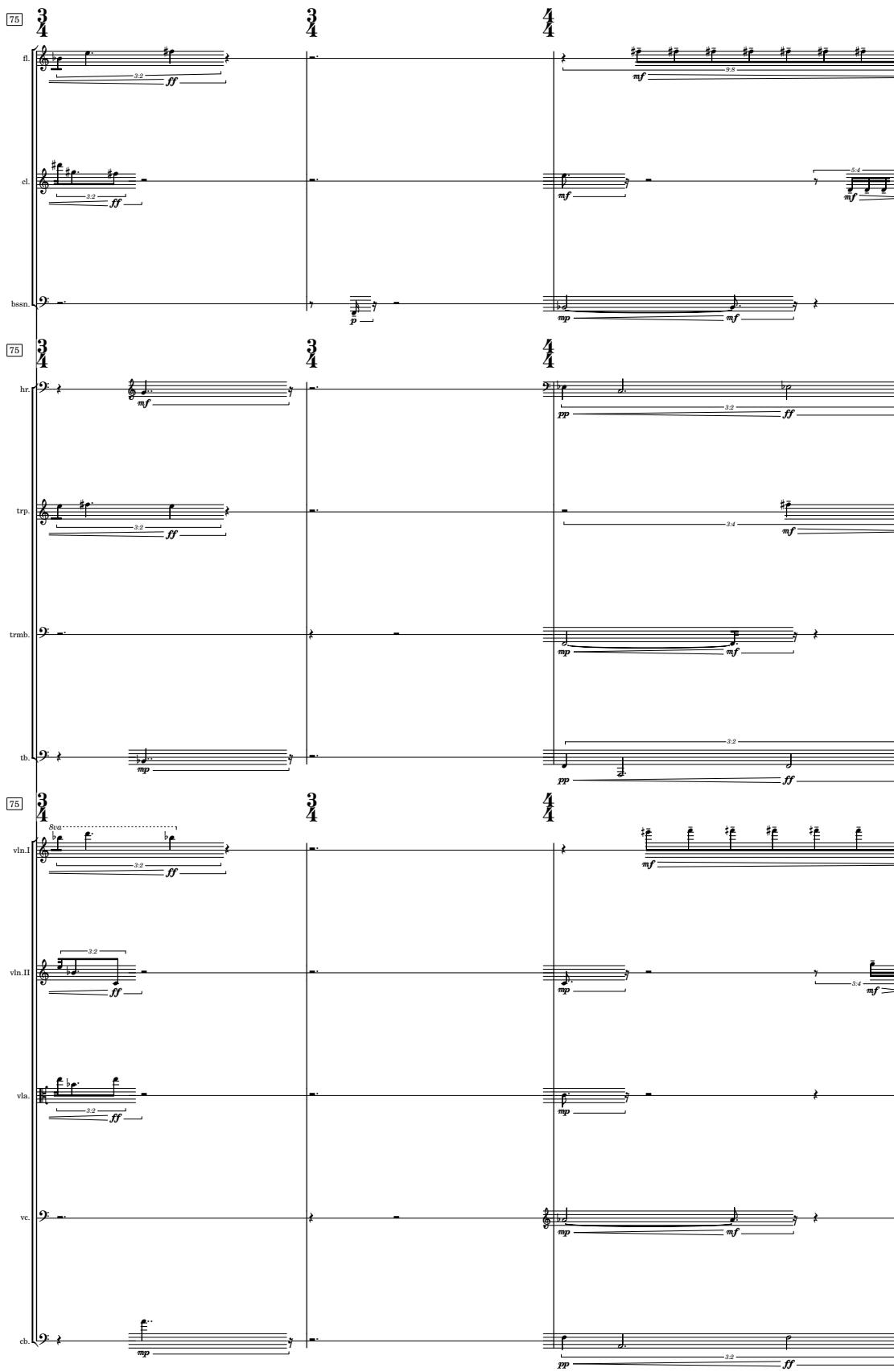
180 

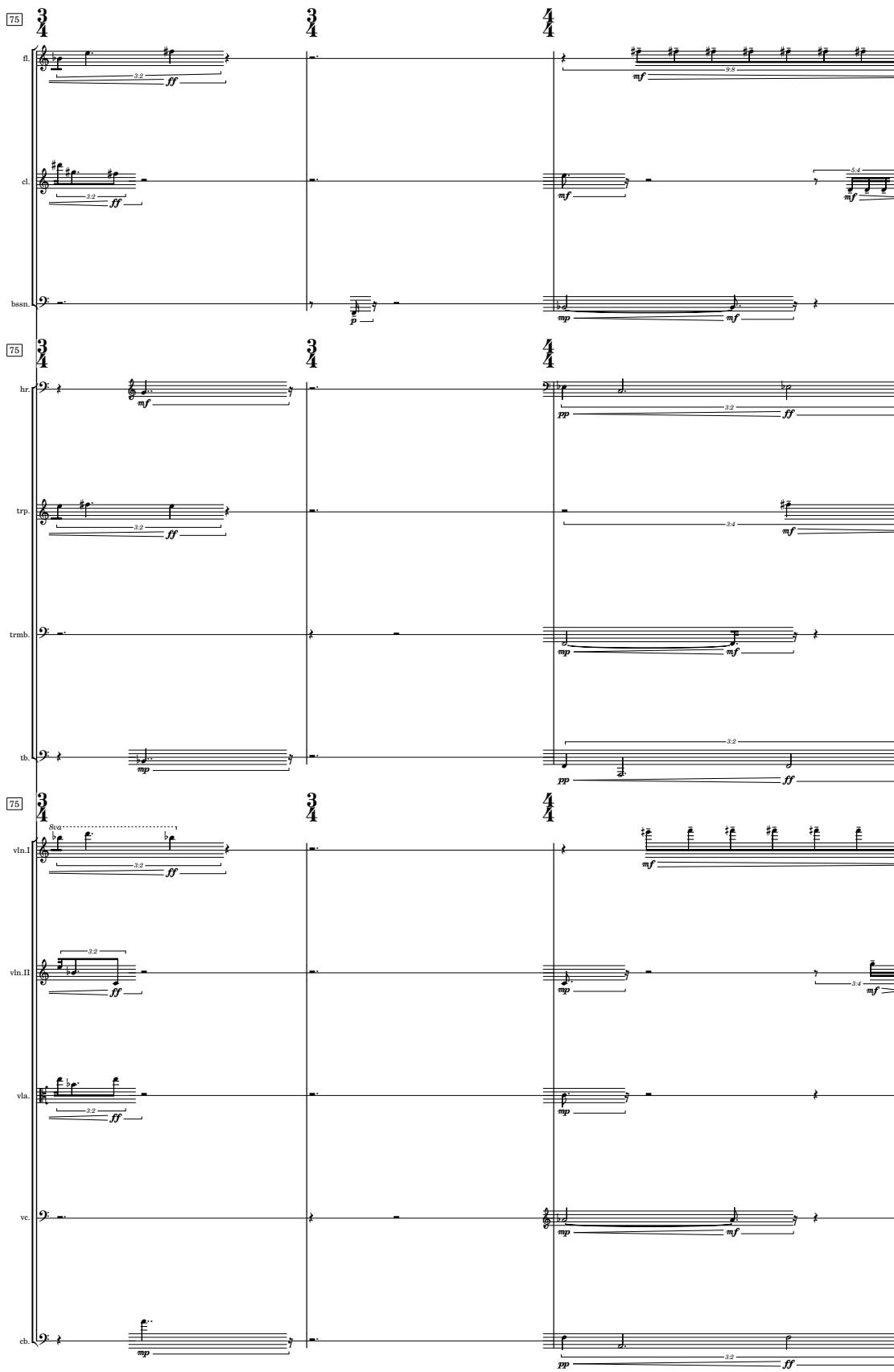
181 

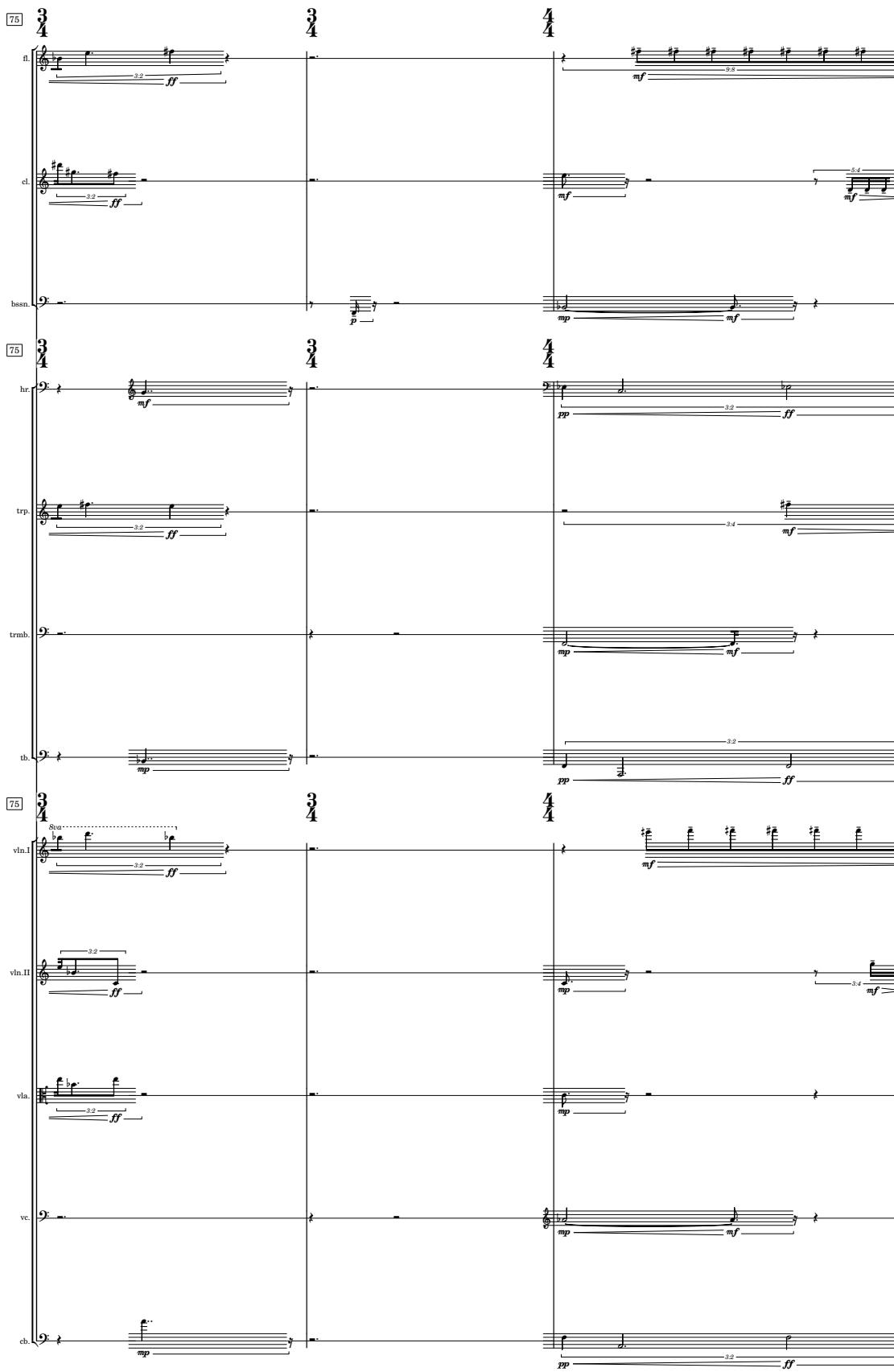
182 

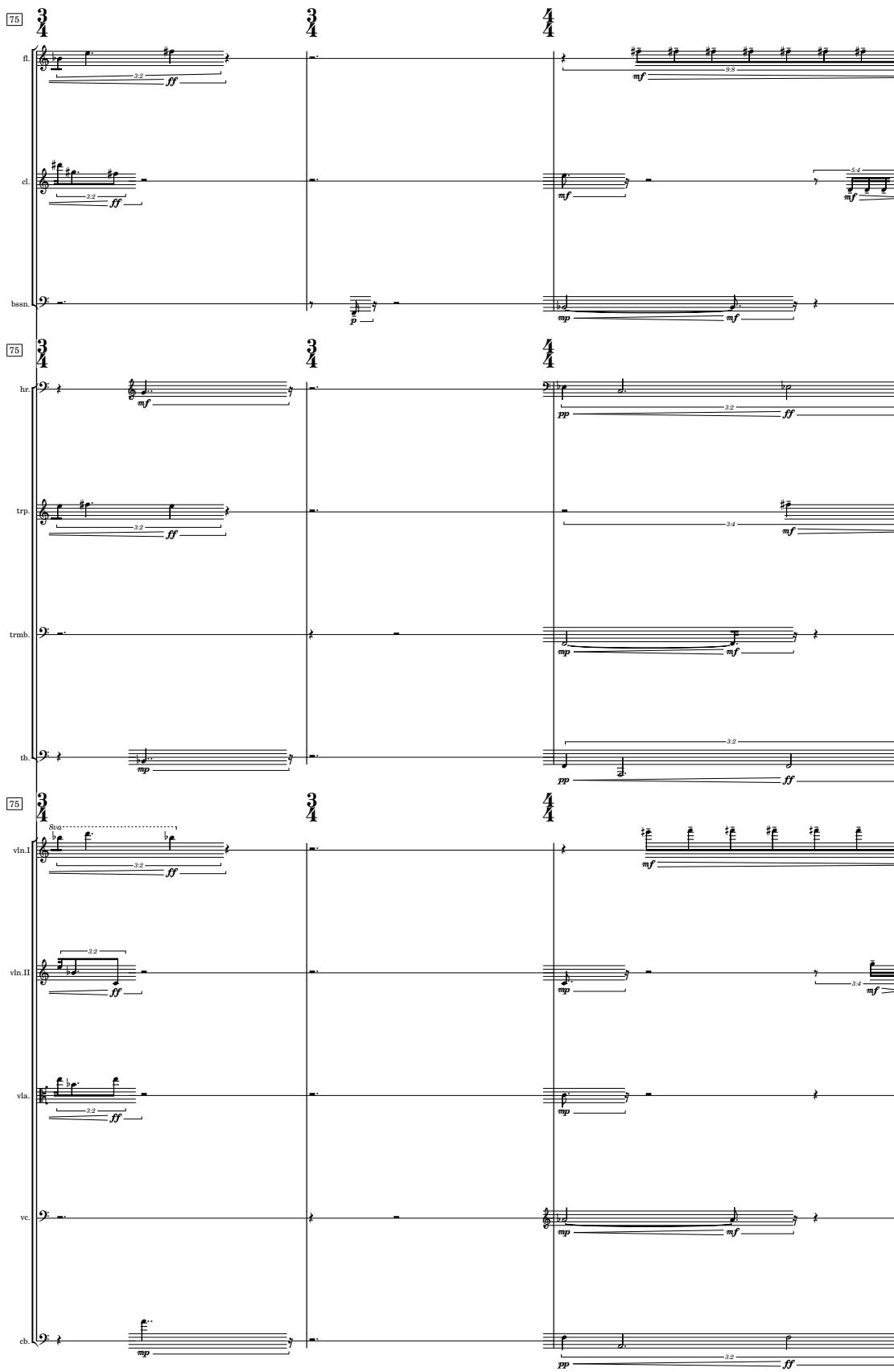
183 

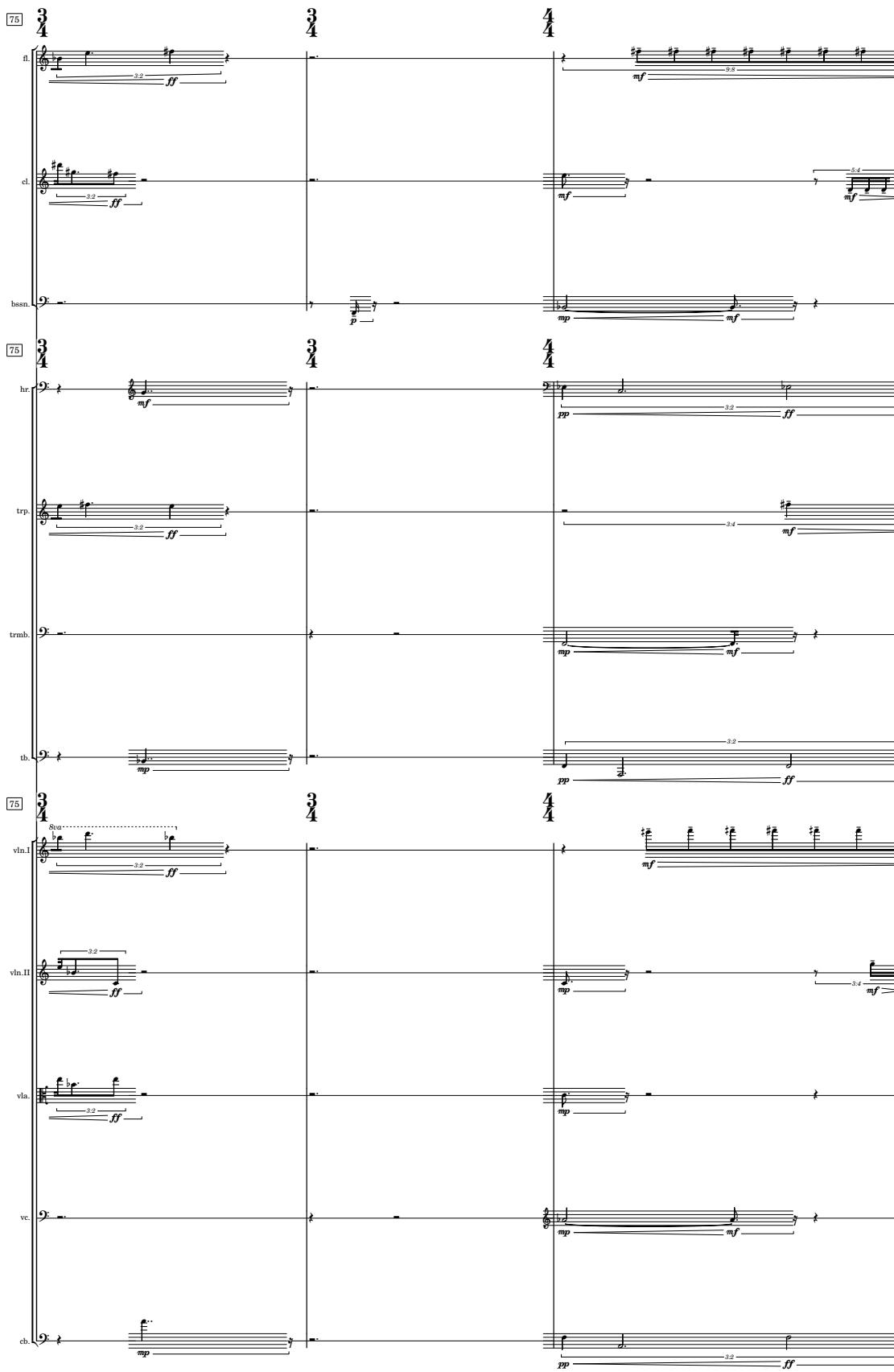
184 

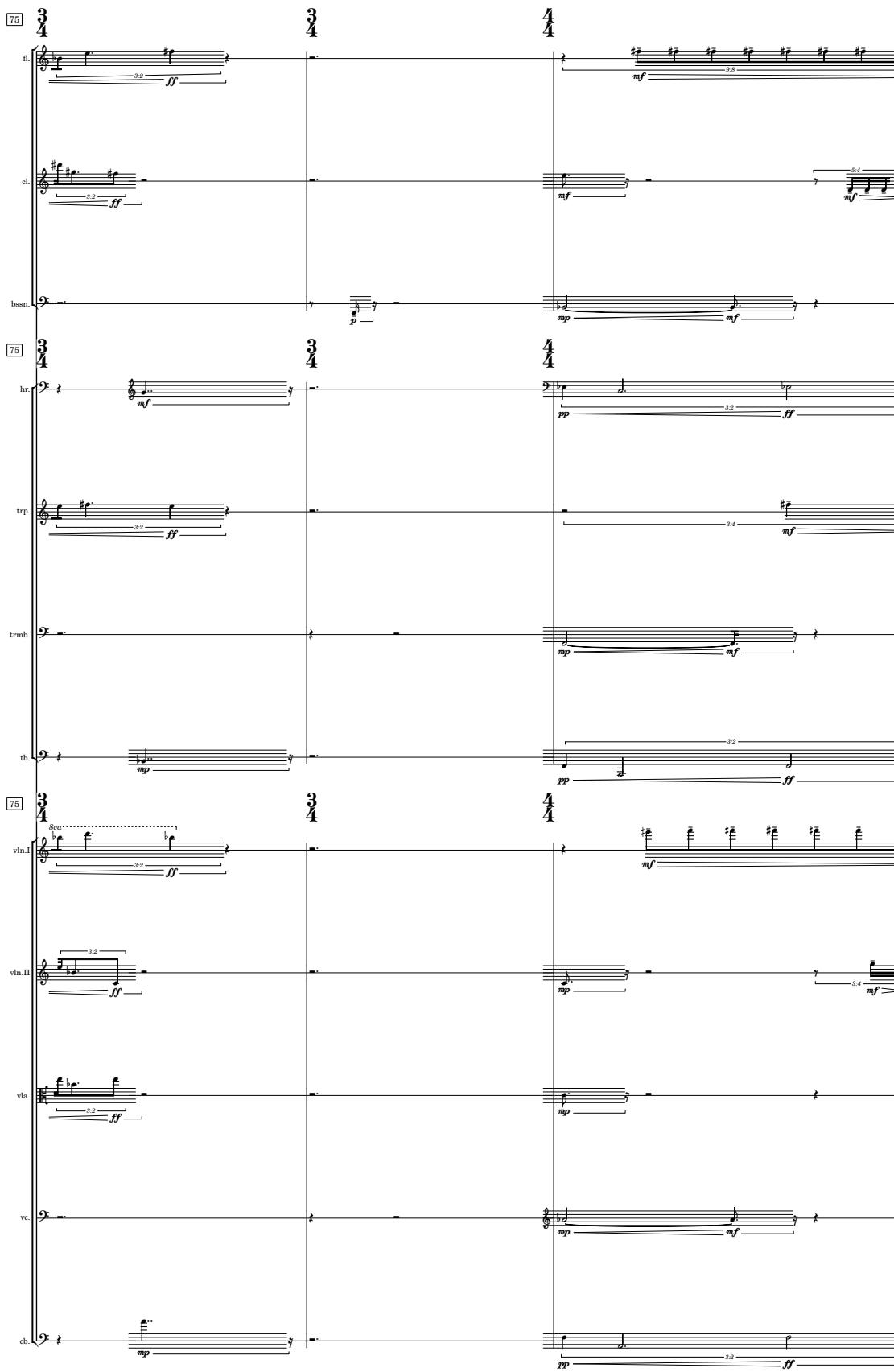
185 

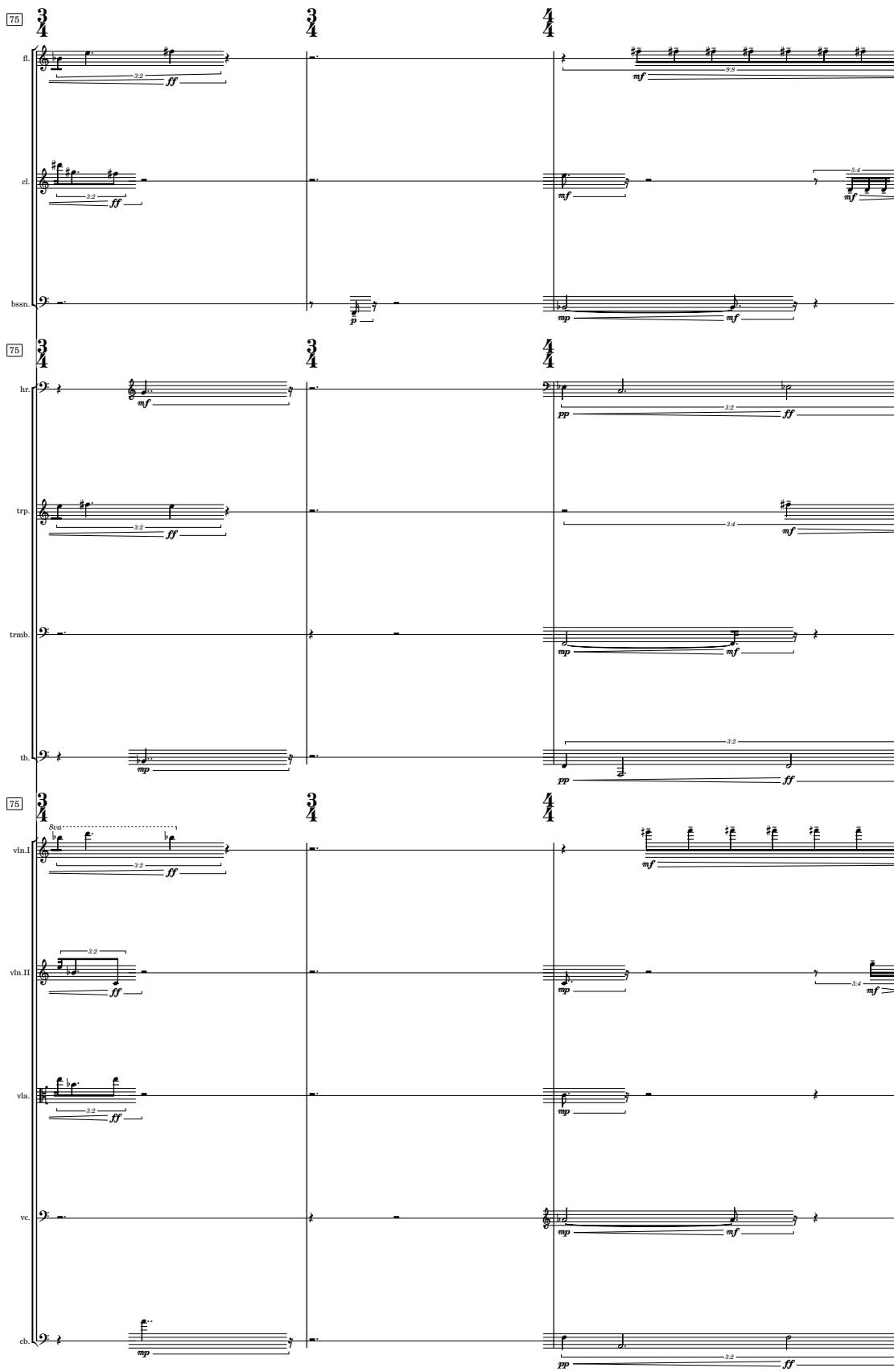
186 

187 

188 

189 

190 

191 

192 <img alt="Measure 192: Flute, Clarinet, Bassoon, Horn, Trumpet, Trombone, Tromba, Trombone, Violin I, Violin II, Viola, Cello. Measures 193-194: Violin I, Violin II, Viola, Cello." data-bbox="1

78  $\frac{3}{4}$   $\frac{3}{4}(\textcircled{J})$   $\frac{4}{4}$

fl.  $p$   $mf$   $mp$   $mf$   $p$

cl.  $p$   $3.2$   $mf$   $p$

bass.  $9$   $pp$   $ff$   $mf$   $p$   $3.4$   $3.4$   $mf$

78  $\frac{3}{4}$   $\frac{3}{4}(\textcircled{J})$   $\frac{4}{4}$

hr.  $9$   $mp$   $mf$

trp.  $p$   $mf$   $mp$   $mf$   $7.6$

trmb.  $9$   $pp$   $ff$   $mf$   $p$

th.  $9$   $mf$   $p$   $3.4$   $3.4$   $mf$

78  $\frac{3}{4}$   $\frac{3}{4}(\textcircled{J})$   $\frac{4}{4}$

vln.I  $p$   $8.021$   $mf$   $mf$   $12.12$

vln.II  $p$   $mf$   $p$   $9.8$

vla.  $mf$   $p$   $12.16$

vc.  $pp$   $ff$   $mf$   $p$   $9.8$   $3.2$   $mf$

cb.  $9$   $mf$

[81] **5** **4** **3**

f.  $\text{mf} \xrightarrow[p]{}$   $\text{mf} \xrightarrow[7.6]{}$   $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$

cl.  $\text{mf} \xrightarrow[7.8]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[7.6]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[\#f]{}$

bassn.  $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[7.6]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$

[81] **5** **4** **3**

hr.  $\text{p} \xrightarrow[3.2]{}$   $\text{mf} \xrightarrow[7.8]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[9.8]{}$   $\text{p} \xrightarrow[5.4]{}$

trp.  $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[9.8]{}$   $\text{p} \xrightarrow[5.4]{}$

trmb.  $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$

th.  $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[7.8]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$

[81] **5** **4** **3**

vin.I  $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[11.12]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[9.8]{}$   $\text{p} \xrightarrow[3.4]{}$

vin.II  $\text{p} \xrightarrow[17.16]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$

vla.  $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[7.8]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$

vc.  $\text{p} \xrightarrow[3.4]{}$   $\text{mf} \xrightarrow[5.4]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[9.8]{}$   $\text{p} \xrightarrow[5.4]{}$

ch.  $\text{p} \xrightarrow[9.8]{}$   $\text{mf} \xrightarrow[9.8]{}$   $\text{p} \xrightarrow[5.4]{}$   $\text{mf} \xrightarrow[3.4]{}$   $\text{p} \xrightarrow[5.4]{}$

84  $\frac{5}{4}$  (K)  $\frac{5}{4}$

fl.  $\frac{5}{4}$   $\frac{5}{4}$

cl.  $\frac{5}{4}$   $\frac{5}{4}$

bassn.  $\frac{5}{4}$   $\frac{5}{4}$

hr.  $\frac{5}{4}$   $\frac{5}{4}$

trp.  $\frac{5}{4}$   $\frac{5}{4}$

trmb.  $\frac{5}{4}$   $\frac{5}{4}$

th.  $\frac{5}{4}$   $\frac{5}{4}$

vln.I  $\frac{5}{4}$   $\frac{5}{4}$

vln.II  $\frac{5}{4}$   $\frac{5}{4}$

vla.  $\frac{5}{4}$   $\frac{5}{4}$

vc.  $\frac{5}{4}$   $\frac{5}{4}$

cb.  $\frac{5}{4}$   $\frac{5}{4}$

[86]  $\frac{5}{4}$

fl.  $\frac{5}{4}$   
cl.  
bassn.

[86]  $\frac{5}{4}$   
hr.  $\frac{5}{4}$   
trp.  $\frac{5}{4}$   
trmb.  $\frac{5}{4}$   
th.  $\frac{5}{4}$

vln.I  $\frac{5}{4}$   
vln.II  $\frac{5}{4}$   
vla.  $\frac{5}{4}$   
vc.  $\frac{5}{4}$   
cb.  $\frac{5}{4}$

88

f fl.

cl.

bassn.

88

hr.

trp.

trmb.

tb.

88

vln.I

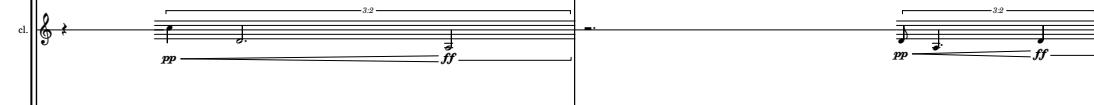
vln.II

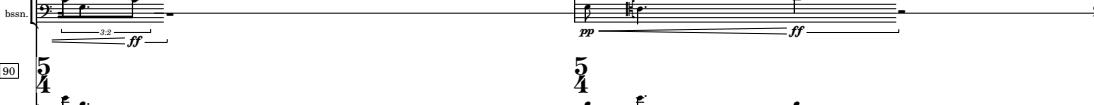
vla.

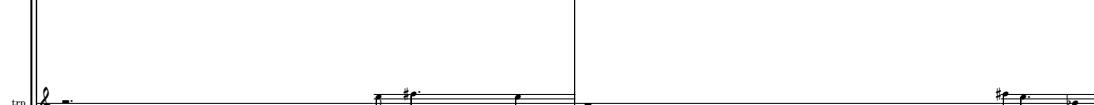
vc.

cb.

90 

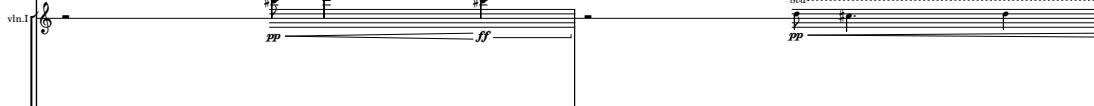
90 

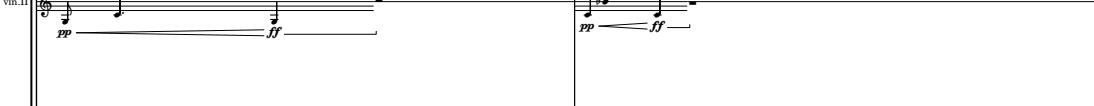
90 

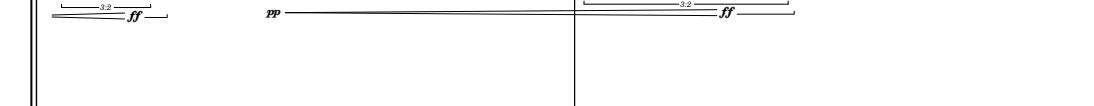
90 

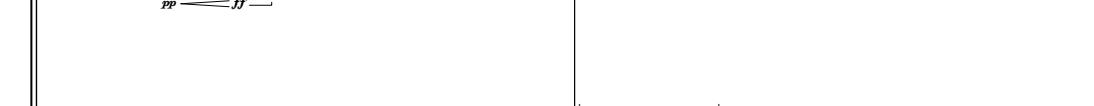
90 

90 

90 

90 

90 

90 

-34-

92

fl.

cl.

bassn.

92

hr.

trp.

trmb.

tb.

vln.I

vln.II

vla.

vc.

cb.

95

fl.

cl.

bsns.

95

hr.

trp.

trmb.

th.

vln.I

vln.II

vla.

vc.

ch.

96

97

98

98 **5**

f. *p* —————— 3.2 —————— **ff** ——————

**5**  $\text{♩} = 60$

cl. *#f* —————— 3.2 —————— **ff** —————— *mp* —————— 13.12 —————— **f** ——————

bassn. *f* —————— 3.2 —————— **ff** —————— *mp* ——————

**2**

98 **5**

hr. *f* —————— 3.2 —————— **ff** —————— *mp* —————— 19.20 —————— **f** ——————

trp. *#f* —————— 3.2 —————— **ff** ——————

trmb. *f* —————— 3.2 —————— **ff** ——————

th. —————— 3.2 —————— **ff** —————— *mp* —————— **f** ——————

**5**

vln.I *f* —————— 3.2 —————— **ff** —————— *p* ——————

**2**

vln.II *#f* —————— 3.2 —————— **ff** —————— *mp* —————— 11.12 —————— **f** ——————

vla. *f* —————— 3.2 —————— **ff** —————— *mp* —————— **f** ——————

vc. *f* —————— 3.2 —————— **ff** ——————

ch. *f* —————— 3.2 —————— **ff** —————— *mp* —————— **f** ——————

101

fl.

cl.

bassn.

hr.

trp.

trmb.

th.

vln.I

vln.II

vla.

ve.

ch.

Measure 1 (4/4): fl. (f), cl. (mp), bassn. (mp), hr. (ff), trp. (f), trmb. (f), th. (mp), vln.I (ff), vln.II (mp), vla. (mp), ve. (ff), ch. (mf). Measures 2-3 (3/4): fl. (ff), cl. (mf), bassn. (mf), hr. (ff), trp. (ff), trmb. (ff), th. (ff), vln.I (ff), vln.II (mf), vla. (mf), ve. (ff), ch. (mf).

[104]

f fl. -

cl. -

bassn. -

[104]

hr. -

trp. -

trmb. -

th. -

[104]

vln. I -

vln. II -

vla. -

vc. -

cb. -

[107]  

f.  
cl.  
bassn.

[107]  

hr.  
trp.  
trmb.

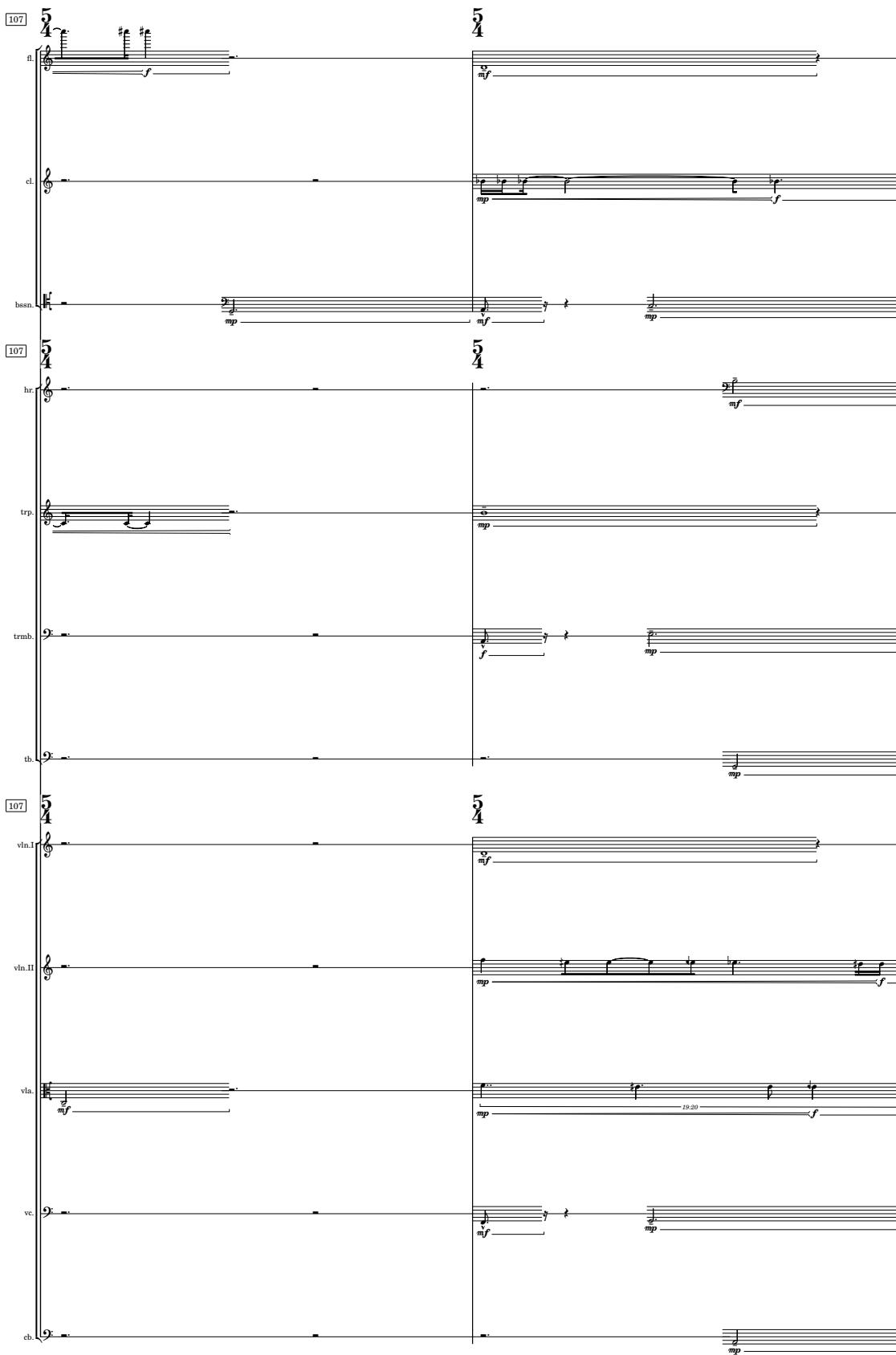
th.

[107]  

vln.I  
vln.II

vla.  
ve.

ch.



[109]  $\frac{3}{4}$        $\frac{3}{4}$        $\frac{4}{4}$

fl.  $\frac{mf}{f}$

cl.

bassn.  $\frac{mf}{f}$

[109]  $\frac{3}{4}$        $\frac{3}{4}$        $\frac{4}{4}$

hr.  $\frac{mp}{f}$   $\frac{f}{11.12}$

trp.  $\frac{f}{mf}$   $\frac{mp}{f}$   $\frac{f}{5.4}$

trmb.  $\frac{mf}{f}$   $\frac{f}{13.12}$

th.  $\frac{mp}{f}$   $\frac{f}{7.8}$

[109]  $\frac{3}{4}$        $\frac{3}{4}$        $\frac{4}{4}$

vln.I  $\frac{f}{mf}$   $\frac{mp}{f}$   $\frac{f}{5.4}$   $\frac{f}{f}$

vln.II  $\frac{f}{mp}$   $\frac{f}{mf}$   $\frac{f}{f}$

vla.  $\frac{mp}{f}$   $\frac{f}{mf}$   $\frac{f}{f}$

vc.  $\frac{mf}{f}$   $\frac{mp}{f}$   $\frac{f}{11.12}$

cb.  $\frac{mp}{f}$   $\frac{f}{9.8}$   $\frac{f}{f}$

[112]

[113]

114  $\frac{5}{4}$  fl.  $\frac{3}{4} \textcircled{N}$  bssn.  $\frac{3}{4}$

cl.  $\frac{3}{4}$  bssn.  $\frac{3}{4}$

bssn.  $\frac{3}{4}$  bssn.  $\frac{3}{4}$

114  $\frac{5}{4}$  hr.  $\frac{3}{4} \textcircled{N}$   $\frac{3}{4}$

trp.  $\frac{3}{4}$  trp.  $\frac{3}{4}$

trmb.  $\frac{3}{4}$  trmb.  $\frac{3}{4}$

th.  $\frac{3}{4}$  th.  $\frac{3}{4}$

114  $\frac{5}{4}$  vln.I  $\frac{3}{4} \textcircled{N}$   $\frac{3}{4}$

vln.II  $\frac{3}{4}$  vln.II  $\frac{3}{4}$

vla.  $\frac{3}{4}$  vla.  $\frac{3}{4}$

vc.  $\frac{3}{4}$  vc.  $\frac{3}{4}$

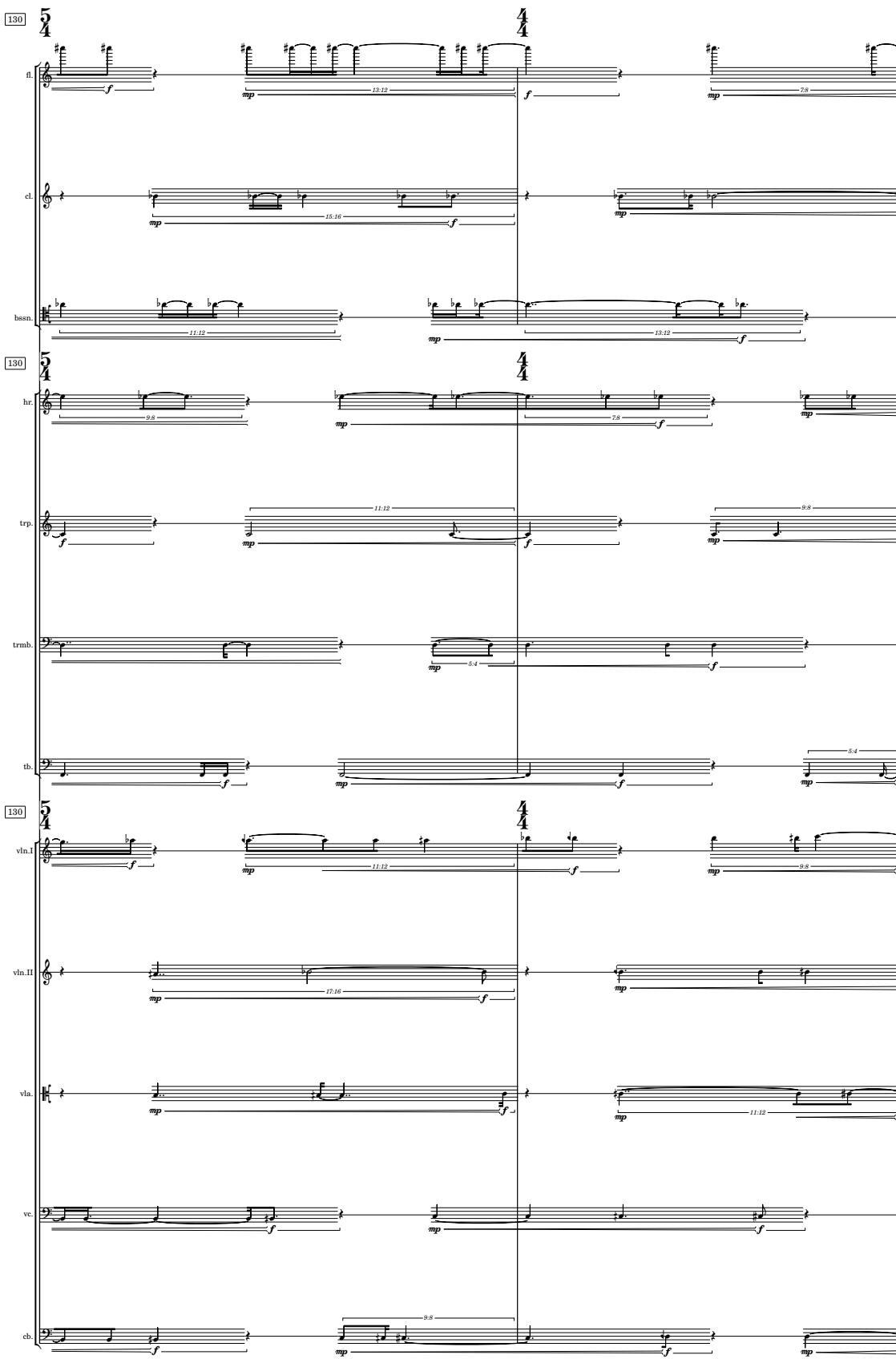
cb.  $\frac{3}{4}$  cb.  $\frac{3}{4}$

|        |             |          |          |          |
|--------|-------------|----------|----------|----------|
|        | <b>2</b>    | <b>3</b> | <b>4</b> | <b>3</b> |
| fl.    | $\text{mp}$ | $mf$     | $f$      | $mp$     |
| cl.    | $f$         | $mp$     | $mf$     | $mp$     |
| bassn. | $mp$        | $f$      | $mp$     | $mf$     |
| hr.    | $mp$        | $mf$     | -        | $f$      |
| trp.   | $mp$        | $mf$     | $mf$     | $mp$     |
| trmb.  | $mp$        | $f$      | $mp$     | $mf$     |
| tb.    | $mp$        | $mf$     | -        | $f$      |
| vln.I  | $mp$        | $mf$     | $mf$     | $mp$     |
| vln.II | $f$         | $mp$     | $mf$     | $mp$     |
| vla.   | $f$         | $mp$     | $mf$     | $mp$     |
| vc.    | $mp$        | $f$      | $mp$     | $mf$     |
| cb.    | $mp$        | $mf$     | -        | $mf$     |

[121]  $\frac{4}{4}$  ○  $\frac{3}{4}$   $\frac{5}{4}$



[127]  $\frac{3}{4}$  fl. f  
cl. 9.8 f  
bsn. mfp  
[127]  $\frac{3}{4}$  (P) trp. f  
trmb. 9.8 f  
th. 9.8  
[127]  $\frac{3}{4}$  vln.I f  
vln.II 7.8 f  
vla. 17.16 f  
vc. 5.4 f  
cb. 7.8 f

[130] 

[132]  $\frac{3}{4}$        $\frac{5}{4}$  (Q)

fl.  $f$

cl.  $f$   
3.4

bassn.  $f$   
mp

[132]  $\frac{3}{4}$        $\frac{5}{4}$  (Q)

hr.  $f$   
3.8

trp.  $f$

trmb.  $mp$   
7.8

th.  $f$

[132]  $\frac{3}{4}$        $\frac{5}{4}$  (Q)

vln.I  $f$

vln.II  $f$   
3.4

vla.  $f$

vc.  $f$   
mp  
9.8

cb.

[134]  $\frac{5}{4}$

f.  
cl.  
bsn.

[134]  $\frac{5}{4}$

hr.  
trp.  
trmb.  
th.

[134]  $\frac{5}{4}$

vln.I  
vln.II  
vla.  
vc.  
cb.

-50-

[136]  $\frac{5}{4}$

f. -

cl.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

bassn.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

[136]  $\frac{5}{4}$   $\frac{4}{4}$

hr.  $\frac{5}{4}$   $\frac{4}{4}$   
mp

trp.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

trmb.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

th.  $\frac{5}{4}$   $\frac{4}{4}$   
mp mf

[136]  $\frac{5}{4}$   $\frac{4}{4}$

vln.I  $\frac{5}{4}$   $\frac{4}{4}$   
mf

vln.II  $\frac{5}{4}$   $\frac{4}{4}$   
mf

vla.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

vc.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

ch.  $\frac{5}{4}$   $\frac{4}{4}$   
mf

[138]  $\frac{4}{4}$  (R)       $\frac{5}{4}$

[138]  $\frac{4}{4}$  (R)       $\frac{5}{4}$

[139]  $\frac{4}{4}$  (R)       $\frac{5}{4}$

[140]  $\frac{4}{4}$  (R)       $\frac{5}{4}$

[140] 

[140] 

[140] 

[140] 

[140] 





[142]

[142]

[142]

[145]  $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

f.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

cl.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

bsn.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

[145]  $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

hr.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

trp.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

trmb.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

th.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

[145]  $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

vln.I       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

vin.II       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

vla.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

vc.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

cb.       $\frac{4}{4}$        $\frac{3}{4}$        $\frac{5}{4}$

[148]  $\frac{5}{4}$   $J = 120$

f.  
cl.  
bassn.

[148]  $\frac{5}{4}$   $\frac{2}{4}$   $\frac{4}{4}$

hr.  
trp.  
trmb.  
th.

[148]  $\frac{5}{4}$   $\frac{2}{4}$   $\frac{4}{4}$

vln.I  
vin.II  
vla.  
vc.  
cb.

151

3 4 4

flute: dynamic ff, measure 9.8, dynamic mf

clarinet: dynamic ff, measure 9.8, dynamic mf

bassoon: dynamic ff, measure 9.8, dynamic mf

horn: dynamic ff, measure 11.12, dynamic mf

trumpet: dynamic ff, measure 15.16, dynamic mf

trombone: dynamic ff, measure 7.8, dynamic mf

tuba: dynamic ff, measure 7.8, dynamic mf

timpani: dynamic p, measure 11.12, dynamic pp

violin I: dynamic ff, measure 7.8, dynamic mf

violin II: dynamic ff, measure 9.8, dynamic mf

viola: dynamic ff, measure 7.8, dynamic mf

cello: dynamic ff, measure 5.4, dynamic pp

bass: dynamic p, measure 3.2, dynamic mf

[154]

[154]

[154]

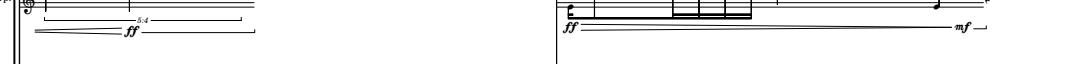
[154]

[156]  

cl. 

bsns. 

[156]  

trp. 

trmb. 

th. 

[156]  

vin.II 

vla. 

vc. 

cb. 

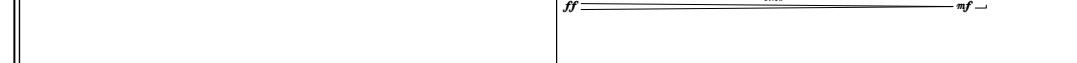
 

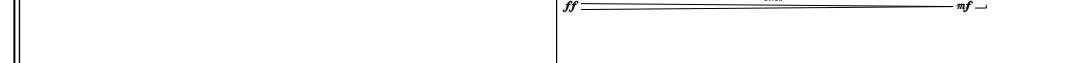
 





163

fl.  $\frac{5}{4}$   $\frac{3}{4}$  (1)  $\frac{3}{4}$

cl.

bassn.  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

163

hr.  $\frac{5}{4}$  (1)  $\frac{3}{4}$

trp.  $\frac{5}{4}$   $\frac{3}{4}$

trmb.  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

tb.  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

163

vln.I  $\frac{5}{4}$  (1)  $\frac{3}{4}$

vln.II  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

vla.  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

vc.  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

cb.  $\frac{5}{4}$   $\frac{3}{4}$   $\frac{3}{4}$

[166]

[167]

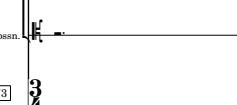
[170]  $\frac{4}{4}$  (1)  $\frac{3}{4}$   $\frac{5}{4}$

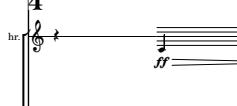
vln.I  
vln.II  
vla.  
vc.  
ch.

173  

fl.  

cl.  

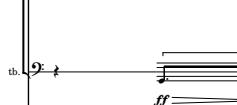
bassn.  

173  

hr.  

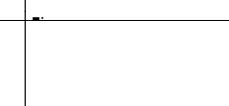
trp.  

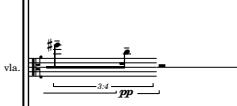
trmb.  

tb.  

173  

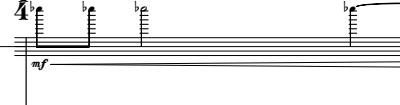
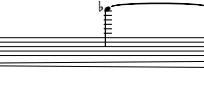
vln.I  

vln.II  

vla.  

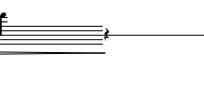
vc.  

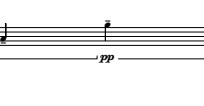
ch.  

173  

fl.  

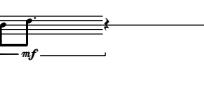
cl.  

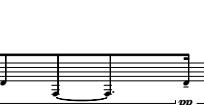
bassn.  

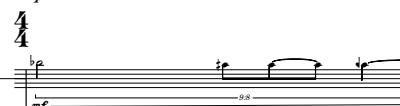
173  

hr.  

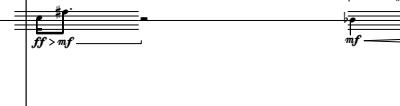
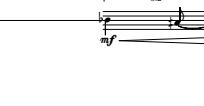
trp.  

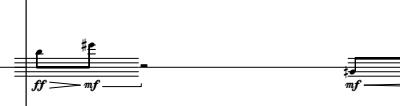
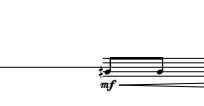
trmb.  

tb.  

173  

vln.I  

vln.II  

vla.  

vc.  

ch.  

[176]  $\frac{3}{4}$  fl. ff — ff = mf —

cl. ff — 5.4 ff —

bassn. —

[176]  $\frac{3}{4}$  (V)  $\frac{4}{4}$  ff — mf — 7.6 —

hr. —

trp. ff — ff = mf — ff > mf —

trmb. — p pp mf ff — mf —

th. — ff = mf ff — mf —

[176]  $\frac{3}{4}$  vln.I ff — ff = mf — ff > mf —

vin.II ff —

vla. ff — 5.4 —

vc. — p pp mf ff — mf —

cb. — ff = mf — mf = ff —

179

Fl. *ff* — *mf* — *ff* — *mf* —

cl. *mf* — *ff* — *mf* — *ff* —

bassoon *f* — *ff* — *mf* — *ff* —

179

hr. *f* — *ff* — *mf* — *ff* — *mf* —

trp. *ff* — *mf* — *ff* — *mf* —

trmb. *f* — *ff* — *mf* — *ff* — *ff* —

thb. *ff* — *mf* — *ff* — *ff* — *mf* —

179

vln. I *ff* — *mf* — *ff* — *mf* —

vln. II *mf* — *ff* — *mf* — *ff* —

vla. *ff* — *mf* — *ff* — *mf* —

vc. *ff* — *mf* — *ff* — *ff* —

ch. *ff* — *mf* — *ff* — *ff* —

181 **3**  
**5 (W)**

fl. *ff* — *mf* —

cl. *ff* — *ff* — *mf* —

bassn. *mf* — *ff* —

181 **3**  
**5 (W)**

hr. *ff* —

trp. *ff* —

trmb. *mf* — *ff* —

tb. *ff* —

181 **3**  
**5 (W)**

vln.I *ff* — *ff* — *mf* —

vln.II *ff* — *ff* — *ff* — *mf* —

vla. *ff* — *ff* — *ff* — *mf* —

vc. *mf* — *ff* —

cb. *ff* —

[183]

[183]

[183]

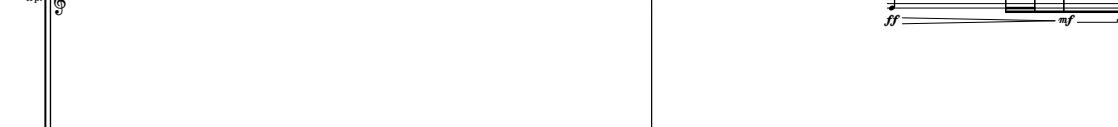
185 

cl. 

bassn. 

185 

trp. 

trmb. 

th. 

185 

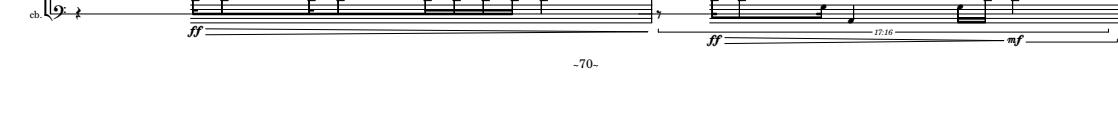
vin.II 

vla. 

vc. 

cb. 



cl. 

bassn. 

[187]  $\frac{4}{4}$  (X)  $\frac{5}{4}$

f.  $p$  — 11.12 —  $pp$  —

cl.

bassn.  $p$  —  $pp$  —

[187]  $\frac{4}{4}$  (X)  $\frac{5}{4}$

hr.  $p$  — 8.4 —  $pp$  —

trp.  $p$  — 15.16 —  $pp$  —

trmb.  $p$  — 15.16 —  $pp$  —

th.  $p$  — 13.12 —  $pp$  —

[187]  $\frac{4}{4}$  (X)  $\frac{5}{4}$

vln.I  $p$  —  $pp$  —

vln.II  $p$  —

vla.  $p$  — 5.4 —  $x$

vc.  $p$  — 13.12 —  $pp$  —

cb.  $p$  — 9.8 —  $pp$  —

189 5

f. -

fl. -

p - 5.8 - pp

cl. -

p - 7.8 - pp

bassn. -

p - pp

5.4 - pp

189 5

hr. -

p - 11.12 -

4

p - pp

trp. -

p - 8.4 - pp

trmb. -

p - 17.16 - pp

th. -

p - pp - 7.8 -

5

vln.I. -

p - pp

4

pp

vln.II. -

p - 8.4 - pp

7.8 - pp

vla. -

p - 7.8 - pp

p - pp

vc. -

p - 11.12 - pp

ch. -

p - 7.8 - pp

p - pp

191

191

fl. *p*

cl. *pp*

bsn. *p* *pp*

hr. *p* *pp*

trp. *p* *pp*

trmb. *p* *pp*

th. *p* *pp*

vln.I *p* *pp*

vln.II *p* *pp*

vla. *p* *pp*

vc. *p* *pp*

cb. *p* *pp*

7.8

7.8

7.8

7.8

7.8

15.16

11.12

9.8

8.8

3.4

5.4

17.16

11.12

9.8

7.8

194 4 3 98

f p 5.8 pp 13.12 p 5.4 pp 15.16

cl. p 5.4 pp

bassn. p 13.12 pp 15.16

194 4 3 98

hr. pp 5.4 pp 15.16

trp. p 9.8 pp 15.16 pp

trmb. p 13.12 pp 15.16 pp

tb. pp 7.8 pp

194 4 3 98

vln.I Sforzando p 15.16 pp

vln.II p 7.8 pp

vla. pp 15.16 pp

vc. p 13.12 pp 15.16 pp

cb. p 17.16 pp

Other scores from Gregory Rowland Evans include:

UNACCOMPANIED  
**Five Excuses** (for cello alone)  
**Five Excuses** (for piano alone)  
**Epihora** (for solo cello)  
**Five Excuses** (for xiao alone)

CHAMBER  
**String Trio no.1**  
**Violin Concerto**  
**Five Excuses** (for string trio)  
**Adumbration "String Trio 2"**  
**Hamon shū "String Quartet 1"**

ELECTROACOUSTIC  
**Bewegt die Erde:**  
*B.E.vii : Obengaußen (for violin)*  
*B.E.viii : Staub (for laptop ensemble)*  
*B.E.x : NGC 3370 (for percussion trio)*  
*B.E.ii : Carinanebel (for viola)*  
*B.E.iv : Arboralkartographie (for cello)*  
**Sidereus Nuncius** (for oboe)

ORCHESTRAL  
**Arquitectura 11611**  
**Metamorphoses** (after Illouz)  
**GUERRERO** (12 saxophones)

B.1.3 FOUR AGES OF SAND (FOR FLUTE, ALTO SAXOPHONE, AND VIOLONCELLO) SCORE

# Four Ages of Sand

for Flute, Alto Saxophone, and Violoncello

2018

Gregory Rowland Evans

## FOREWORD

"Four Ages of Sand" is the title and theme of a lecture given by Douglas Adams, outlining the history of humanity's awareness of their surroundings, leading to the present day where much of biological functions can be modeled computationally. The silicon chip is the fourth age of sand.

(G.R.E.)

## PERFORMANCE NOTES

Accidentals apply only to the pitch which they immediately precede.

c.2'30"

# Four Ages of Sand

for flute, saxophone, and violoncello

Gregory Rowland Evans

$\text{♩} = 60$

Flute: Measure 9:  $9/8$ . Dynamics:  $mf$ ,  $p$ ,  $ff$ . Measure 10:  $ff$ ,  $mp$ . Measure 11:  $5:4$ ,  $13:12$ .

Saxophone: Measures 9-11:  $ff$ ,  $p$ .

Violoncello: Measures 9-11:  $p$ .

(3)

fl. (Measure 5):  $mf$ ,  $ff$ . (Measure 6):  $ff$ ,  $mp$ .

sx. (Measure 5):  $mf$ ,  $ff$ . (Measure 6):  $ff$ ,  $mp$ .

vc. (Measure 5):  $13:12$ . (Measure 6):  $ff$ ,  $mp$ ,  $p$ .

(5)

fl. (Measure 8):  $mf$ ,  $p$ . (Measure 9):  $mf < ff$ . (Measure 10):  $mf$ ,  $p$ . (Measure 11):  $ff$ .

sx. (Measure 8):  $13:12$ ,  $p$ . (Measure 9):  $9:8$ . (Measure 10):  $mf$ ,  $ff$ . (Measure 11):  $21:20$ .

vc. (Measure 8):  $ff$ . (Measure 9):  $ff$ ,  $mp$ . (Measure 10):  $ff$ . (Measure 11):  $p$ .

[8] **3**  
 fl. *mf*  
 sx. *ff*—*mp* *mf*—*p* *mf*—*ff* *ff*—*mf*  
 vc. *mf*—*ff* *mf*—*p*—*ff* *ff*—*mp*

[11] **11**  
**8**  
 fl. *mf*—*p* *mf*—*ff* *mf*—*ff* *mf*—*ff* *ff*—*mf*  
 sx. *ff*—*mp* *mf*—*p* *mf*—*ff* *mf*—*ff* *ff*—*mf*  
 vc. *mf*—*ff* *mf*—*ff*

[13] **8**  
**5**  
 fl. *mf*—*ff* *f*—*mp* *fff*—*mf* *fff*—*mf*  
 sx. *f*—*mp* *fff*—*mf*  
 vc. *f*—*mp* *fff*—*mf*

[15] **4**  
**6**  
**3**  
 fl. *pp*—*3:4* *f*—*mp* *fff*—*7:8* *mf*—*pp* *mf*—*5:4*  
 sx. *pp*—*11:10* *mf*—*f*—*mp* *fff*—*mf*  
 vc. *mf*—*f*—*mp*

18

**4**

fl.

sx.

vc.

9:10

**3**

20

**5**

**4**

fl.

sx.

vc.

3:2

**4**

**3**

22

**11**

fl.

sx.

vc.

7:6

**8**

23

**9**

fl.

sx.

vc.

7:6

**8**

Four Ages of Sand ~3~ Evans

25

5

fl.

sx.

vc.

$p$

$fff$

$mf$

$9.8$

$mf$

$mf$

$mf$

26

4

6

fl.

sx.

vc.

$mp$

$7.8$

$ff$

$p$

$mp$

$5.4$

$ff$

$7.6$

$p$

$mp$

$7.6$

$p$

$mp$

$9.8$

$mf$

$mf$

$mf$

$mf$

$6.5$

$ff$

$p$

$mp$

$3.4$

$ff$

28

3

4

3

fl.

sx.

vc.

$ff$

$mp$

$ff$

$p$

$mp$

$ff$

$9.8$

$mf$

$ff$

$mp$

$ff$

$p$

$mp$

$ff$

$mp$

$ff$

$ff$

$mf$

$ff$

31

5

8

4

fl.

sx.

vc.

$p$

$p$

$9.10$

$mp$

$ff$

$ff$

$9.10$

$mf$

$ff$

$mp$

$ff$

$p$

$fff$

$mp$

$ff$

$7.8$

$ff$

33

11

fl. *mp*

sx. *p* *mp*

vc. *p* *mp*

*fff* *mf*

*fff* *mf*

34

98

68

fl. *p* *mp*

*fff* *mf*

fl. *mp* *ff* *p*

sx. *p* *13:12* *mp*

vc. *p*

36

5

fl. *fff* *mf*

sx. *fff* *mf*

vc. *p* *mp* *fff* *mf*

*13:12*

*9:8*

Four Ages of Sand ~5~ Evans

37

**4**

fl.

sx.

vc.

**6**

39

**3**

fl.

sx.

vc.

**4**

**3**

42

**5**

fl.

sx.

vc.

**4**

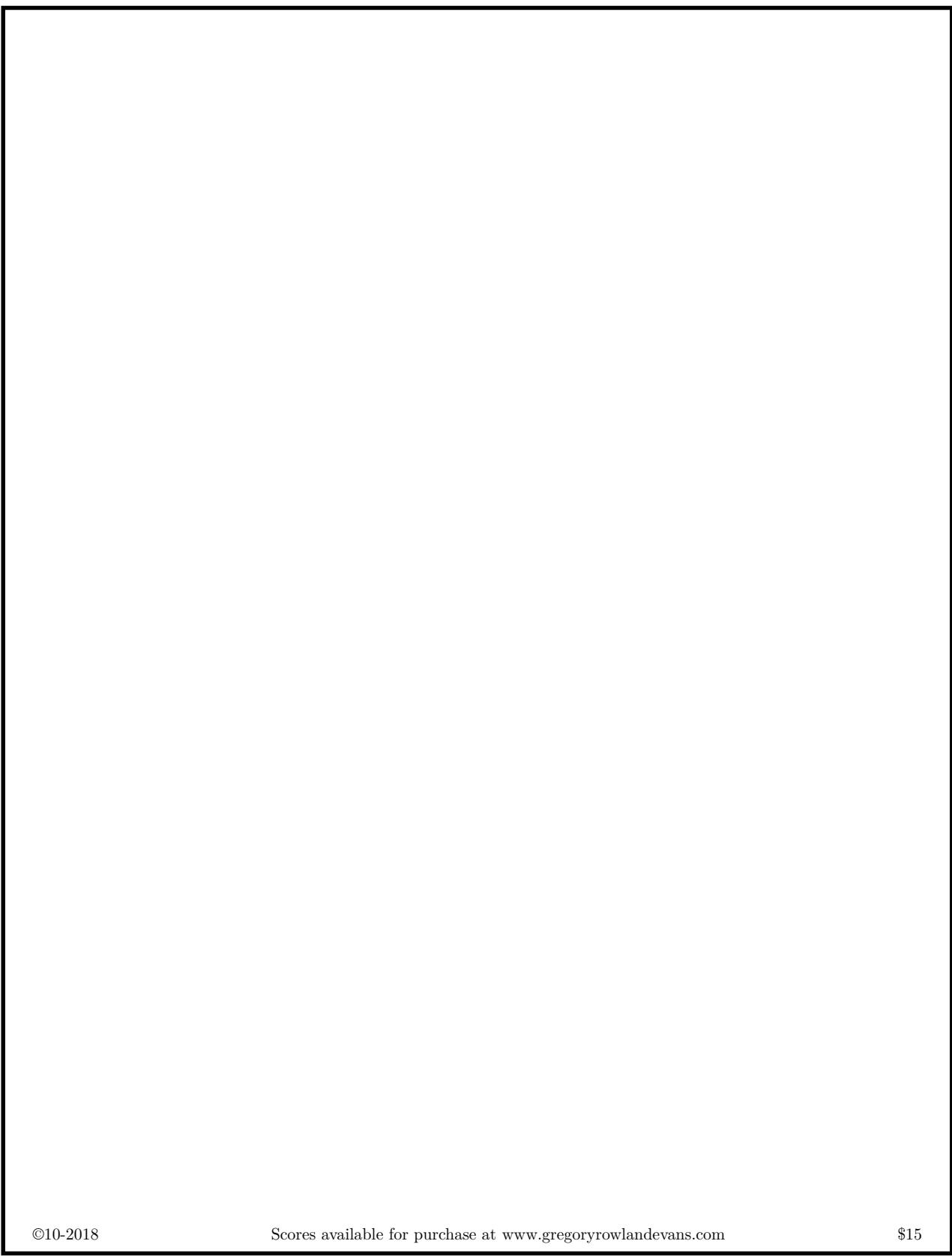
44

**1**

fl.

sx.

vc.



## References

José L. Besada. *Metamodels in Compositional Practices: The Case of Alberto Posadas's Liturgia Fractal*. Editions DELATOURE FRANCE/Ircam-Centre Pompidou, Paris, France, 2017.

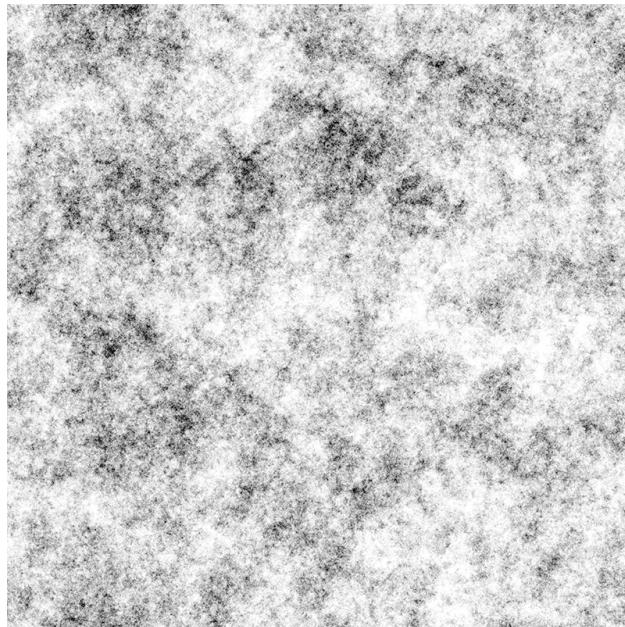
Brian Ferneyhough. *The Collected Writings of Brian Ferneyhough*. Routledge, New York, NY, 1998.

Luigi Nono. *Nostalgia for the Future: Luigi Nono's Selected Writings*. University of California Press, Oakland, CA, 2018.

Josiah W. Oberholtzer. *A Computational Model of Music Composition*. PhD thesis, Harvard University, 2015.

Iannis Xenakis. *Arts/Sciences: Alloys*. Pendragon Press, New York, NY, 1985.

# Colophon



**Figure B.1.1:** A 2-dimensional random walk of 1000 iterations by Paul Bourke.

**T**HIS THESIS WAS TYPESET using L<sup>A</sup>T<sub>E</sub>X, originally developed by Leslie Lamport and based on Donald Knuth's T<sub>E</sub>X. The body text is set in 12 point Arno Pro, designed by Robert Slimbach in the style of book types from the Aldine Press in Venice, and issued by Adobe in 2007. A template, which can be used to format a PhD thesis with this look and feel, has been released under the permissive MIT (x11) license, and can be found online at [github.com/suchow/](https://github.com/suchow/) or from the author at [suchow@post.harvard.edu](mailto:suchow@post.harvard.edu).