

UNIVERSITY OF MIAMI

COMPUTATIONAL NOTATIONS

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, Ph.D.
Professor of Music Composition

Juraj Kojš, Ph.D.
Professor of Music Composition

William Pirkle, M.S.

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

EVANS, GREGORY ROWLAND
Computational Notations;

(M.A., Digital Arts and Sound Design)
(May 2019)

Computational Notations

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. The first chapter of this paper outlines various fundamentals in Lilypond and Python. The second chapter discusses the underlying methodology behind the use of Abjad as a tool for both music composition and music engraving. The third and final chapter consists of an appendix of 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.

**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.

Contents

COVER IMAGE	xii
BLANK PAGE	xiii
1 SOME PREREQUISITE KNOWLEDGE OF LILYPOND AND PYTHON	1
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
2 MY COMPOSITIONAL PRACTICE WITH ABJAD, LILYPOND, AND PYTHON	12
2.1 Methodology	12
2.1.1 The Usefulness of Abjad for me as a Composer	15
2.1.2 Automating Potentially Tedious Tasks	16
2.1.2.1 Creating Notes	16
2.1.2.2 Dynamics, Articulations, and Hairpins	21
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	44

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	52
2.1.3.3	rmakers	53
2.1.4	Other Packages	53
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
A	APPENDIX OF SOURCE CODE	82
A.1	AttachmentHandlers	82
A.1.1	ArticulationHandler	82
A.1.2	ClefHandler	83
A.1.3	DynamicHandler	85
A.1.4	GlissandoHandler	86
A.1.5	NoteheadHandler	87
A.1.6	PitchHandler	88
A.1.7	SlurHandler	89
A.1.8	TrillHandler	90
A.2	Cchar (for two cellos) Source Code	91
A.2.1	Segment	91
A.2.1.1	Segment_I	91
A.2.2	Stylesheet	107
A.3	Tianshu (for 12 players) Source Code	114
A.3.1	Segments	114
A.3.1.1	Segment_I	114
A.3.1.2	Segment_II	150
A.3.1.3	Segment_III	185
A.3.1.4	Segment_IV	221

A.3.2	Stylesheet	257
A.4	Four Ages of Sand (for Flute, Alto Saxophone, and Violoncello) Source Code . . .	262
A.4.1	Segments	262
A.4.1.1	Segment_I	262
A.4.1.2	Segment_II	274
A.4.1.3	Segment_III	285
A.4.1.4	Segment_IV	297
A.4.2	Stylesheet	309
B	APPENDIX OF SCORES	315
B.1	Scores	315
B.1.1	Cthar (for two cellos) Score	315
B.1.2	Tianshu (for 12 players) Score	331
B.1.3	Four Ages of Sand (for flute, alto saxophone, and violoncello) Score . . .	409
REFERENCES		419

List of figures

0.0.1 A 2-dimensional random walk of 100 iterations by Paul Bourke.	xii
2.1.1 A default note.	17
2.1.2 A note with the user-input duration value of (1, 8) and pitch value of 11.	18
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.	22
2.1.6 More notes with dynamics.	23
2.1.7 Notes with algorithmic dynamics.	25
2.1.8 Bow tablature.	27
2.1.9 Extended bow tablature.	30
2.1.10 Very long bow tablature.	34
2.1.11 Extremely long bow tablature.	39
2.1.12 A mapping of a random walk.	47
2.1.13 A mensuration canon.	51
2.1.14 Notation from MusicMaker with two rmakers and PitchHandlers.	61
2.1.15 Trills.	62
2.1.16 Four voice demonstration of AttachmentHandlers.	77
2.2.1 ekmelicStyle evans scale.	80
2.2.2 An eighth tone random walk.	80
B.1.1 A 2-dimensional random walk of 1000 iterations by Paul Bourke.	420

List of Code Examples

1.1	Lilypond syntax	4
1.2	Printing an item of a list through indexing	6
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	8
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	9
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	10
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	11
1.30	Interacting with objects	11
1.31	Interacting with objects: RESULT	11
2.1	Import statement format	16
2.2	Format for object instantiation	16
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	18
2.8	Populating a staff with notes: RESULT	18
2.9	Faster note creation	19
2.10	Tuple zip: RESULT	19
2.11	Duration list comprehension	19
2.12	Duration list comprehension: RESULT	20
2.13	Pitch and duration zip: RESULT	20
2.14	Note object list comprehension: RESULT	20
2.15	Showing the staff	20
2.16	Showing the staff: RESULT	20
2.17	Attaching dynamics	21
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	25
2.24	Bow tablature: RESULT	26
2.25	Extended bow tablature	27
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	Cchar 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	54
2.37	Using MusicMaker with PitchHandler	56
2.38	Using MusicMaker with PitchHandler: RESULT	59
2.39	Using TrillHandler	61
2.40	Using TrillHandler: RESULT	61
2.41	Demonstration of AttachmentHandlers	63
2.42	Demonstration of AttachmentHandlers: RESULT	70
A.1	ArticulationHandler	82
A.2	ClefHandler	83
A.3	DynamicHandler	85
A.4	GlissandoHandler	86
A.5	NoteheadHandler	87
A.6	PitchHandler	88
A.7	SlurHandler	89
A.8	TrillHandler	90
A.9	Cchar Segment_I	91
A.10	Cchar Stylesheet	107
A.11	Tianshu Segment_I	114
A.12	Tianshu Segment_II	150
A.13	Tianshu Segment_III	185
A.14	Tianshu Segment_IV	221
A.15	Tianshu Stylesheet	257
A.16	Four Ages of Sand Segment_I	262
A.17	Four Ages of Sand Segment_II	274
A.18	Four Ages of Sand Segment_III	285
A.19	Four Ages of Sand Segment_IV	297
A.20	Four Ages of Sand Stylesheet	309

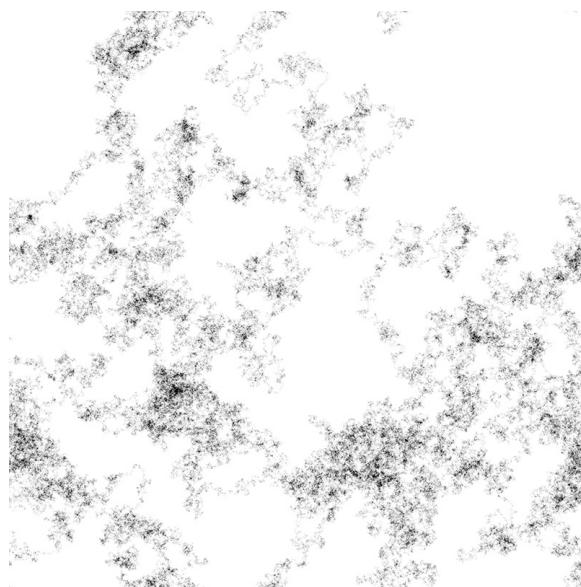


Figure 0.0.1: A 2-dimensional random walk of 100 iterations by Paul Bourke.

This page is intentionally left blank.

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 Interface¹ 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.² 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,³ 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.

¹<http://abjad.mbrsi.org/>

²<http://lilypond.org/>

³<https://www.python.org/>

1.1 LILY POND

1.1.1 COMPARISON 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 Finale⁴ and Sibelius,⁵ with a few new robust programs like Dorico⁶ 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 shortcuts⁷ and opened a clearer accessibility to the engraving of insected 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

⁴<https://www.finalemusic.com/>

⁵<https://www.avid.com/sibelius>

⁶<https://www.dorico.com/>

⁷Most notation software also allows the user to define their own keyboard shortcuts.

engraving programs, like NoteAbility Pro,⁸ which can handle a number of contemporary techniques with ease, other 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,⁹ but these are limited. Programs like Patch Work (and its kin Patch Work Graphic Language¹⁰) and OpenMusic¹¹ 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,¹² 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.¹³ 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.¹⁴

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

⁸[⁹I have mostly seen plugins related to layout spacing and harmonic analysis, but it is my understanding there are more capabilities available.](http://debussy.music.ubc.ca>NoteAbility/</p>
</div>
<div data-bbox=)

¹⁰<http://www2.siba.fi/PWGL/>

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

¹²<https://www.musicxml.com/>

¹³Incidentally, 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.

¹⁴This 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.

like OpenMusic, a set of triplet eighth notes would be written as durations of $\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$ ¹⁵ 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 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.¹⁶ 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^AT_EX¹⁷ syntax, both of which share conceptual similarities with HTML¹⁸ code. The following is a simple example of Lilypond syntax containing an unusual tuplet:¹⁹

```

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

```

¹⁵Often, 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))

¹⁶i.e. instrumentation as well as other invisible contexts

¹⁷<https://www.latex-project.org/>

¹⁸<https://www.w3.org/html/>

¹⁹Note 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.

```

17     a '4  }
18 }
```

Code Example 1.1: Lilypond syntax

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.²⁰ 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 $[0, 1, 2, 3, 4, 5]$,²¹ 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:²²

²⁰In 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.

²¹It 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.

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

```

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:

```

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 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'}
2 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     }
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 Purrr...
6 Purrr...
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.

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,²³ 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,²⁴ 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

²³Graphic User Interface

²⁴<https://www.scheme.com/>

to seem continuous.²⁵ This duplication of processes that were carried out in other voices clutters up the workspace with redundant information. In Abjad, the two concepts of copying and continuing are very distinct,²⁶ allowing the composer to specifically use either technique as needed. Since Abjad is an API,²⁷ 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.²⁸ This is, in part, because these programs have different foci and goals. OM is typically used like a calculator

²⁵This 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.

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

²⁷Application Programming Interface

²⁸Although 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.

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,²⁹ with which music is rendered. Both OM and PWGL are based on CLOS,³⁰ but I believe that the legibility of Python scripts as well as the large number of Python programmers makes it a much better candidate 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

²⁹Expressive Notation Package

³⁰Common LISP Object System

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.³¹

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

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

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:³²

³²http://abjad.mbrsi.org/appendices/pitch_conventions.html

```

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

Code Example 2.11: Duration list comprehension

resulting in:

```

1 [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:

```

1 [(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 loops³³ 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 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])

```

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

```
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

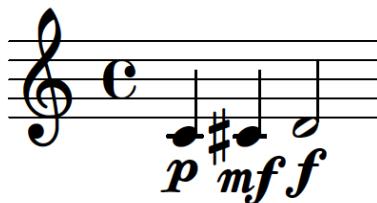


Figure 2.1.5: Notes with dynamics.

Simplifying this further by making use of a loop to attach the dynamics to each leaf³⁴ in the staff, dynamic objects can be created and attached at once:

```
1 >>> import abjad
2 >>> new_staff = abjad.Staff()
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
4   bf'16 b'16")
5 >>> dynamics = ['niente', 'pppp', 'ppp', 'pp', 'p', 'mp', 'mf', 'f', 'ff',
6   'fff', 'ffff', 'sfz', ]
7 >>> leaves = abjad.select(new_staff).leaves()
8 >>> for leaf, dynamic in zip(leaves, dynamics):
...     abjad.attach(abjad.Dynamic(dynamic), leaf)
8 >>> abjad.show(new_staff)
```

Code Example 2.19: Attaching more dynamics

resulting in the Lilypond code and figure: 2.1.6:

³⁴http://abjad.mbrsi.org/core_concepts/lcsi.html

```

1 \score { %! LilyPondFile
2   \new Staff
3   {
4     c'4
5     _ #(make-dynamic-script (markup #:whiteout #:normal-text #:
6       italic "niente"))
7     cs'4
8     \ppp
9     d'2
10    \pp
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  }
30 } %! LilyPondFile

```

Code Example 2.20: Attaching more dynamics: RESULT

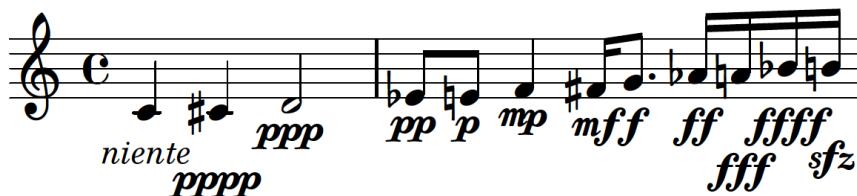


Figure 2.1.6: More notes with dynamics.

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.
4   a'1")
5 >>> for run in abjad.select(music_staff).runs():
6 ...     if len(run) > 3:
7 ...         leaves = abjad.select(run).leaves()
8 ...         abjad.attach(abjad.Dynamic('mf'), run[0])
9 ...         for leaf in leaves:
10 ...             abjad.attach(abjad.Articulation('tenuto'), leaf)
11 ...     elif len(run) == 3:
12 ...         abjad.attach(abjad.Dynamic('f'), run[0])
13 ...         abjad.attach(abjad.StartHairpin('>'), run[0])
14 ...         abjad.attach(abjad.Dynamic('mp'), run[-1])
15 ...     elif len(run) == 1:
16 ...         abjad.attach(abjad.Dynamic('ppp'), run[0])
>>> 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.³⁵

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 represent points along the length of a bow.³⁶ 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])

```

³⁵For 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* too are engraved with the default staff context.

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

```

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      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 }
48 } %! 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
4   '16 c'16")
5 >>> indicator_1 = abjad.BowContactPoint((3, 3))
6 >>> indicator_2 = abjad.BowContactPoint((2, 3))
7 >>> indicator_3 = abjad.BowContactPoint((1, 3))
8 >>> indicator_4 = abjad.BowContactPoint((0, 3))
9 >>> indicator_5 = abjad.BowContactPoint((2, 3))
10 >>> indicator_6 = abjad.BowContactPoint((1, 3))
11 >>> indicator_7 = abjad.BowContactPoint((3, 3))
12 >>> indicator_8 = abjad.BowContactPoint((0, 3))
13 >>> indicator_9 = abjad.BowContactPoint((1, 3))
14 >>> indicator_10 = abjad.BowContactPoint((2, 3))
15 >>> indicator_11 = abjad.BowContactPoint((3, 3))
16 >>> indicator_12 = abjad.BowContactPoint((0, 3))
17 >>> indicators = [indicator_1, indicator_2, indicator_3, indicator_4,
18   indicator_5,
19   ...      indicator_6, indicator_7, indicator_8, indicator_9,
20   indicator_10,
21   ...      indicator_11, indicator_12, ]
22 >>> leaves = abjad.select(new_bow_staff).leaves()
23 >>> for leaf, indicator in zip(leaves, indicators):

```

```

21     ...
22     abjad.attach(indicator, leaf)
>>> 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:

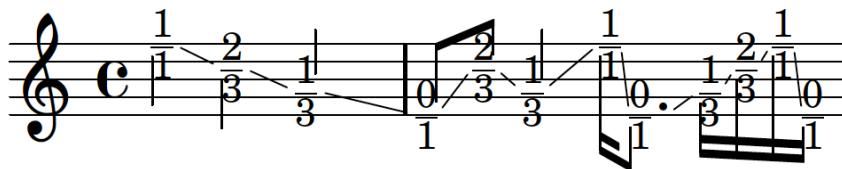


Figure 2.1.9: Extended bow tablature.

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 c'4 c'4 c'4 c'4 c'4 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
6     in numerators]
7 >>> leaves = abjad.select(new_bow_staff).leaves()
8 ...     abjad.attach(indicator, leaf)
9 >>> abjad.bow_contact_spinner(new_bow_staff, omit_bow_changes=True)
10 >>> 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

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:



Figure 2.1.10: Very long bow tablature.

```

1 >>> import abjad
2 >>> newer_bow_staff = abjad.Staff()
3 >>> newer_bow_staff.extend(r"c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
4           c'8 c'8 c'8 ...           c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8")
5 ...
6 >>> leaves = abjad.select(newer_bow_staff).leaves()
7 >>> indicator_numerators = [3, 2, 1, 2, 1, 0, 3, 2, 0, 3, 1, 0, 0, 3, 2,
8   3, 2, 1, 1, 0, 3, ...           2, 1, 0,
9 ...
10 >>> for leaf, numerator in zip(leaves, indicator_numerators):
11   ...     abjad.attach(abjad.BowContactPoint((numerator, 3)), leaf)
12 >>> abjad.bow_contact_spanner(newer_bow_staff, omit_bow_changes=True)
13 >>> 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 #-.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

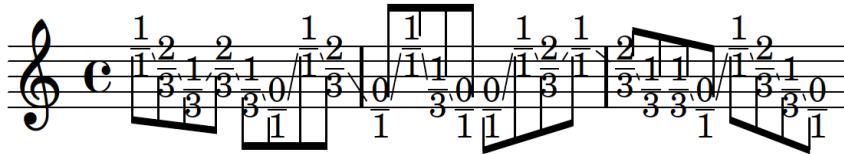


Figure 2.1.11: Extremely long bow tablature.

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*³⁷ by Josiah Wolf Oberholtzer and *Cthar* by Gregory Rowland Evans.

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

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.ily"
8 \ekmelicStyle evans
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
78       :x-aligned-on-self
79         \override TimeSignature.Y-extent = #'(0 . 0)
80     \override TimeSignature.Y-offset = 3
81       \override TimeSignature.break-align-symbol = ##f
82       \override TimeSignature.break-visibility = #end-of-line-
83       invisible
84         \override TimeSignature.font-size = #7
85         \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 = #tuplet-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
155 \context {
156     \Staff
157     \name BeamStaff
158     \type Engraver_group
159     \alias Staff
160     \override Beam.direction = #down
161     \override Beam.positions = #'(5 . 5)
162     \override Clef.stencil = ##f
163     \override Dots.staff-position = #-2
164     \override Flag.Y-offset = #2.93
165     \override NoteHead.no-ledgers = ##t
166     \override NoteHead.stencil = ##f
167     \override Rest.transparent = ##t
168         \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.³⁸ 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
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

```

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

```

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

It is also possible to model more traditional compositional algorithms.³⁹ This code is more

³⁹Admittedly, mensuration canons do not present an extremely complex algorithm, but it meets my definition of



Figure 2.1.12: A mapping of a random walk.

complex than what we have seen before.⁴⁰ 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 = [
8 ...     chain.written_duration*scale_factor for chain in abjad.iterate(staff)
9 ...     .logical_ties()]
10 ...     scaled_staff = abjad.Staff()
11 ...     maker = abjad.NoteMaker()
12 ...     selections = maker(staff_pitches, staff_durations)
13 ...     scaled_staff.extend(selections)
14 ...     return scaled_staff
15
16 >>> def partition_value(value):
17 ...     if x >= 16:
18 ...         divisions, remainder = divmod(value, 8)
19 ...         parts = [8] * divisions
20 ...         if remainder:
21 ...             parts.append(remainder)
22 ...     return parts
23
24 >>> def process_maxima(durations):

```

algorithm.

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

```

25     ...
26     ...
27     ...
28     ...
29     ...
30     ...
31
32 >>> def scale_and_chop_staff(voice_number, staff, time_signature):
33     ...
34     ...
35     ...
36     ...
37     ...
38     ...
39     ...
40     ...
41     ...
42     ...
43     ...
44
45 >>> def make_scaled_staves(melody_staff, time_signature):
46     ...
47     ...
48     ...
49     ...
50     ...
51
52 >>> def duplicate_score(scaled_staves):
53     ...
54     ...
55     ...
56     ...
57     ...
58     ...
59
60 >>> def format_score(score, key_signature, time_signature):
61     ...
62     ...
63     ...
64     ...
65     ...
66     ...
67     ...
68
69 >>> def make_canon(melody_staff, key_signature, time_signature):
70     ...
71     ...
72     ...
73     ...
74

```

This code block contains 74 numbered lines of Python code. Lines 1 through 31 define the `scale_and_chop_staff` function, which takes a voice number, a staff object, and a time signature. It iterates over the durations in the staff, partitioning them if they are 16 or greater. Lines 32 through 37 define the `duplicate_music` function, which creates multiple copies of a staff object. Lines 38 through 50 define the `make_scaled_staves` function, which generates three scaled staves from a melody staff and time signature. Lines 51 through 68 define the `duplicate_score` and `format_score` functions, which handle the creation and formatting of a musical score. Lines 69 through 73 define the `make_canon` function, which combines the previous steps to create a canon piece.

```

75 >>> def rewrite_meter(score):
76 ...     meter = abjad.Meter()
77 ...     for staff in score:
78 ...         for shard in abjad.mutate(staff[:]).split([abjad.Duration(4,
79 ...             4)], cyclic=True):
80 ...             abjad.mutate(shard).rewrite_meter(meter)
81
82 >>> melody_staff = abjad.Staff("c'4 cs'8 d' ds' e' f'4 fs' g' gs'8 a' b'
83 ... c''")
84 >>> score = make_canon(
85 ...     melody_staff, abjad.KeySignature('c', 'major'), abjad.TimeSignature(
86 ...         (4, 4))
87 ... )
88 >>> rewrite_meter(score)
89 >>> abjad.show(score)

```

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''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''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''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

Figure 2.1.13: A mensuration canon.

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.⁴¹ Abjad and 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.⁴² 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.

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,⁴³ a tonal analysis extension, abjad-ext-book,⁴⁴ an extension for rendering Abjad code in L^AT_EX, abjad-ext-ipython,⁴⁵ an extension for rendering Abjad code in IPython and Jupyter notebooks, abjad-ext-nauert,⁴⁶ an extension of quantization tools based on Paul Nauert's Q-Grids, abjad-ext-cli,⁴⁷ a Command Line Interface extension, and

⁴¹ 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.

⁴² As I have previously written, even composing in notation programs such as Finale or Sibelius has restrictions and makes certain procedures difficult or impossible.

⁴³<https://github.com/Abjad/abjad-ext-tonality>

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

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

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

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

abjad-ext-rmakers,⁴⁸ 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.⁴⁹

2.1.3.3 RMAKERS

⁵⁰

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.

2.1.4 OTHER PACKAGES

Along with the official abjad-ext packages are other packages by composers who make use of Abjad, including Consort,⁵¹ a package written by Josiah Wolf Oberholtzer and described in detail in his dissertation, mtools⁵² by Ivan Alexander Moscotta, and calliope⁵³ 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.

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

⁴⁹The 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.

⁵⁰make a subsubsubsection

⁵¹<https://github.com/josiah-wolf-oberholtzer/consort>

⁵²<https://github.com/ivanalexandermoscotta/mtools>

⁵³<https://github.com/mirrorecho/calliope/tree/new-base>

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 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__(self,
14         rmaker,
15         glissando_handler=None,
16         notehead_handler=None,
17         pitch_handler=None,
18         articulation_handler=None,
19         dynamic_handler=None,
20         text_span_handler=None,
21         clef_handler=None,
22         slur_handler=None,
23         trill_handler=None,
24         continuous=False,
25         state=None,
26     ):
27         self.glissando_handler = glissando_handler
28         self.notehead_handler = notehead_handler
29         self.pitch_handler = pitch_handler
30         self.articulation_handler = articulation_handler
31         self.dynamic_handler = dynamic_handler
32         self.text_span_handler = text_span_handler
33         self.clef_handler = clef_handler
34         self.slur_handler = slur_handler
35         self.trill_handler = trill_handler
36         self.continuous = continuous
37         self.rmaker = rmaker
38         self.state = self.rmaker.state
39         self._count = 0
40
41     def __call__(self, durations):
42         return self._make_music(durations)
43
44     def _make_basic_rhythm(self, durations):
45         if self.continuous == True:
46             state = self.state
47             selections = self.rmaker(durations, previous_state=self.
48             rmaker.state)
49             self.state = self.rmaker.state
50         else:
51             selections = self.rmaker(durations, )
52         return selections
53
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-
59                 count = #1 \startStaff',
60                 format_slot='before',
61             )

```

```

61     stop_command = abjad.LilyPondLiteral(
62         r'\stopStaff \startStaff',
63         format_slot='after',
64     )
65     literal = abjad.LilyPondLiteral(r'\once \override Staff.Clef
66 .transparent = ##t', 'before')
67     c_clef = abjad.LilyPondLiteral(r'\clef alto', 'before')
68     abjad.attach(literal, selections[0][0])
69     abjad.attach(c_clef, selections[0][0])
70     abjad.attach(start_command, selections[0][0])
71     abjad.attach(stop_command, selections[0][-1])
72     if self.pitch_handler != None:
73         selections = self.pitch_handler(selections)
74     if self.clef_handler != None:
75         selections = self.clef_handler(selections)
76     if self.glissando_handler != None:
77         selections = self.glissando_handler(selections)
78     if self.notehead_handler != None:
79         selections = self.notehead_handler(selections)
80     if self.articulation_handler != None:
81         selections = self.articulation_handler(selections)
82     if self.dynamic_handler != None:
83         selections = self.dynamic_handler(selections)
84     if self.text_span_handler != None:
85         selections = self.text_span_handler(selections)
86     if self.slur_handler != None:
87         selections = self.slur_handler(selections)
88     if self.trill_handler != None:
89         selections = self.trill_handler(selections)
90     return selections

```

Code Example 2.36: MusicMaker source

MusicMaker is made to be called with timespans⁵⁴ and can be used as follows:

```

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
16 >>> rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
...     talea=abjadext.rmakers.Talea(

```

⁵⁴Timespans are also described in great detail in Josiah Oberholtzer's dissertation.(Oberholtzer, 2015, pp. 78–118)

```

17     ...
18     ...
19     ...
20     ...
21     beamSpecifier=abjadext.rmakers.BeamSpecifier(
22     ...
23         beam_divisions_together=True,
24         beam_rests=False,
25         ),
26     ...
27     extra_counts_per_division=[0, 1,],
28     ...
29     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
30         ...
31             trivialize=True,
32             extract_trivial=True,
33             rewrite_rest_filled=True,
34             rewrite_sustained=True,
35             ),
36     ...
37     )
38
39
40
41
42
43 >>> rmaker_two = abjadext.rmakers.EvenDivisionRhythmMaker(
44     ...
45         denominators=[8, 16,],
46         extra_counts_per_division=[0,],
47         tupletSpecifier=abjadext.rmakers.TupletSpecifier(
48             ...
49                 trivialize=True,
50                 extract_trivial=True,
51                 rewrite_rest_filled=True,
52                 ),
53             ...
54             )
55
56
57
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     ...
69         abjad.AnnotatedTimespan(
70             start_offset=start_offset,
71             stop_offset=stop_offset,
72             annotation=MusicSpecifier(

```

```
71 ...     music_maker=music_maker,
72 ...     voice_name='Voice 1',
73 ... ),
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 >>> score = abjad.Score([
120 ...     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
121 ...     Context'),
122 ...     abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
123 ...     lilypond_type='Staff').,
```

```

122     ... ],
123     ... )
124
125 >>> for time_signature in time_signatures:
126     ...     skip = abjad.Skip(1, multiplier=(time_signature))
127     ...     abjad.attach(time_signature, skip)
128     ...     score['Global Context'].append(skip)
129
130 >>> def make_container(music_maker, durations):
131     ...     selections = music_maker(durations)
132     ...     container = abjad.Container([])
133     ...     container.extend(selections)
134     ...     return container
135
136 >>> def key_function(timespan):
137     ...     return timespan.annotation.music_maker or silence_maker
138
139 >>> for voice_name, timespan_list in all_timespan_lists.items():
140     ...     for music_maker, grouper in itertools.groupby(
141     ...         timespan_list,
142     ...         key=key_function,
143     ...     ):
144     ...         durations = [timespan.duration for timespan in grouper]
145     ...         container = make_container(music_maker, durations)
146     ...         voice = score[voice_name]
147     ...         voice.append(container)
148
149 >>> for voice in abjad.iterate(score['Voice 1']).components(abjad.Voice):
150     ...     for i, shard in enumerate(abjad.mutate(voice[:]).split(
151     ...         time_signatures)):
152     ...         time_signature = time_signatures[i]
153     ...         abjad.mutate(shard).rewrite_meter(time_signature)
154
155 >>> for voice in abjad.select(score).components(abjad.Voice):
156     ...     for run in abjad.select(voice).runs():
157     ...         specifier = abjadext.rmakers.BeamSpecifier(
158     ...             beam_each_division=False,
159     ...             )
160     ...         specifier(run)
161     ...         abjad.beam(voice[:], beam_lone_notes=False, beam_rests=False,)
162
163 >>> 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 {
6 ...     b8 a b }'
7 ... )
8 >>> score = abjad.Score([staff])
9 >>> 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  \layout {}
8
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
28 >>> rmaker = abjadext.rmakers.TaleaRhythmMaker(
29 ...     talea=abjadext.rmakers.Talea(
30 ...         counts=[2, 1, 5, 1, 3, 3, 1, 2, ],
31 ...         denominator=16,
32 ...         ),
33 ...     beamSpecifier=abjadext.rmakers.BeamSpecifier(
34 ...         beamDivisionsTogether=True,
35 ...         beamRests=False,
36 ...         ),
37 ...     extraCountsPerDivision=[0, 0, 1, 0, -1],
38 ...     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
39 ...         leftClasses=[abjad.Note, abjad.Rest],
40 ...         leftCounts=[1, 0, 1],
41 ...         ),
42 ...     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
43 ...         trivialize=True,
44 ...         extractTrivial=True,
45 ...         rewriteRestFilled=True,
46 ...         ),
47 ... )
48
49 >>> articulation_handler = ArticulationHandler(
50 ...     articulationList=['tenuto', 'staccato', 'portato', ],
51 ...     continuous=True,
52 ... )
53 >>> clef_handler = ClefHandler(

```

```
54 ...
55 ...
56 ...
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     ...         annotation=MusicSpecifier(
150     ...             rhythm_maker=rhythm_maker,
151     ...             voice_name='Voice 2',
152     ...             ),
153     ...         )
154     ...         for start_offset, stop_offset, rhythm_maker in [
155     ...             [(0, 8), (4, 8), music_maker],
156     ...             [(5, 8), (7, 8), music_maker],
157     ...             [(7, 8), (8, 8), silence_maker],
158     ...             ]
159     ...         ])
160
161 >>> voice_3_timespan_list = abjad.TimespanList([

```

```
162 ...     abjad.AnnotatedTimespan(
163 ...         start_offset=start_offset,
164 ...         stop_offset=stop_offset,
165 ...         annotation=MusicSpecifier(
166 ...             rhythm_maker=rhythm_maker,
167 ...             voice_name='Voice 3',
168 ...         ),
169 ...     )
170 ...     for start_offset, stop_offset, rhythm_maker in [
171 ...         [(0, 8), (4, 8), music_maker],
172 ...         [(5, 8), (7, 8), music_maker],
173 ...         [(7, 8), (8, 8), silence_maker],
174 ...     ]
175 ... ]
176
177 >>> voice_4_timespan_list = abjad.TimespanList([
178 ...     abjad.AnnotatedTimespan(
179 ...         start_offset=start_offset,
180 ...         stop_offset=stop_offset,
181 ...         annotation=MusicSpecifier(
182 ...             rhythm_maker=rhythm_maker,
183 ...             voice_name='Voice 4',
184 ...         ),
185 ...     )
186 ...     for start_offset, stop_offset, rhythm_maker in [
187 ...         [(0, 8), (4, 8), music_maker],
188 ...         [(5, 8), (7, 8), music_maker],
189 ...         [(7, 8), (8, 8), silence_maker],
190 ...     ]
191 ... ]
192
193 >>> all_timespan_lists = {
194 ...     'Voice 1': voice_1_timespan_list,
195 ...     'Voice 2': voice_2_timespan_list,
196 ...     'Voice 3': voice_3_timespan_list,
197 ...     'Voice 4': voice_4_timespan_list,
198 ... }
199
200 >>> global_timespan = abjad.Timespan(
201 ...     start_offset=0,
202 ...     stop_offset=max(_.stop_offset for _ in all_timespan_lists.values
203 ...     ())
204 ... )
205
206 >>> for voice_name, timespan_list in all_timespan_lists.items():
207 ...     silences = abjad.TimespanList([global_timespan])
208 ...     silences.extend(timespan_list)
209 ...     silences.sort()
210 ...     silences.compute_logical_xor()
211 ...     for silence_timespan in silences:
212 ...         timespan_list.append(
213 ...             abjad.AnnotatedTimespan(
214 ...                 start_offset=silence_timespan.start_offset,
```

```

215     ...
216     ...
217     ...
218     ...
219     ...
220     ...
221     timespan_list.sort()
222
223 >>> for voice_name, timespan_list in all_timespan_lists.items():
224     ...
225     shards = timespan_list.split_at_offsets(bounds)
226     split_timespan_list = abjad.TimespanList()
227     for shard in shards:
228         split_timespan_list.extend(shard)
229     split_timespan_list.sort()
230     all_timespan_lists[voice_name] = timespan_list
231
232 >>> score = abjad.Score([
233     ...
234     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
235     Context'),
236     ...
237     abjad.StaffGroup(
238         [
239             abjad.Staff(
240                 [abjad.Voice(name='Voice 1')], name='Staff 1', lilypond_type='Staff',
241             ),
242             abjad.Staff(
243                 [abjad.Voice(name='Voice 2')], name='Staff 2', lilypond_type='Staff',
244             ),
245             abjad.Staff(
246                 [abjad.Voice(name='Voice 3')], name='Staff 3', lilypond_type='Staff',
247             ),
248             abjad.Staff(
249                 [abjad.Voice(name='Voice 4')], name='Staff 4', lilypond_type='Staff',
250             ),
251             ...
252             ],
253             name='Staff Group',
254             )
255         ...
256     ])
257
258 >>> for time_signature in time_signatures:
259     skip = abjad.Skip(1, multiplier=(time_signature))
260     abjad.attach(time_signature, skip)
261     score['Global Context'].append(skip)
262
263 >>> print('Making containers ...')
264
265 >>> def make_container(rhythm_maker, durations):
266     ...
267     selections = rhythm_maker(durations)
268     container = abjad.Container([])
269     container.extend(selections)
270     return container
271
272 >>> def key_function(timespan):
273     ...
274     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         % [Voice 1 measure 1] %! COMMENT_MEASURE_NUMBERS
43         \set Staff.shortInstrumentName =
44         \markup { "vln. I" }
45         \set Staff.instrumentName =
46         \markup { "Violin I" }
47         \tempo 4=108
48         \once \override Staff.NoteHead.style = #'cross
49         \clef "bass"
50         c'8
51         \f
52         - \tenuto
53       \>

```

```

53           - \tweak style #'dotted-line
54           \glissando
55           (
56           [
57           \once \override Staff.NoteHead.style = #
58           harmonic-mixed
59           d'16
60           - \staccato
61           - \tweak style #'dotted-line
62           \glissando
63           ]
64           \once \override Staff.NoteHead.style = #'diamond
65           cs'4
66           ~
67           - \tweak style #'dotted-line
68           \glissando
69           \once \override Staff.NoteHead.style = #'diamond
70           cs'16
71           \p
72           - \tweak stencil #constante-hairpin
73           \lless
74           )
75       }
76       r8
77       \!
78   }
79   {
80     \once \override Staff.NoteHead.style = #
81     triangle
82     \clef "bass"
83     cs'16
84     \f
85     - \portato
86     \r
87     - \tweak style #'dotted-line
88     \glissando
89     (
90     [
91     \ottava 1
92     \once \override Staff.NoteHead.style = #'slash
93     <ef' bf'>8.
94     \p
95     - \tenuto
96     - \tweak stencil #constante-hairpin
97     \lless
98     )
99     ]
100    \ottava 0
101  }
102  {
103    r8
104    \!
105    \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   {
117     \times 8/9 {
118       % [Voice 2 measure 1] %!
119
120   COMMENT_MEASURE_NUMBERS
121
122   \set Staff.shortInstrumentName =
123   \markup { "vln. II" }
124   \set Staff.instrumentName =
125   \markup { "Violin II" }
126   \ottava 1
127   \once \override Staff.NoteHead.style = #
128
129 default
130
131   \clef "bass"
132   <ef' bf'>8.
133   \f
134   - \staccato
135   \>
136   - \tweak style #'dotted-line
137   \glissando
138   (
139   [
140   \ottava 0
141   \ottava 1
142   \once \override Staff.NoteHead.style = #
143
144 cross
145
146   e'16
147   - \portato
148   - \tweak style #'dotted-line
149   \glissando
150   \ottava 0
151   \ottava 1
152   \once \override Staff.NoteHead.style = #
153
154 harmonic-mixed
155
156   af'8
157   - \tenuto
158   - \tweak style #'dotted-line
159   \glissando
160   \ottava 0
161   \ottava 1
162   \ottava 1
163   \once \override Staff.NoteHead.style = #
164
165 diamond
166
167   <g' a'>8
168   - \staccato

```

```

154           - \tweak style #'dotted-line
155           \glissando
156           \ottava 0
157           \ottava 0
158           \ottava 1
159           \once \override Staff.NoteHead.style = #
160           triangle
161           fs'16
162           \p
163           - \portato
164           - \tweak stencil #constante-hairpin
165           \(
166           )
167           ]
168           \ottava 0
169       }
170   {
171       r8
172       \!
173   {
174       \ottava 1
175       \once \override Staff.NoteHead.style = #'slash
176       \clef "bass"
177       fs'4
178       \f
179       - \tenuto
180       - \tweak stencil #constante-hairpin
181       \(
182       )
183       (
184       \ottava 0
185   }
186   {
187       r8
188       \!
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         {
201             \tweak text #tuple-number::calc-fraction-text
202             \times 8/7 {
203                 % [Voice 3 measure 1] %
204 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 = #' 
211 default
212                               \clef "bass"
213                               fs'16
214                               \f
215                               - \staccato
216                               \>
217                               - \tweak style #'dotted-line
218                               \glissando
219                               (
220                               [
221                               \ottava 0
222                               \once \override Staff.NoteHead.style = #' 
223 cross
224                               c'16
225                               - \portato
226                               - \tweak style #'dotted-line
227                               \glissando
228                               \once \override Staff.NoteHead.style = #' 
229 harmonic-mixed
230                               d'8.
231                               - \tenuto
232                               - \tweak style #'dotted-line
233                               \glissando
234                               \once \override Staff.NoteHead.style = #' 
235 diamond
236                               cs'8
237                               \p
238                               - \staccato
239                               - \tweak stencil #constante-hairpin
240                               \<
241                               )
242                               ]
243                               }
244                               }
245                               {
246                               r8
247                               \!
248                               }
249                               {
250                               \once \override Staff.NoteHead.style = #' 
251 triangle
252                               \clef "bass"
253                               cs'16
254                               \f

```

```

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 = #
307
harmonic-mixed

```

```

308                         af '16
309
310             - \tenuto
311             - \tweak style #'dotted-line
312             \glissando
313             ]
314             \ottava 0
315             \ottava 1
316             \ottava 1
317             \once \override Staff.NoteHead.style = #'diamond
318             <g' a'>4
319             ~
320             - \tweak style #'dotted-line
321             \glissando
322             \once \override Staff.NoteHead.style = #'diamond
323             <g' a'>16
324             \p
325             - \tweak stencil #constante-hairpin
326             \(
327             )
328             \ottava 0
329             \ottava 0
330         }
331         {
332             r8
333             \!
334         {
335             \times 4/5 {
336                 \ottava 1
337                 \ottava 1
338                 \once \override Staff.NoteHead.style = #
triangle
339
340                 \clef "bass"
341                 <g' a'>16
342                 \f
343                 - \staccato
344                 \(
345                 \tweak style #'dotted-line
346                 \glissando
347                 (
348                 [
349                 \ottava 0
350                 \ottava 0
351                 \ottava 1
352                 \once \override Staff.NoteHead.style = #
slash
353
354                 fs'8.
355                 - \portato
356                 - \tweak style #'dotted-line
357                 \glissando
358                 \ottava 0
359                 \once \override Staff.NoteHead.style = #
default
360
361                 c'16

```

```

359          \p
360          - \tenuto
361          - \tweak stencil #constante-hairpin
362          \<
363          )
364          ]
365      }
366  }
367  {
368      r8
369      \!
370  }
371  }
372  >>
373  >>
374 } %! 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,⁵⁵ 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

⁵⁵These are often called grobs in Lilypond-lingo.

3.1, but will hopefully be given new birth in a future release.⁵⁶

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.⁵⁷ 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 tones⁵⁸ and the Ferneyhough notation for all other microtonal alterations.⁵⁹ This became cumbersome and inconsistent and I looked for an alternative. Fortunately, I found the Ekmelily system.⁶⁰ 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:

⁵⁶However, these changes are available in my greg/dev branch of Abjad.

⁵⁷This decision is informed, in part, by my familiarity with OpenMusic's accidentals.

⁵⁸<https://w3c.github.io/smufi/gitbook/tables/extended-stein-zimmermann-accidentals.html>

⁵⁹<https://w3c.github.io/smufi/gitbook/tables/other-accidentals.html>

⁶⁰<http://www.ekmelic-music.org/en/extra/ekmelily.htm>

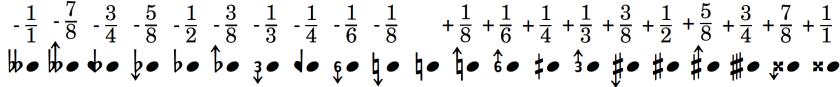


Figure 2.2.1: ekmelicStyle evans scale.

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.

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__(self,
6                  articulation_list=None,
7                  continuous=False,
8                  ):
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)),
31 tie[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
38                 =True):
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
46                     =True):
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
53                     =True):
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
61                     =True):
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
68                     =True):
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
76                     =True):
77                         for pitch in abjad.inspect(tie[0]).pitches():
78                             if pitch > shelf:
79                                 abjad.ottava(tie)
80                         else:
81                             shelf = 10
82                             for tie in abjad.select(selections).logical_ties(pitched
83                     =True):
84                         for pitch in abjad.inspect(tie[0]).pitches():
85                             if pitch > shelf:
86                                 abjad.ottava(tie)
87                         if self.clef == 'tenorvarC':
88                             if self.ottava_shelf != None:
89                                 shelf = self.ottava_shelf
90                                 for tie in abjad.select(selections).logical_ties(pitched

```

```

= True):
    for pitch in abjad.inspect.tie[0].pitches():
        if pitch > shelf:
            abjad.ottava(tie)
else:
    shelf = 10
    for tie in abjad.select(selections).logical_ties(pitched
=True):
        for pitch in abjad.inspect.tie[0].pitches():
            if pitch > shelf:
                abjad.ottava(tie)
if self.clef == 'bass':
    if self.ottava_shelf != None:
        shelf = self.ottava_shelf
        for tie in abjad.select(selections).logical_ties(pitched
=True):
            for pitch in abjad.inspect.tie[0].pitches():
                if pitch > shelf:
                    abjad.ottava(tie)
else:
    shelf = 3
    for tie in abjad.select(selections).logical_ties(pitched
=True):
        for pitch in abjad.inspect.tie[0].pitches():
            if pitch > shelf:
                abjad.ottava(tie)
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),
36                     leaves[0])
37                     if self.hairpin != None:
38                         abjad.attach(abjad.StartHairpin(self.hairpin),
39                         leaves[0])
39                     if self.ending_dynamic != None:
40                         abjad.attach(abjad.Dynamic(self.ending_dynamic),
41                         leaves[-1])
41                         abjad.attach(abjad.StartHairpin('---'), leaves[-1])
42                 else:
43                     leaves = abjad.select(run).leaves()
44                     dynamic = next(self._cyc_dynamics)
45                     if self.starting_dynamic != None:
46                         if self.ending_dynamic != None:
47                             abjad.attach(abjad.Dynamic(dynamic), leaves[0])
48                         else:
49                             abjad.attach(abjad.Dynamic(self.starting_dynamic
50 ), leaves[0])
51                         if self.starting_dynamic == None:
52                             if self.ending_dynamic != None:
53                                 abjad.attach(abjad.Dynamic(self.ending_dynamic),
54                                 leaves[0])
55                                 abjad.attach(abjad.StartHairpin('---'), leaves[0])
56             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.
23                             line_style).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=
28                             True, )
29             elif self.glissando_style == 'hide_middle_stems':
30                 if self.line_style != None:
31                     for run in runs:
32                         if len(run) > 1:
33                             abjad.glissando(run[:], abjad.tweak(self.
34                             line_style).style, hide_middle_note_heads=True, hide_middle_stems=
35                             True, )
36             else:
37                 for run in runs:
38                     if len(run) > 1:
39                         abjad.glissando(run[:], hide_middle_note_heads=
40                             True, hide_middle_stems=True, )
41             else:
42                 if self.line_style != None:
43                     for run in runs:
44                         if len(run) > 1:
45                             abjad.glissando(run[:], abjad.tweak(self.
46                             line_style).style, allow_repeats=True, allow_ties=True, )
47             else:
48                 for run in runs:
49                     if len(run) > 1:
50                         abjad.glissando(run[:], allow_repeats=True,
51                             allow_ties=True, )
52         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                for leaf in abjad.select(tie).leaves(pitched=True):
33                    abjad.attach(style, leaf)
34        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__(self, slurs=None):
6         self.slurs = slurs
7
8     def __call__(self, selections):
9         return self.add_slurs(selections)
10
11    def add_slurs(self, selections):
12        if self.slurs == 'selections':
13            abjad.slur(selections[:])
14        elif self.slurs == 'runs':
15            for run in abjad.select(selections).runs():
16                abjad.slur(run[:])
17        else:
18            pass
19        return selections
20
21
22

```

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.
14                   iterate(tie).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.
21                    written_duration)
22
23                trill_start = abjad.LilyPondLiteral(r'\pitchedTrill',
24                    format_slot='before')
25                trill_literal = abjad.LilyPondLiteral(f'\startTrillSpan
26 {trill_pitch}\', format_slot='after')
27                trill_stop = abjad.LilyPondLiteral(r'\stopTrillSpan',
28                    format_slot='after')
29                abjad.attach(trill_start, new_leaf)
30                abjad.attach(trill_literal, new_leaf)
31
32
33

```

```

25         last_leaf = tie[-1]
26         next_leaf = abjad.inspect(last_leaf).leaf(1)
27         if next_leaf != None:
28             abjad.attach(trill_stop, next_leaf)
29
30         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.
40             written_duration)
41                 parent = abjad.inspect(leaf).parentage().parent
42                 parent[parent.index(leaf)] = new_tail
43                 indicators = abjad.inspect(leaf).indicators()
44                 for indicator in indicators:
45                     abjad.attach(indicator, new_tail)
46
47     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 ]
26
27 bounds = abjad.mathtools.cumulative_sums([_.duration for _ in
28 time_signatures])
29
30 def reduceMod7(rw):
31     return [(x % 8) for x in rw]
32
33 def reduceMod9(rw):
34     return [(x % 10) for x in rw]
35
36 def reduceMod17(rw):
37     return [(x % 18) for x in rw]
38
39 def reduceMod21(rw):
40     return [(x % 22) for x in rw]
41
42 def reduceMod47(rw):
43     return [(x % 48) for x in rw]
44
45 def cyc(lst):
46     count = 0
47     while True:
48         yield lst[count%len(lst)]
49         count += 1
50
51 def grouper(lst1, lst2):
52     def cyc(lst):
53         c = 0
54         while True:
55             yield lst[c%len(lst)]
56             c += 1
57     lst1 = cyc(lst1)
58     return [next(lst1) if i == 1 else [next(lst1) for _ in range(i)] for
59     i in lst2]
60
61 seed(1)
62 cello_random_walk_one = []
63 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
64 for i in range(1, 1000):
65     movement = -1 if random() < 0.5 else 1
66     value = cello_random_walk_one[i-1] + movement
67     cello_random_walk_one.append(value)
68 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
69 cello_chord_one = [-12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8,
70     -7.5, -7, -6.5, -6, -5.5, -5, -4.5, -4, -3.5, -3, -2.5, -2, -1.5,
71     -1, -0.5, 0, -0.5, -1, -1.5, -2, -2.5, -3, -3.5, -4, -4.5, -5, -5.5,
72     -6, -6.5, -7, -7.5, -8, -8.5, -9, -9.5, -10, -10.5, -11, -11.5, ]

```

```

68 cello_notes_one = [cello_chord_one[x] for x in reduceMod47(
    cello_random_walk_one)]
69
70 seed(2)
71 cello_random_walk_two = []
72 cello_random_walk_two.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_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, -6, -24, -8, 0, ]
79 cello_notes_two_walk = [cello_chord_two[x] for x in reduceMod17(
    cello_random_walk_two)]
80 map_1 = [1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, ]
81 cello_notes_two = grouper(cello_notes_two_walk, map_1)
82
83 seed(3)
84 cello_random_walk_three = []
85 cello_random_walk_three.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 = cello_random_walk_three[i-1] + movement
89     cello_random_walk_three.append(value)
90 cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
91 cello_chord_three = [-24, -20, -15, -14, -4, 5, 11, 19, 26, 37, 39, 42,
    39, 37, 26, 19, 11, 5, -4, -14, -15, -20, ]
92 cello_notes_three = [cello_chord_three[x] for x in reduceMod21(
    cello_random_walk_three)]
93
94 seed(4)
95 cello_random_walk_four = []
96 cello_random_walk_four.append(-1 if random() < 0.5 else 1)
97 for i in range(1, 2000):
98     movement = -1 if random() < 0.5 else 1
99     value = cello_random_walk_four[i-1] + movement
100    cello_random_walk_four.append(value)
101 cello_random_walk_four = [abs(x) for x in cello_random_walk_four]
102 cello_chord_four = [-17, -8, -13, -5, 5, -5, -13, -8, ]
103 map_2 = [2, 1, 2, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 2, 1, ]
104 cello_notes_four_walk = [cello_chord_four[x] for x in reduceMod7(
    cello_random_walk_four)]
105 cello_notes_four = grouper(cello_notes_four_walk, map_2)
106
107 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
108     talea=abjadext.rmakers.Talea(
109         counts=[7, 4, 6, 3, 5, 3, 5, 3, 6, 4],
110         denominator=32,
111         ),
112     beamSpecifier=abjadext.rmakers.BeamSpecifier(
113         beamDivisionsTogether=True,
114         beamRests=False,
115         ),

```

```

116     extra_counts_per_division=[0, 1, 0, -1],
117     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
118         trivialize=True,
119         extract_trivial=True,
120         rewrite_rest_filled=True,
121         ),
122     )
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         beam_divisions_together=True,
131         beam_rests=False,
132         ),
133     extra_counts_per_division=[1, 0, -1, 0, 1],
134     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
135         trivialize=True,
136         extract_trivial=True,
137         rewrite_rest_filled=True,
138         ),
139     )
140
141 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
142     denominators=[8, 8, 16, 8, 8, 16],
143     extra_counts_per_division=[0, 1, 0, 0, -1, 0, 1, -1],
144     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
145         trivialize=True,
146         extract_trivial=True,
147         rewrite_rest_filled=True,
148         ),
149     )
150
151 attachment_handler_one = AttachmentHandler(
152     starting_dynamic='p',
153     ending_dynamic='mp',
154     hairpin_indicator='--',
155     articulation='accent',
156     )
157
158 attachment_handler_two = AttachmentHandler(
159     starting_dynamic='fff',
160     ending_dynamic='mf',
161     hairpin_indicator='>',
162     articulation='tenuto',
163     )
164
165 attachment_handler_three = AttachmentHandler(
166     starting_dynamic='mp',
167     ending_dynamic='ff',
168     hairpin_indicator='<|',
169     articulation='|',

```



```

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',
508                 Context'),
509     abjad.StaffGroup(
510         [
511             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
512                         lilypond_type='BowStaff',),
513             abjad.Staff([abjad.Voice(name='Voice 5')], name='Staff 5',
514                         lilypond_type='BeamStaff',),
515             abjad.Staff([abjad.Voice(name='Voice 2')], name='Staff 2',
516                         lilypond_type='Staff',),
517             ],
518             name='Staff Group 1',
519         ),
520     abjad.StaffGroup(
521         [
522             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
523                         lilypond_type='BowStaff',),
524             abjad.Staff([abjad.Voice(name='Voice 6')], name='Staff 6',
525                         lilypond_type='BeamStaff',),
526             abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
527                         lilypond_type='Staff',),
528             ],
529             name='Staff Group 2',
530         )
531     ],
532 )
533
534 for time_signature in time_signatures:
535     skip = abjad.Skip(1, multiplier=(time_signature))
536     abjad.attach(time_signature, skip)
537     score['Global Context'].append(skip)
538
539 print('Making containers ...')
540
541 def make_container(music_maker, durations):
542     selections = music_maker(durations)
543     container = abjad.Container([])
544     container.extend(selections)
545     return container
546
547 def key_function(timespan):
```

```

541     return timespan.annotation.music_maker or silence_maker
542
543 for voice_name, timespan_list in all_timespan_lists.items():
544     for music_maker, grouper in itertools.groupby(
545         timespan_list,
546         key=key_function,
547     ):
548         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(
564         time_signatures)):
565         time_signature = time_signatures[i]
566         abjad.mutate(shard).rewrite_meter(time_signature)
567
568 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
569     for i, shard in enumerate(abjad.mutate(voice[:]).split(
570         time_signatures)):
571         time_signature = time_signatures[i]
572         abjad.mutate(shard).rewrite_meter(time_signature)
573
574 print('Beaming runs ...')
575
576 for voice in abjad.select(score).components(abjad.Voice):
577     for run in abjad.select(voice).runs():
578         if 1 < len(run):
579             specifier = abjadext.rmakers.BeamSpecifier(
580                 beam_each_division=True,
581             )
582             specifier(abjad.select(run))
583             abjad.attach(abjad.StartBeam(), run[0])
584             abjad.attach(abjad.StopBeam(), run[-1])
585
586 print('Stopping Hairpins ...')
587 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.Staff):
588     for rest in abjad.iterate(staff).components(abjad.Rest):
589         previous_leaf = abjad.inspect(rest).leaf(-1)
590         if isinstance(previous_leaf, abjad.Note):
591             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
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', 'ordinario',
649                   'circular', 'circular', 'ordinario', 'ordinario', 'ordinario', 'ordinario',
650                   'ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario', 'jete',
651                   'ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario',
652                   'jete', 'jete', 'jete', 'jete', 'jete']
653     _apply_numerators_and_tech(staff=voice, nums=num_list, tech=
654                               tech_list)
655
656 for voice in abjad.select(score['Voice 3']).components(abjad.Voice):
657     seed(5)
658     nums_random_walk = []
659     nums_random_walk.append(-1 if random() < 0.5 else 1)
660     for i in range(1, 1000):
661         movement = -1 if random() < 0.5 else 1
662         value = nums_random_walk[i-1] + movement
663         nums_random_walk.append(value)
664     nums_random_walk = [abs(x) for x in nums_random_walk]
665     nums_chord = [0, 1, 2, 3, 4, 5, 4, 3, 2, 1]
666     num_list = [nums_chord[x] for x in reduceMod9(nums_random_walk)]
667     tech_list = ['ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario',
668                   'circular', 'circular', 'ordinario', 'ordinario', 'ordinario', 'ordinario',
669                   'ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario', 'jete',
670                   'ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario', 'ordinario',
671                   'jete', 'jete', 'jete', 'jete', 'jete']
672     _apply_numerators_and_tech(staff=voice, nums=num_list, tech=
673                               tech_list)
674
675 def _apply_position_and_span(staff, poses):
676     positions = cyc(poses)
677     for run in abjad.select(staff).runs():
678         span = abjad.StartTextSpan(
679             left_text=abjad.Markup(next(positions)).upright(),
680             right_text=abjad.Markup(next(positions)).upright(),
681             style='dashed-line-with-arrow',
682         )
683         abjad.attach(span, run[0])
684         abjad.attach(abjad.StopTextSpan(), run[-1])
685         abjad.override(staff).text_spinner.staff_padding = 0
686
687 for voice in abjad.select(score['Voice 5']).components(abjad.Voice):
688     pos_list_1 = ['st.', 'ord.', 'sp.', 'msp.', 'ord.']
689     _apply_position_and_span(staff=voice, poses=pos_list_1)
690
691 for voice in abjad.select(score['Voice 6']).components(abjad.Voice):
692     pos_list_2 = ['sp.', 'msp.', 'ord.', 'st.', 'ord.']
693     _apply_position_and_span(staff=voice, poses=pos_list_2)
694
695 for voice in abjad.select(score['Voice 1']).components(abjad.Voice):
696     for run in abjad.select(voice).runs():
697         specifier = abjadext.rmakers.BeamSpecifier(
698             beam_each_division=False,

```

```

689     )
690     specifier(run)
691
692 for voice in abjad.select(score['Voice 3']).components(abjad.Voice):
693     for run in abjad.select(voice).runs():
694         specifier = abjadext.rmakers.BeamSpecifier(
695             beam_each_division=False,
696             )
697         specifier(run)
698
699 instruments1 = cyc([
700     abjad.Cello(),
701 ])
702
703 instruments2 = cyc([
704     abjad.Cello(),
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.
744     Staff):
745     leaf1 = abjad.select(staff).leaves()[0]
746     abjad.attach(next(instruments1), leaf1)
747     abjad.attach(next(abbreviations1), leaf1)
748     abjad.attach(next(names1), leaf1)
749     abjad.attach(next(clefs1), leaf1)
750
751 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
752     Staff):
753     leaf1 = abjad.select(staff).leaves()[0]
754     abjad.attach(next(instruments2), leaf1)
755     abjad.attach(next(abbreviations2), leaf1)
756     abjad.attach(next(names2), leaf1)
757     abjad.attach(next(clefs2), leaf1)
758
759 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff
760     )[0]:
761     leaf1 = abjad.select(staff).leaves()[0]
762     last_leaf = abjad.select(staff).leaves()[-1]
763     abjad.attach(metro, leaf1)
764     abjad.attach(bar_line, last_leaf)
765
766 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff
767     )[0]:
768     leaf1 = abjad.select(staff).leaves()[0]
769     last_leaf = abjad.select(staff).leaves()[-1]
770     abjad.attach(metro, leaf1)
771     abjad.attach(bar_line, last_leaf)
772
773 for staff in abjad.iterate(score['Global Context']).components(abjad.
774     Staff):
775     leaf1_start = abjad.select(staff).leaves()[7]
776     leaf1 = abjad.select(staff).leaves()[8]
777     abjad.attach(mark1, leaf1)
778     abjad.attach(section_bar_line, leaf1_start)
779
780 for staff in abjad.iterate(score['Global Context']).components(abjad.
781     Staff):
782     leaf2_start = abjad.select(staff).leaves()[15]
783     leaf2 = abjad.select(staff).leaves()[16]
784     abjad.attach(mark2, leaf2)
785     abjad.attach(section_bar_line, leaf2_start)
786
787 for staff in abjad.iterate(score['Global Context']).components(abjad.
788     Staff):
789     leaf3_start = abjad.select(staff).leaves()[23]
790     leaf3 = abjad.select(staff).leaves()[24]
791     abjad.attach(mark3, leaf3)
792     abjad.attach(section_bar_line, leaf3_start)
793
794 for staff in abjad.iterate(score['Global Context']).components(abjad.
795     Staff):
796     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/_stylesheets/abjad.ily'],
802     )
803
804 abjad.SegmentMaker.comment_measure_numbers(score)
805 ######
806
807 directory = '/Users/evansdsg2/Scores/cthar/cthar/Segments/Segment_I'
808 pdf_path = f'{directory}/Segment_I.pdf'
809 path = pathlib.Path('Segment_I.pdf')
810 if path.exists():
811     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   \consists Mark_engraver
56   \consists Metronome_mark_engraver
57   \override BarNumber.Y-extent = #'(0 . 0)
58   \override BarNumber.Y-offset = 0
59   \override BarNumber.extra-offset = #'(-4 . 0)
60   %\override BarNumber.font-name = "Didot"
61   \override BarNumber.stencil = #(
62     make-stencil-boxer 0.1 0.7 ly:text-interface::print
63   )
64   \override BarNumber.fontSize = 1
65   \override BarNumber.padding = 4
66   \override MetronomeMark.X-extent = #'(0 . 0)
67   \override MetronomeMark.Y-extent = #'(0 . 0)
68   \override MetronomeMark.break-align-symbols = #'(left-edge)
69   \override MetronomeMark.extra-offset = #'(0 . 4)
70   \override MetronomeMark.fontSize = 10

```

```

71 \override RehearsalMark.stencil = #(
72     make-stencil-circler 0.1 0.7 ly:text-interface::print
73 )
74 \override RehearsalMark.X-extent = #'(0 . 0)
75 \override RehearsalMark.X-offset = 6
76 \override RehearsalMark.Y-offset = -2.25
77 \override RehearsalMark.break-align-symbols = #'(time-signature)
78 \override RehearsalMark.break-visibility = #end-of-line-invisible
79 \override RehearsalMark.font-name = "Didot"
80 \override RehearsalMark.font-size = 8
81 \override RehearsalMark.outside-staff-priority = 500
82 \override RehearsalMark.self-alignment-X = #center
83     \override TimeSignature.X-extent = #'(0 . 0)
84         \override TimeSignature.X-offset = #ly:self-alignment-interface
85         ::x-aligned-on-self
86             \override TimeSignature.Y-extent = #'(0 . 0)
87 \override TimeSignature.Y-offset = 3
88     \override TimeSignature.break-align-symbol = ##f
89     \override TimeSignature.break-visibility = #end-of-line-
90     invisible
91         \override TimeSignature.font-size = #7
92         \override TimeSignature.self-alignment-X = #center
93             \override VerticalAxisGroup.default-staff-staff-spacing = #'(
94                 (basic-distance . 0) (minimum-distance . 10) (padding . 6) (
95                 stretchability . 0)
96             )
97         }
98 \context {
99     \Score
100     \remove Bar_number_engraver

```

```

101   \override BarLine.bar-extent = #'(-2 . 2)
102     \override Beam.breakable = ##t
103   \override Beam.concaveness = #10000
104   \override Glissando.breakable = ##t
105   \override MetronomeMark.font-size = 5
106     \override SpacingSpanner.strict-grace-spacing = ##t
107     \override SpacingSpanner.strict-note-spacing = ##t
108     \override SpacingSpanner.uniform-stretching = ##t
109     \override StaffGrouper.staff-staff-spacing = #'(
110       (basic-distance . 0) (minimum-distance . 6) (padding . 2)
111     )
112     \override TupletBracket.bracket-visibility = ##t
113     \override TupletBracket.minimum-length = #3
114     \override TupletBracket.padding = #2
115     \override TupletBracket.springs-and-rods = #ly:spanner::set-
116       spacing-rods
117     \override TupletNumber.text = #tuplet-number::calc-fraction-text
118   \override TextSpanner.Y-offset = 1
119   proportionalNotationDuration = #(ly:make-moment 1 50)
120     autoBeaming = ##f
121     tupletFullLength = ##t
122   }
123   \context {
124     \Voice
125     \remove Forbid_line_break_engraver
126   }
127   \context {
128     \Staff
129     \remove Time_signature_engraver
130   }
131   \context {
132     \Staff
133     \name BowStaff

```

```

133     \type Engraver_group
134     \alias Staff
135     \bowtab
136     \override Beam.stencil = ##f
137     \override Dots.stencil = ##f
138     \override Flag.stencil = ##f
139     \override Glissando.bound-details.left.padding = #0.5
140     \override Glissando.bound-details.right.padding = #0.5
141     \override Glissando.thickness = #2
142     \override NoteHead.Y-offset = #-5
143     \override NoteHead.extra-offset = #'(0.05 . 0)
144     \override NoteHead.stencil = ##f
145     \override Rest.transparent = ##t
146         \override Script.staff-padding = #2
147         \override StaffSymbol.transparent = ##t
148         \override Stem.direction = #down
149         \override Stem.stencil = ##f
150         \override TimeSignature.stencil = ##f
151     \override Tie.stencil = ##f
152         \override TupletBracket.stencil = ##f
153         \override TupletNumber.stencil = ##f
154     %\RemoveEmptyStaves
155 }
156
157 \context {
158     \Staff
159     \name BeamStaff
160     \type Engraver_group
161     \alias Staff
162     \override Beam.direction = #down
163     \override Beam.positions = #'(5 . 5)
164     \override Clef.stencil = ##f
165     \override Dots.staff-position = #-2

```

```

166     \override Flag.Y-offset = #2.93
167     \override NoteHead.no-ledgers = ##t
168     \override NoteHead.stencil = ##f
169     \override Rest.transparent = ##t
170         \override Script.staff-padding = #3
171         \override StaffSymbol.transparent = ##t
172         \override Stem.direction = #down
173         \override Stem.length = #0.5
174         \override Stem.stem-begin-position = #15.975
175         \override TimeSignature.stencil = ##f
176     \override Tie.stencil = ##f
177         \override TupletBracket.positions = #'(3 . 3)
178 \%RemoveEmptyStaves
179 }
180
181 \context {
182     \RhythmicStaff
183     \remove Time_signature_engraver
184 }
185 \context {
186     \StaffGroup
187     \accepts BowStaff
188     \accepts BeamStaff
189 }
190 }

191
192 \paper {
193
194     top-margin = 1.5\cm
195     bottom-margin = 1.5\cm
196
197     %top-margin = .90\in
198     oddHeaderMarkup = \markup ""

```

```

199 evenHeaderMarkup = \markup ""
200 oddFooterMarkup = \markup \fill-line {
201   " "
202   \concat {
203     "C" \text{char} ~
204     \fontsize #2
205     \fromproperty #'page:page-number-string "~ Evans"
206   }
207   " "
208 }
209 evenFooterMarkup = \markup \fill-line {
210   " "
211   \concat { "C" \text{char} ~ \fontsize #2
212     \fromproperty #'page:page-number-string "~ Evans"
213   } ""
214 }
215 }
```

Code Example A.10: Cchar 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
29     time_signatures])
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
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 reduceMod13(rw):
47     return [(x % 14) for x in rw]
48
49 def reduceMod15(rw):
50     return [(x % 16) for x in rw]
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 = [8, 14, 23, 27, 28, 30, 37, 30, 28, 27, 23, 14, ]
61 flute_notes_one = [flute_chord_one[x] for x in reduceMod11(
62     flute_random_walk_one)]
63
64 seed(2)
65 clarinet_random_walk_one = []
66 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
67 for i in range(1, 1000):

```

```

66     movement = -1 if random() < 0.5 else 1
67     value = clarinet_random_walk_one[i-1] + movement
68     clarinet_random_walk_one.append(value)
69 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
70 clarinet_chord_one = [-3, 5, 8, 14, 23, 27, 23, 14, 8, 5, ]
71 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod9(
    clarinet_random_walk_one)]
72
73 seed(3)
74 bassoon_random_walk_one = []
75 bassoon_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 = bassoon_random_walk_one[i-1] + movement
79     bassoon_random_walk_one.append(value)
80 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
81 bassoon_chord_one = [-24, -14, -3, 5, 8, 5, -3, -14, ]
82 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod7(
    bassoon_random_walk_one)]
83
84 seed(4)
85 horn_random_walk_one = []
86 horn_random_walk_one.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 = 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(
    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)
113 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
114 trombone_chord_one = [-14, -3, 5, -3, ]
115 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod3(

```

```

    trombone_random_walk_one)]
116
117 seed(7)
118 tuba_random_walk_one = []
119 tuba_random_walk_one.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 = tuba_random_walk_one[i-1] + movement
123     tuba_random_walk_one.append(value)
124 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
125 tuba_chord_one = [-29, -24, -14, -3, 5, -3, -14, -24, ]
126 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod7(
127     tuba_random_walk_one)]
128
129 seed(8)
130 violin1_random_walk_one = []
131 violin1_random_walk_one.append(-1 if random() < 0.5 else 1)
132 for i in range(1, 1000):
133     movement = -1 if random() < 0.5 else 1
134     value = violin1_random_walk_one[i-1] + movement
135     violin1_random_walk_one.append(value)
136 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
137 violin1_chord_one = [-3, 5, 8, 14, 23, 27, 28, 30, 37, 30, 28, 27, 23,
138     14, 8, 5, ]
139 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod15(
140     violin1_random_walk_one)]
141
142 seed(9)
143 violin2_random_walk_one = []
144 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
145 for i in range(1, 1000):
146     movement = -1 if random() < 0.5 else 1
147     value = violin2_random_walk_one[i-1] + movement
148     violin2_random_walk_one.append(value)
149 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
150 violin2_chord_one = [-3, 5, 8, 14, 23, 27, 28, 27, 23, 14, 8, 5, ]
151 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod11(
152     violin2_random_walk_one)]
153
154 seed(10)
155 viola_random_walk_one = []
156 viola_random_walk_one.append(-1 if random() < 0.5 else 1)
157 for i in range(1, 1000):
158     movement = -1 if random() < 0.5 else 1
159     value = viola_random_walk_one[i-1] + movement
160     viola_random_walk_one.append(value)
161 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
162 viola_chord_one = [-3, 5, 8, 14, 23, 27, 28, 27, 23, 14, 8, 5, ]
163 viola_notes_one = [viola_chord_one[x] for x in reduceMod11(
164     viola_random_walk_one)]
165
166 seed(11)
167 cello_random_walk_one = []
168 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
173     seed(12)
174     bass_random_walk_one = []
175     bass_random_walk_one.append(-1 if random() < 0.5 else 1)
176     for i in range(1, 1000):
177         movement = -1 if random() < 0.5 else 1
178         value = bass_random_walk_one[i-1] + movement
179         bass_random_walk_one.append(value)
180     bass_random_walk_one = [abs(x) for x in bass_random_walk_one]
181     bass_chord_one = [-29, -24, -14, -3, -14, -24, ]
182     bass_notes_one = [bass_chord_one[x] for x in reduceMod5(
183         bass_random_walk_one)]
184
185     flute_scale = [30, 23, 5, 23, ]
186     clarinet_scale = [23, 5, ]
187     bassoon_scale = [-24, ]
188     horn_scale = [5, ]
189     trumpet_scale = [23, ]
190     trombone_scale = [5, ]
191     tuba_scale = [-24, ]
192
193     violin1_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25,
194         24.5, 24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5,
195         20, 20.5, 21, 21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26,
196         26.5, 27, 27.5, 28, 28.5, 29, 29.5, ]
197     violin2_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14,
198         13.5, 13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5,
199             10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16,
200             16.5, 17, 17.5, 18, 18.5, ]
201     viola_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5,
202         1, 0.5, 0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5,
203         0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]
204     cello_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8,
205         -8.5, -9, -9.5, -10, -10.5, -11, -11.5, -12, -12.5, -13, -13.5, -14,
206         -13.5, -13, -12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8,
207         -7.5, -7, -6.5, -6, -5.5, -5, -4.5, -4, -3.5, ]
208     bass_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18,
209         -18.5, -19, -19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5,
210         -24, -24.5, -25, -24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21,
211         -20.5, -20, -19.5, -19, -18.5, -18, -17.5, -17, -16.5, -16, -15.5,
212         -15, -14.5, ]
213
214     seed(1)
215     flute_random_walk_two = []
216     flute_random_walk_two.append(-1 if random() < 0.5 else 1)
217     for i in range(1, 1000):
218         movement = -1 if random() < 0.5 else 1
219
220

```

```

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(
206     flute_random_walk_two)]
207
208 seed(2)
209 clarinet_random_walk_two = []
210 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
211 for i in range(1, 1000):
212     movement = -1 if random() < 0.5 else 1
213     value = clarinet_random_walk_two[i-1] + movement
214     clarinet_random_walk_two.append(value)
215 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
216 clarinet_chord_two = [-5, 5, 10, 16, 23, 25, 26, 25, 23, 16, 10, 5, ]
217 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod11(
218     clarinet_random_walk_two)]
219
220 seed(3)
221 bassoon_random_walk_two = []
222 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
223 for i in range(1, 1000):
224     movement = -1 if random() < 0.5 else 1
225     value = bassoon_random_walk_two[i-1] + movement
226     bassoon_random_walk_two.append(value)
227 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
228 bassoon_chord_two = [-24, -16, -5, 5, -5, -16, ]
229 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod5(
230     bassoon_random_walk_two)]
231
232 seed(4)
233 horn_random_walk_two = []
234 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
235 for i in range(1, 1000):
236     movement = -1 if random() < 0.5 else 1
237     value = horn_random_walk_two[i-1] + movement
238     horn_random_walk_two.append(value)
239 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
240 horn_chord_two = [-16, -5, 5, 10, 5, -5, ]
241 horn_notes_two = [horn_chord_two[x] for x in reduceMod5(
242     horn_random_walk_two)]
243
244 seed(5)
245 trumpet_random_walk_two = []
246 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)
247 for i in range(1, 1000):
248     movement = -1 if random() < 0.5 else 1
249     value = trumpet_random_walk_two[i-1] + movement
250     trumpet_random_walk_two.append(value)
251 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]
252 trumpet_chord_two = [-5, 5, 10, 16, 23, 16, 10, 5, ]
253 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod7(
254     trumpet_random_walk_two)]

```

```

250
251 seed(6)
252 trombone_random_walk_two = []
253 trombone_random_walk_two.append(-1 if random() < 0.5 else 1)
254 for i in range(1, 1000):
255     movement = -1 if random() < 0.5 else 1
256     value = trombone_random_walk_two[i-1] + movement
257     trombone_random_walk_two.append(value)
258 trombone_random_walk_two = [abs(x) for x in trombone_random_walk_two]
259 trombone_chord_two = [-16, -5, 5, -5, ]
260 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod3(
261     trombone_random_walk_two)]
262
263 seed(7)
264 tuba_random_walk_two = []
265 tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
266 for i in range(1, 1000):
267     movement = -1 if random() < 0.5 else 1
268     value = tuba_random_walk_two[i-1] + movement
269     tuba_random_walk_two.append(value)
270 tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
271 tuba_chord_two = [-27, -24, -16, -5, -16, -24, ]
272 tuba_notes_two = [tuba_chord_two[x] for x in reduceMod5(
273     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     value = violin1_random_walk_two[i-1] + movement
281     violin1_random_walk_two.append(value)
282 violin1_random_walk_two = [abs(x) for x in violin1_random_walk_two]
283 violin1_chord_two = [-5, 5, 10, 16, 23, 25, 26, 30, 38, 30, 26, 25, 23,
284     16, 10, 5, ]
285 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod15(
286     violin1_random_walk_two)]
287
288 seed(9)
289 violin2_random_walk_two = []
290 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
291 for i in range(1, 1000):
292     movement = -1 if random() < 0.5 else 1
293     value = violin2_random_walk_two[i-1] + movement
294     violin2_random_walk_two.append(value)
295 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
296 violin2_chord_two = [-5, 5, 10, 16, 23, 25, 26, 25, 23, 16, 10, 5, ]
297 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod11(
298     violin2_random_walk_two)]
299
300 seed(10)
301 viola_random_walk_two = []
302 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
303 for i in range(1, 1000):

```

```

299 movement = -1 if random() < 0.5 else 1
300 value = viola_random_walk_two[i-1] + movement
301 viola_random_walk_two.append(value)
302 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
303 viola_chord_two = [-5, 5, 10, 16, 23, 16, 10, 5, ]
304 viola_notes_two = [viola_chord_two[x] for x in reduceMod7(
    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):
    movement = -1 if random() < 0.5 else 1
    value = cello_random_walk_two[i-1] + movement
    cello_random_walk_two.append(value)
310 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
311 cello_chord_two = [-24, -16, -5, 5, 10, 16, 23, 16, 10, 5, -5, -16]
312 cello_notes_two = [cello_chord_two[x] for x in reduceMod11(
    cello_random_walk_two)]
313
314 seed(12)
315 bass_random_walk_two = []
316 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
317 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)
318 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
319 bass_chord_two = [-27, -24, -16, -5, -16, -24, ]
320 bass_notes_two = [bass_chord_two[x] for x in reduceMod5(
    bass_random_walk_two)]
321
322 rmaker_one = abjadext.rmakers.NoteRhythmMaker()
323
324 rmaker_two = abjadext.rmakers.EvenDivisionRhythmMaker(
    denominators=[16, 16, 8, 16, 4, 16, 8],
    extra_counts_per_division=[0, 1, 0, 0, -1, 0, 1, -1],
    burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
        left_classes=[abjad.Rest],
        left_counts=[1],
        right_classes=[abjad.Rest],
        right_counts=[1],
        outer_divisions_only=True,
    ),
    tupletSpecifier=abjadext.rmakers.TupletSpecifier(
        trivialize=True,
        extract_trivial=True,
        rewrite_rest_filled=True,
    ),
)
325
326 rmaker_three = abjadext.rmakers.TaleaRhythmMaker(
    talea=abjadext.rmakers.Talea(
        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
382 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,

```

```

$12     pitches=bassoon_notes_two,
$13     continuous=True,
$14     attachment_handler=attachment_handler_two,
$15   )
$16 bassoonmusicmaker_three = MusicMaker(
$17     rmaker=rmaker_three,
$18     pitches=bassoon_notes_one,
$19     continuous=True,
$20     attachment_handler=attachment_handler_three,
$21   )
$22 #####trombone#####
$23 trombonemusicmaker_one = MusicMaker(
$24     rmaker=rmaker_one,
$25     pitches=trombone_scale,
$26     continuous=True,
$27     attachment_handler=attachment_handler_one,
$28   )
$29 trombonemusicmaker_two = MusicMaker(
$30     rmaker=rmaker_two,
$31     pitches=trombone_notes_two,
$32     continuous=True,
$33     attachment_handler=attachment_handler_two,
$34   )
$35 trombonemusicmaker_three = MusicMaker(
$36     rmaker=rmaker_three,
$37     pitches=trombone_notes_one,
$38     continuous=True,
$39     attachment_handler=attachment_handler_three,
$40   )
$41 #####cello#####
$42 celломusicmaker_one = MusicMaker(
$43     rmaker=rmaker_one,
$44     pitches=cello_scale,
$45     continuous=True,
$46     attachment_handler=attachment_handler_one,
$47   )
$48 celломusicmaker_two = MusicMaker(
$49     rmaker=rmaker_two,
$50     pitches=cello_notes_two,
$51     continuous=True,
$52     attachment_handler=attachment_handler_two,
$53   )
$54 celломusicmaker_three = MusicMaker(
$55     rmaker=rmaker_three,
$56     pitches=cello_notes_one,
$57     continuous=True,
$58     attachment_handler=attachment_handler_three,
$59   )
$60 #####horn#####
$61 hornmusicmaker_one = MusicMaker(
$62     rmaker=rmaker_one,
$63     pitches=horn_scale,
$64     continuous=True,
$65     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_timespan_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_timespan_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
1359     Context 1'),
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
1372     Context 2'),
1373     abjad.StaffGroup(
1374       [
1375         abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
1376

```

```

lilypond_type='Staff',),
1371     abjad.Staff([abjad.Voice(name='Voice 5')],name='Staff 5',
lilypond_type='Staff',),
1372     abjad.Staff([abjad.Voice(name='Voice 6')],name='Staff 6',
lilypond_type='Staff',),
1373     abjad.Staff([abjad.Voice(name='Voice 7')],name='Staff 7',
lilypond_type='Staff',),
1374     ],
1375     name='Staff Group 2',
1376   ),
1377   abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
1378 Context 3'),
1379   abjad.StaffGroup(
1380   [
1381     abjad.Staff([abjad.Voice(name='Voice 8')],name='Staff 8',
lilypond_type='Staff',),
1382     abjad.Staff([abjad.Voice(name='Voice 9')],name='Staff 9',
lilypond_type='Staff',),
1383     abjad.Staff([abjad.Voice(name='Voice 10')],name='Staff 10',
lilypond_type='Staff',),
1384     abjad.Staff([abjad.Voice(name='Voice 11')],name='Staff 11',
lilypond_type='Staff',),
1385     abjad.Staff([abjad.Voice(name='Voice 12')],name='Staff 12',
lilypond_type='Staff',),
1386     ],
1387     name='Staff Group 3',
1388   )
1389 )
1390
1391 for time_signature in time_signatures:
1392   skip = abjad.Skip(1, multiplier=(time_signature))
1393   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(
1431         time_signatures)):
1432         time_signature = time_signatures[i]
1433         abjad.mutate(shard).rewrite_meter(time_signature)
1434
1435 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
1436     for i, shard in enumerate(abjad.mutate(voice[:]).split(
1437         time_signatures)):
1438         time_signature = time_signatures[i]
1439         abjad.mutate(shard).rewrite_meter(time_signature)
1440
1441 for voice in abjad.iterate(score['Staff Group 3']).components(abjad.Voice):
1442     for i, shard in enumerate(abjad.mutate(voice[:]).split(
1443         time_signatures)):
1444         time_signature = time_signatures[i]
1445         abjad.mutate(shard).rewrite_meter(time_signature)
1446
1447 print('Beaming runs ...')
1448
1449 for voice in abjad.select(score).components(abjad.Voice):
1450     for run in abjad.select(voice).runs():
1451         if 1 < len(run):
1452             specifier = abjadext.rmakers.BeamSpecifier(
1453                 beam_each_division=False,
1454             )
1455             specifier(run)
1456             abjad.attach(abjad.StartBeam(), run[0])
1457             abjad.attach(abjad.StopBeam(), run[-1])
1458             for leaf in run:
1459                 if abjad.Duration(1, 4) <= leaf.written_duration:
1460                     continue
1461                 previous_leaf = abjad.inspect(leaf).leaf(-1)
1462                 next_leaf = abjad.inspect(leaf).leaf(1)
1463                 if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
1464                     abjad.Duration(1, 4) <= next_leaf.written_duration):
1465                     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)))
1471 and
1472     abjad.Duration(1, 4) <= previous_leaf.
1473 written_duration):
1474     left = leaf.written_duration.flag_count
1475     right = next_leaf.written_duration.flag_count
1476     beam_count = abjad.BeamCount(
1477         left=left,
1478         right=right,
1479         )
1480     abjad.attach(beam_count, leaf)
1481
1480 print('Beautifying score ...')
1481 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
1482 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 =
1486             #1 \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
1494 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1495 Staff):
1495     for selection in abjad.select(staff).components(abjad.Rest).
1496 group_by_contiguity():
1497         start_command = abjad.LilyPondLiteral(
1498             r'\stopStaff \once \override Staff.StaffSymbol.line-count =
1499             #1 \startStaff',
1500             format_slot='before',
1501             )
1502         stop_command = abjad.LilyPondLiteral(
1503             r'\stopStaff \startStaff',
1504             format_slot='after',
1505             )
1506         abjad.attach(start_command, selection[0])
1507         abjad.attach(stop_command, selection[-1])
1508
1507 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1508 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 =
1512                 #1 \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.
1524         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 2']).components(abjad.
1535         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     for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1546         Staff):
1547         for rest in abjad.iterate(staff).components(abjad.Rest):
1548             previous_leaf = abjad.inspect(rest).leaf(-1)
1549             if isinstance(previous_leaf, abjad.Note):
1550                 abjad.attach(abjad.StopHairpin(), rest)
1551             elif isinstance(previous_leaf, abjad.Chord):
1552                 abjad.attach(abjad.StopHairpin(), rest)
1553             elif isinstance(previous_leaf, abjad.Rest):
1554                 pass
1555
1556     print('Adding pitch material ...')
1557     def cyc(lst):
1558         count = 0
1559         while True:
1560             yield lst[count%len(lst)]
1561             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.
1666     Staff):
1667     leaf1 = abjad.select(staff).leaves()[0]
1668     abjad.attach(next(instruments2), leaf1)
1669     abjad.attach(next(abbreviations2), leaf1)
1670     abjad.attach(next(names2), leaf1)
1671     abjad.attach(next(clefs2), leaf1)
1672
1673 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1674     Staff):
1675     leaf1 = abjad.select(staff).leaves()[0]
1676     abjad.attach(next(instruments3), leaf1)
1677     abjad.attach(next(abbreviations3), leaf1)
1678     abjad.attach(next(names3), leaf1)
1679     abjad.attach(next(clefs3), leaf1)
1680
1681 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff
1682     )[0]:
1683     leaf1 = abjad.select(staff).leaves()[0]
1684     last_leaf = abjad.select(staff).leaves()[-1]
1685     abjad.attach(metro, leaf1)
1686     abjad.attach(bar_line, last_leaf)
1687
1688 for staff in abjad.select(score['Staff Group 2']).components(abjad.Staff
1689     )[0]:
1690     leaf1 = abjad.select(staff).leaves()[0]
1691     last_leaf = abjad.select(staff).leaves()[-1]
1692     abjad.attach(metro, leaf1)
1693     abjad.attach(bar_line, last_leaf)
1694
1695 for staff in abjad.select(score['Staff Group 3']).components(abjad.Staff
1696     )[0]:
1697     leaf1 = abjad.select(staff).leaves()[0]
1698     last_leaf = abjad.select(staff).leaves()[-1]
1699     abjad.attach(metro, leaf1)
1700     abjad.attach(bar_line, last_leaf)
1701
1702 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1703     Staff):
1704     leaf1 = abjad.select(staff).leaves()[7]
1705     abjad.attach(mark1, leaf1)
1706
1707 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1708     Staff):
1709     leaf1 = abjad.select(staff).leaves()[7]
1710     abjad.attach(mark1, leaf1)
1711
1712 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1713     Staff):
1714     leaf1 = abjad.select(staff).leaves()[7]
1715     abjad.attach(mark1, leaf1)
1716
1717 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     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1713         Staff):
1714         leaf2 = abjad.select(staff).leaves()[16]
1715         abjad.attach(mark2, leaf2)

1716     for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1717         Staff):
1718         leaf2 = abjad.select(staff).leaves()[16]
1719         abjad.attach(mark2, leaf2)

1720     for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1721         Staff):
1722         leaf3 = abjad.select(staff).leaves()[22]
1723         abjad.attach(mark3, leaf3)

1724     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1725         Staff):
1726         leaf3 = abjad.select(staff).leaves()[22]
1727         abjad.attach(mark3, leaf3)

1728     for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1729         Staff):
1730         leaf3 = abjad.select(staff).leaves()[22]
1731         abjad.attach(mark3, leaf3)

1732     for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1733         Staff):
1734         leaf4 = abjad.select(staff).leaves()[29]
1735         abjad.attach(mark4, leaf4)

1736     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1737         Staff):
1738         leaf4 = abjad.select(staff).leaves()[29]
1739         abjad.attach(mark4, leaf4)

1740     for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1741         Staff):
1742         leaf4 = abjad.select(staff).leaves()[29]
1743         abjad.attach(mark4, leaf4)

1744     for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1745         Staff):
1746         leaf5 = abjad.select(staff).leaves()[34]
1747         abjad.attach(mark5, leaf5)

1748     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1749         Staff):
1750         leaf5 = abjad.select(staff).leaves()[34]
1751         abjad.attach(mark5, leaf5)

```

```

1753 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1754     Staff):
1755     leaf5 = abjad.select(staff).leaves() [34]
1756     abjad.attach(mark5, leaf5)
1757
1758 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1759     Staff):
1760     leaf6 = abjad.select(staff).leaves() [39]
1761     abjad.attach(mark6, leaf6)
1762
1763 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1764     Staff):
1765     leaf6 = abjad.select(staff).leaves() [39]
1766     abjad.attach(mark6, leaf6)
1767
1768 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
1769     Staff):
1770     abjad.Instrument.transpose_from_sounding_pitch(staff)
1771
1772 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1773     Staff):
1774     abjad.Instrument.transpose_from_sounding_pitch(staff)
1775
1776 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1777     Staff):
1778     abjad.Instrument.transpose_from_sounding_pitch(staff)
1779
1780 score_file = abjad.LilyPondFile.new(
1781     score,
1782     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/
1783         source/_stylesheets/abjad.ily'],
1784     )
1785
1786 abjad.SegmentMaker.comment_measure_numbers(score)
1787 ##########
1788
1789 directory = '/Users/evansdsg2/Scores//tianshu/tianshu/Segments/Segment_I'
1790
1791 pdf_path = f'{directory}/Segment_I.pdf'
1792 path = pathlib.Path('Segment_I.pdf')
1793 if path.exists():
1794     print(f'Removing {pdf_path} ...')
1795     path.unlink()
1796 time_1 = time.time()
1797 print(f'Persisting {pdf_path} ...')
1798 result = abjad.persist(score_file).as_pdf(pdf_path)
1799 print(result[0])
1800 print(result[1])
1801 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), (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
29     time_signatures])
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 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 reduceMod15(rw):
46     return [(x % 16) 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, 10, 16, 18, 25, 26, 25, 18, 16, 10 ]
57 flute_notes_one = [flute_chord_one[x] for x in reduceMod9(
58     flute_random_walk_one)]
59
60 seed(2)
61 clarinet_random_walk_one = []
62 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
63 for i in range(1, 1000):
64     movement = -1 if random() < 0.5 else 1
65     value = clarinet_random_walk_one[i-1] + movement
66     clarinet_random_walk_one.append(value)
67 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
68 clarinet_chord_one = [-5, 0, 10, 16, 18, 25, 18, 16, 10, 0 ]
69 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod9(
70     clarinet_random_walk_one)]
71
72 seed(3)
73 bassoon_random_walk_one = []
74 bassoon_random_walk_one.append(-1 if random() < 0.5 else 1)
75 for i in range(1, 1000):
76     movement = -1 if random() < 0.5 else 1
77     value = bassoon_random_walk_one[i-1] + movement
78     bassoon_random_walk_one.append(value)
79 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
80 bassoon_chord_one = [-19, -16, -5, 0, 10, 0, -5, -16 ]
81 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod7(
82     bassoon_random_walk_one)]
83
84 seed(4)
85 horn_random_walk_one = []
86 horn_random_walk_one.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 = 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 = [-19, -16, -5, 0, -5, -16 ]

```

```

90 horn_notes_one = [horn_chord_one[x] for x in reduceMod5(
91     horn_random_walk_one)]
92
93 seed(5)
94 trumpet_random_walk_one = []
95 trumpet_random_walk_one.append(-1 if random() < 0.5 else 1)
96 for i in range(1, 1000):
97     movement = -1 if random() < 0.5 else 1
98     value = trumpet_random_walk_one[i-1] + movement
99     trumpet_random_walk_one.append(value)
100 trumpet_random_walk_one = [abs(x) for x in trumpet_random_walk_one]
101 trumpet_chord_one = [-5, 0, 10, 16, 18, 16, 10, 0 ]
102 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod7(
103     trumpet_random_walk_one)]
104
105 seed(6)
106 trombone_random_walk_one = []
107 trombone_random_walk_one.append(-1 if random() < 0.5 else 1)
108 for i in range(1, 1000):
109     movement = -1 if random() < 0.5 else 1
110     value = trombone_random_walk_one[i-1] + movement
111     trombone_random_walk_one.append(value)
112 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
113 trombone_chord_one = [-19, -16, -5, 0, -5, -16 ]
114 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod5(
115     trombone_random_walk_one)]
116
117 seed(7)
118 tuba_random_walk_one = []
119 tuba_random_walk_one.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 = tuba_random_walk_one[i-1] + movement
123     tuba_random_walk_one.append(value)
124 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
125 tuba_chord_one = [-27, -19, -16, -5, -16, -19 ]
126 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod5(
127     tuba_random_walk_one)]
128
129 seed(8)
130 violin1_random_walk_one = []
131 violin1_random_walk_one.append(-1 if random() < 0.5 else 1)
132 for i in range(1, 1000):
133     movement = -1 if random() < 0.5 else 1
134     value = violin1_random_walk_one[i-1] + movement
135     violin1_random_walk_one.append(value)
136 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
137 violin1_chord_one = [-5, 0, 10, 16, 18, 25, 26, 34, 38, 34, 26, 25, 18,
138     16, 10, 0 ]
139 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod15(
140     violin1_random_walk_one)]
141
142 seed(9)
143 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 = [-5, 0, 10, 16, 18, 25, 26, 25, 18, 16, 10, 0 ]
145 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod11(
146     violin2_random_walk_one)]
147
148 seed(10)
149 viola_random_walk_one = []
150 for i in range(1, 1000):
151     movement = -1 if random() < 0.5 else 1
152     value = viola_random_walk_one[i-1] + movement
153     viola_random_walk_one.append(value)
154 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
155 viola_chord_one = [-5, 0, 10, 16, 18, 16, 10, 0 ]
156 viola_notes_one = [viola_chord_one[x] for x in reduceMod7(
157     viola_random_walk_one)]
158
159 seed(11)
160 cello_random_walk_one = []
161 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
162 for i in range(1, 1000):
163     movement = -1 if random() < 0.5 else 1
164     value = cello_random_walk_one[i-1] + movement
165     cello_random_walk_one.append(value)
166 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
167 cello_chord_one = [-19, -16, -5, 0, 10, 16, 18, 16, 10, 0, -5, -16 ]
168 cello_notes_one = [cello_chord_one[x] for x in reduceMod11(
169     cello_random_walk_one)]
170
171 seed(12)
172 bass_random_walk_one = []
173 bass_random_walk_one.append(-1 if random() < 0.5 else 1)
174 for i in range(1, 1000):
175     movement = -1 if random() < 0.5 else 1
176     value = bass_random_walk_one[i-1] + movement
177     bass_random_walk_one.append(value)
178 bass_random_walk_one = [abs(x) for x in bass_random_walk_one]
179 bass_chord_one = [-27, -19, -16, -5, 0, -5, -19 ]
180 bass_notes_one = [bass_chord_one[x] for x in reduceMod6(
181     bass_random_walk_one)]
182
183 flute_scale = [18 ]
184 clarinet_scale = [0 ]
185 bassoon_scale = [-19 ]
186 horn_scale = [-19 ]
187 trumpet_scale = [18 ]
188 trombone_scale = [-0 ]
189 tuba_scale = [-19 ]
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, ]
188 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, ]
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, 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, ]
190 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, ]
191 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, ]
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(
    flute_random_walk_two)]
203
204 seed(2)
205 clarinet_random_walk_two = []
206 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
207 for i in range(1, 1000):
208     movement = -1 if random() < 0.5 else 1
209     value = clarinet_random_walk_two[i-1] + movement
210     clarinet_random_walk_two.append(value)
211 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
212 clarinet_chord_two = [-3, 0, 8, 14, 18, 14, 8, 0]
213 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod7(
    clarinet_random_walk_two)]
214
215 seed(3)
216 bassoon_random_walk_two = []
217 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
218 for i in range(1, 1000):
219     movement = -1 if random() < 0.5 else 1
220     value = bassoon_random_walk_two[i-1] + movement
221     bassoon_random_walk_two.append(value)
222 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
223 bassoon_chord_two = [-19, -14, -3, 0, 8, 0, -3, -14]
224 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod7(

```

```

bassoon_random_walk_two)]  

225  

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 = [-19, -14, -3, 0, 8, 0, -3, -14 ]  

235 horn_notes_two = [horn_chord_two[x] for x in reduceMod7(  

    horn_random_walk_two)]  

236  

237 seed(5)  

238 trumpet_random_walk_two = []  

239 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)  

240 for i in range(1, 1000):  

241     movement = -1 if random() < 0.5 else 1  

242     value = trumpet_random_walk_two[i-1] + movement  

243     trumpet_random_walk_two.append(value)  

244 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]  

245 trumpet_chord_two = [-3, 0, 8, 14, 18, 14, 8, 0 ]  

246 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod7(  

    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 = [-19, -14, -3, 0, -3, -14 ]  

257 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod5(  

    trombone_random_walk_two)]  

258  

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,
279   28, 35 ]
280 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod15(
281   violin1_random_walk_two)]
282
283 seed(9)
284 violin2_random_walk_two = []
285 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
286 for i in range(1, 1000):
287     movement = -1 if random() < 0.5 else 1
288     value = violin2_random_walk_two[i-1] + movement
289     violin2_random_walk_two.append(value)
290 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
291 violin2_chord_two = [-3, 0, 8, 14, 18, 27, 18, 14, 8, 0 ]
292 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod9(
293   violin2_random_walk_two)]
294
295 seed(10)
296 viola_random_walk_two = []
297 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
298 for i in range(1, 1000):
299     movement = -1 if random() < 0.5 else 1
300     value = viola_random_walk_two[i-1] + movement
301     viola_random_walk_two.append(value)
302 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
303 viola_chord_two = [-3, 0, 8, 14, 18, 14, 8, 0 ]
304 viola_notes_two = [viola_chord_two[x] for x in reduceMod7(
305   viola_random_walk_two)]
306
307 seed(11)
308 cello_random_walk_two = []
309 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
310 for i in range(1, 1000):
311     movement = -1 if random() < 0.5 else 1
312     value = cello_random_walk_two[i-1] + movement
313     cello_random_walk_two.append(value)
314 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
315 cello_chord_two = [-19, -14, -3, 0, 8, 14, 18, 14, 8, 0, -3, -14 ]
316 cello_notes_two = [cello_chord_two[x] for x in reduceMod11(
317   cello_random_walk_two)]
318
319 seed(12)
320 bass_random_walk_two = []
321 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
322 for i in range(1, 1000):
323     movement = -1 if random() < 0.5 else 1
324     value = bass_random_walk_two[i-1] + movement
325     bass_random_walk_two.append(value)
326 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
327 bass_chord_two = [-29, -19, -14, -3, 0, -3, -14, -19 ]

```

```

323 bass_notes_two = [bass_chord_two[x] for x in reduceMod7(
324     bass_random_walk_two)]
325
325 rmaker_one = abjadext.rmakers.EvenDivisionRhythmMaker(
326     denominators=[16, 16, 8, 16, 4, 16, 8],
327     extra_counts_per_division=[1, 1, 0, -1, 0, 1, -1, 0],
328     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
329         left_classes=[abjad.Rest],
330         left_counts=[2],
331         right_classes=[abjad.Rest],
332         right_counts=[1],
333         outer_divisions_only=True,
334         ),
335     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
336         trivialize=True,
337         extract_trivial=True,
338         rewrite_rest_filled=True,
339         ),
340     )
341
342 rmaker_two = abjadext.rmakers.IncisedRhythmMaker(
343     inciseSpecifier=abjadext.rmakers.InciseSpecifier(
344         prefix_talea=[-1],
345         prefix_counts=[0, 1],
346         suffix_talea=[-1],
347         suffix_counts=[1],
348         talea_denominator=16,
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 #####oboe#####
382 flutemusicmaker_one = MusicMaker(
383     rmaker=rmaker_one,
384     pitches=flute_scale,
385     continuous=True,
386     attachment_handler=attachment_handler_one,
387 )
388 flutemusicmaker_two = MusicMaker(
389     rmaker=rmaker_two,
390     pitches=flute_notes_two,
391     continuous=True,
392     attachment_handler=attachment_handler_two,
393 )
394 flutemusicmaker_three = MusicMaker(
395     rmaker=rmaker_three,
396     pitches=flute_notes_one,
397     continuous=True,
398     attachment_handler=attachment_handler_three,
399 )
400 #####violin1#####
401 violin1musicmaker_one = MusicMaker(
402     rmaker=rmaker_one,
403     pitches=violin1_scale,
404     continuous=True,
405     attachment_handler=attachment_handler_one,
406 )
407 violin1musicmaker_two = MusicMaker(
408     rmaker=rmaker_two,
409     pitches=violin1_notes_two,
410     continuous=True,
411     attachment_handler=attachment_handler_two,
412 )
413 violin1musicmaker_three = MusicMaker(
414     rmaker=rmaker_three,
415     pitches=violin1_notes_one,
416     continuous=True,
417     attachment_handler=attachment_handler_three,
418 )
419 #####trumpet#####
420 trumpetmusicmaker_one = MusicMaker(
421     rmaker=rmaker_one,
422     pitches=trumpet_scale,
423     continuous=True,
424     attachment_handler=attachment_handler_one,
425 )
426 trumpetmusicmaker_two = MusicMaker(
427     rmaker=rmaker_two,
428     pitches=trumpet_notes_two,
429

```

```
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 )
```



```

538     continuous=True,
539     attachment_handler=attachment_handler_one,
540 )
541 celломusicmaker_two = MusicMaker(
542     rmaker=rmaker_two,
543     pitches=cello_notes_two,
544     continuous=True,
545     attachment_handler=attachment_handler_two,
546 )
547 celломusicmaker_three = MusicMaker(
548     rmaker=rmaker_three,
549     pitches=cello_notes_one,
550     continuous=True,
551     attachment_handler=attachment_handler_three,
552 )
553 #####horn#####
554 hornmusicmaker_one = MusicMaker(
555     rmaker=rmaker_one,
556     pitches=horn_scale,
557     continuous=True,
558     attachment_handler=attachment_handler_one,
559 )
560 hornmusicmaker_two = MusicMaker(
561     rmaker=rmaker_two,
562     pitches=horn_notes_two,
563     continuous=True,
564     attachment_handler=attachment_handler_two,
565 )
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_timespan_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_timespan_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_timespan_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(
1200         start_offset=start_offset,
1201         stop_offset=stop_offset,
1202         annotation=MusicSpecifier(
1203             music_maker=music_maker,
1204             voice_name='Voice 7',
1205         ),
1206     )
1207     for start_offset, stop_offset, music_maker in [
1208         [(0, 4), (5, 4), tubamusicmaker_one],
1209         [(8, 4), (10, 4), tubamusicmaker_two],
1210         [(14, 4), (18, 4), tubamusicmaker_three],
1211         [(21, 4), (22, 4), tubamusicmaker_one],
1212         [(22, 4), (23, 4), tubamusicmaker_one],
1213         [(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())
1322 )
1323
1324 for voice_name, timespan_list in all_timespan_lists.items():
1325     silences = abjad.TimespanList([global_timespan])
1326     silences.extend(timespan_list)
1327     silences.sort()
1328     silences.compute_logical_xor()
1329     for silence_timespan in silences:
1330         timespan_list.append(
1331             abjad.AnnotatedTimespan(
1332                 start_offset=silence_timespan.start_offset,
1333                 stop_offset=silence_timespan.stop_offset,
1334                 annotation=MusicSpecifier(
1335                     music_maker=None,
1336                     voice_name=voice_name,
1337                     ),
1338                 ),
1339             )
1340     timespan_list.sort()
1341
1342 for voice_name, timespan_list in all_timespan_lists.items():
1343     shards = timespan_list.split_at_offsets(bounds)
1344     split_timespan_list = abjad.TimespanList()
1345     for shard in shards:
1346         split_timespan_list.extend(shard)
1347     split_timespan_list.sort()

```

```

1348     all_timespan_lists[voice_name] = timespan_list
1349
1350 score = abjad.Score([
1351     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
1352     Context 1'),
1353     abjad.StaffGroup(
1354         [
1355             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
1356             lilypond_type='Staff',),
1357             abjad.Staff([abjad.Voice(name='Voice 2')], name='Staff 2',
1358             lilypond_type='Staff',),
1359             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
1360             lilypond_type='Staff',),
1361             ],
1362             name='Staff Group 1',
1363         ),
1364         abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
1365     Context 2'),
1366     abjad.StaffGroup(
1367         [
1368             abjad.Staff([abjad.Voice(name='Voice 4')], name='Staff 4',
1369             lilypond_type='Staff',),
1370             abjad.Staff([abjad.Voice(name='Voice 5')], name='Staff 5',
1371             lilypond_type='Staff',),
1372             abjad.Staff([abjad.Voice(name='Voice 6')], name='Staff 6',
1373             lilypond_type='Staff',),
1374             abjad.Staff([abjad.Voice(name='Voice 7')], name='Staff 7',
1375             lilypond_type='Staff',),
1376             ],
1377             name='Staff Group 2',
1378         ),
1379         abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
1380     Context 3'),
1381     abjad.StaffGroup(
1382         [
1383             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1384             lilypond_type='Staff',),
1385             abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1386             lilypond_type='Staff',),
1387             abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1388             lilypond_type='Staff',),
1389             abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1390             lilypond_type='Staff',),
1391             abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1392             lilypond_type='Staff',),
1393             ],
1394             name='Staff Group 3',
1395         )
1396     ],
1397 )
1398
1399 for time_signature in time_signatures:
1400     skip = abjad.Skip(1, multiplier=(time_signature))
1401     abjad.attach(time_signature, skip)

```

```

1387     score['Global Context 1'].append(skip)
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 2'].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 3'].append(skip)
1398
1399 print('Making containers ...')
1400
1401 def make_container(music_maker, durations):
1402     selections = music_maker(durations)
1403     container = abjad.Container([])
1404     container.extend(selections)
1405     return container
1406
1407 def key_function(timespan):
1408     return timespan.annotation.music_maker or silence_maker
1409
1410 for voice_name, timespan_list in all_timespan_lists.items():
1411     for music_maker, grouper in itertools.groupby(
1412         timespan_list,
1413         key=key_function,
1414     ):
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(
1424         time_signatures)):
1425         time_signature = time_signatures[i]
1426         abjad.mutate(shard).rewrite_meter(time_signature)
1427
1428 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.Voice):
1429     for i, shard in enumerate(abjad.mutate(voice[:]).split(
1430         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 3']).components(abjad.Voice):
1435     for i, shard in enumerate(abjad.mutate(voice[:]).split(
1436         time_signatures)):
1437         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.
1465                     written_duration):
1466                     left = leaf.written_duration.flag_count
1467                     right = next_leaf.written_duration.flag_count
1468                     beam_count = abjad.BeamCount(
1469                         left=left,
1470                         right=right,
1471                         )
1472                     abjad.attach(beam_count, leaf)
1473
1474 print('Beautifying score ...')
1475 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
1476     Staff):
1477     for selection in abjad.select(staff).components(abjad.Rest).
1478         group_by_contiguity():
1479             start_command = abjad.LilyPondLiteral(
1480                 r'\stopStaff \once \override Staff.StaffSymbol.line-count =
1481                 #1 \startStaff',
1482                 format_slot='before',
1483                 )
1484             stop_command = abjad.LilyPondLiteral(
1485                 r'\stopStaff \startStaff',
1486                 format_slot='after',
1487                 )

```

```

1484     abjad.attach(start_command, selection[0])
1485     abjad.attach(stop_command, selection[-1])
1486
1487 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1488     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 =
1493             #1 \startStaff',
1494             format_slot='before',
1495             )
1496         stop_command = abjad.LilyPondLiteral(
1497             r'\stopStaff \startStaff',
1498             format_slot='after',
1499             )
1500     abjad.attach(start_command, selection[0])
1501     abjad.attach(stop_command, selection[-1])
1502
1503 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1504     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 =
1509             #1 \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.
1521     Staff):
1522     for rest in abjad.iterate(staff).components(abjad.Rest):
1523         previous_leaf = abjad.inspect(rest).leaf(-1)
1524         if isinstance(previous_leaf, abjad.Note):
1525             abjad.attach(abjad.StopHairpin(), rest)
1526         elif isinstance(previous_leaf, abjad.Chord):
1527             abjad.attach(abjad.StopHairpin(), rest)
1528         elif isinstance(previous_leaf, abjad.Rest):
1529             pass
1530
1531 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1532     Staff):
1533     for rest in abjad.iterate(staff).components(abjad.Rest):
1534         previous_leaf = abjad.inspect(rest).leaf(-1)
1535         if isinstance(previous_leaf, abjad.Note):
1536             abjad.attach(abjad.StopHairpin(), rest)
1537         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.
1535     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 ...')
1546 def cyc(lst):
1547     count = 0
1548     while True:
1549         yield lst[count%len(lst)]
1550         count += 1
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]:
```

```

)[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.
1692     Staff):
1693     leaf1 = abjad.select(staff).leaves()[7]
1694     abjad.attach(mark1, leaf1)

1695 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1696     Staff):
1697     leaf1 = abjad.select(staff).leaves()[7]
1698     abjad.attach(mark1, leaf1)

1699 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1700     Staff):
1701     leaf1 = abjad.select(staff).leaves()[7]
1702     abjad.attach(mark1, leaf1)

1703 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1704     Staff):
1705     leaf2 = abjad.select(staff).leaves()[16]
1706     abjad.attach(mark2, leaf2)

1707 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1708     Staff):
1709     leaf2 = abjad.select(staff).leaves()[16]
1710     abjad.attach(mark2, leaf2)

1711 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1712     Staff):
1713     leaf2 = abjad.select(staff).leaves()[16]
1714     abjad.attach(mark2, leaf2)

1715 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1716     Staff):
1717     leaf3 = abjad.select(staff).leaves()[22]
1718     abjad.attach(mark3, leaf3)

1719 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1720     Staff):
1721     leaf3 = abjad.select(staff).leaves()[22]
1722     abjad.attach(mark3, leaf3)

1723 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1724     Staff):
1725     leaf3 = abjad.select(staff).leaves()[22]
1726     abjad.attach(mark3, leaf3)

1727 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1728     Staff):
1729     leaf4 = abjad.select(staff).leaves()[29]

```

```

1729     abjad.attach(mark4, leaf4)
1730
1731 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1732     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.
1736     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.
1740     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.
1744     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.
1748     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.
1752     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.
1756     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.
1760     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.
1764     Staff):
1764     abjad.Instrument.transpose_from_sounding_pitch(staff)
1765
1766 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1767     Staff):
1767     abjad.Instrument.transpose_from_sounding_pitch(staff)
1768
1769 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1770     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/
1775 source/_stylesheets/abjad.ily'],
1776 )
1777
1778 abjad.SegmentMaker.comment_measure_numbers(score)
1779 ######
1780 directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/Segment_II
1781 '
1782 pdf_path = f'{directory}Segment_II.pdf'
1783 path = pathlib.Path('Segment_II.pdf')
1784 if path.exists():
1785     print(f'Removing {pdf_path} ...')
1786     path.unlink()
1787 time_1 = time.time()
1788 print(f'Persisting {pdf_path} ...')
1789 result = abjad.persist(score_file).as_pdf(pdf_path)
1790 print(result[0])
1791 print(result[1])
1792 print(result[2])
1793 success = result[3]
1794 if success is False:
1795     print('LilyPond failed!')
1796 time_2 = time.time()
1797 total_time = time_2 - time_1
1798 print(f'Total time: {total_time} seconds')
1799 if path.exists():
1800     print(f'Opening {pdf_path} ...')
1801     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
29     time_signatures])
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
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 reduceMod13(rw):
47     return [(x % 14) 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, 6, 8, 15, 23, 15, 8, 6, ]
58 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
59     flute_random_walk_one)]
60
61 seed(2)
62 clarinet_random_walk_one = []
63 clarinet_random_walk_one.append(-1 if random() < 0.5 else 1)
64 for i in range(1, 1000):
65     movement = -1 if random() < 0.5 else 1
66     value = clarinet_random_walk_one[i-1] + movement
67     clarinet_random_walk_one.append(value)
68 clarinet_random_walk_one = [abs(x) for x in clarinet_random_walk_one]
69 clarinet_chord_one = [-10, 0, 6, 8, 15, 23, 15, 8, 6, 0, ]
70 clarinet_notes_one = [clarinet_chord_one[x] for x in reduceMod9(
71     clarinet_random_walk_one)]
```

```

69
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
82 seed(4)
83 horn_random_walk_one = []
84 horn_random_walk_one.append(-1 if random() < 0.5 else 1)
85 for i in range(1, 1000):
86     movement = -1 if random() < 0.5 else 1
87     value = horn_random_walk_one[i-1] + movement
88     horn_random_walk_one.append(value)
89 horn_random_walk_one = [abs(x) for x in horn_random_walk_one]
90 horn_chord_one = [-19, -15, -10, 0, 6, 0, -10, -15, ]
91 horn_notes_one = [horn_chord_one[x] for x in reduceMod7(
92     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 = [0, 6, 8, 15, 8, 6, ]
103 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod5(
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)
113 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
114 trombone_chord_one = [-19, -15, -10, 0, -10, -15, ]
115 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod5(
116     trombone_random_walk_one)]
117
118 seed(7)
119 tuba_random_walk_one = []
120 tuba_random_walk_one.append(-1 if random() < 0.5 else 1)
121 for i in range(1, 1000):
122     movement = -1 if random() < 0.5 else 1

```

```

119     value = tuba_random_walk_one[i-1] + movement
120     tuba_random_walk_one.append(value)
121 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
122 tuba_chord_one = [-29, -20, -19, -15, -10, -15, -19, -20, ]
123 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod7(
124     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 = [0, 6, 8, 15, 23, 34, 37, 34, 23, 15, 8, 6, ]
135 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod11(
136     violin1_random_walk_one)]
137
138 seed(9)
139 violin2_random_walk_one = []
140 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
141 for i in range(1, 1000):
142     movement = -1 if random() < 0.5 else 1
143     value = violin2_random_walk_one[i-1] + movement
144     violin2_random_walk_one.append(value)
145 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
146 violin2_chord_one = [0, 6, 8, 15, 23, 15, 8, 6, ]
147 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod7(
148     violin2_random_walk_one)]
149
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 = [-10, 0, 6, 8, 15, 8, 6, 0, ]
159 viola_notes_one = [viola_chord_one[x] for x in reduceMod7(
160     viola_random_walk_one)]
161
162 seed(11)
163 cello_random_walk_one = []
164 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
165 for i in range(1, 1000):
166     movement = -1 if random() < 0.5 else 1
167     value = cello_random_walk_one[i-1] + movement
168     cello_random_walk_one.append(value)
169 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
170 cello_chord_one = [-20, -19, -15, -10, 0, 6, 8, 15, 8, 6, 0, -10, -15,
171     -19, ]
172 cello_notes_one = [cello_chord_one[x] for x in reduceMod13(

```

```

168     cello_random_walk_one)]
169
170 seed(12)
171 bass_random_walk_one = []
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 = [-29, -20, -19, -15, -10, 0, -10, -15, -19, -20, ]
178 bass_notes_one = [bass_chord_one[x] for x in reduceMod9(
179     bass_random_walk_one)]
180
181 flute_scale = [37, ]
182 clarinet_scale = [8, ]
183 bassoon_scale = [8, ]
184 horn_scale = [8, ]
185 trumpet_scale = [0, ]
186 trombone_scale = [-10, ]
187 tuba_scale = [-19, ]
188 violin1_scale = [30, 29.5, 29, 28.5, 28, 27.5, 27, 26.5, 26, 25.5, 25,
189     24.5, 24, 23.5, 23, 22.5, 22, 21.5, 21, 20.5, 20, 19.5, 19, 19.5,
190     20, 20.5, 21, 21.5, 22, 22.5, 23, 23.5, 24, 24.5, 25, 25.5, 26,
191     26.5, 27, 27.5, 28, 28.5, 29, 29.5, ]
192 violin2_scale = [19, 18.5, 18, 17.5, 17, 16.5, 16, 15.5, 15, 14.5, 14,
193     13.5, 13, 12.5, 12, 11.5, 11, 10.5, 10, 9.5, 9, 8.5, 8, 8.5, 9, 9.5,
194     10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5, 16,
195     16.5, 17, 17.5, 18, 18.5, ]
196 viola_scale = [8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5,
197     1, 0.5, 0, -0.5, -1, -1.5, -2, -2.5, -3, -2.5, -2, -1.5, -1, -0.5,
198     0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, ]
199 cello_scale = [-3, -3.5, -4, -4.5, -5, -5.5, -6, -6.5, -7, -7.5, -8,
200     -8.5, -9, -9.5, -10, -10.5, -11, -11.5, -12, -12.5, -13, -13.5, -14,
201     -13.5, -13, -12.5, -12, -11.5, -11, -10.5, -10, -9.5, -9, -8.5, -8,
202     -7.5, -7, -6.5, -6, -5.5, -5, -4.5, -4, -3.5, ]
203 bass_scale = [-14, -14.5, -15, -15.5, -16, -16.5, -17, -17.5, -18,
204     -18.5, -19, -19.5, -20, -20.5, -21, -21.5, -22, -22.5, -23, -23.5,
205     -24, -24.5, -25, -24.5, -24, -23.5, -23, -22.5, -22, -21.5, -21,
206     -20.5, -20, -19.5, -19, -18.5, -18, -17.5, -17, -16.5, -16, -15.5,
207     -15, -14.5, ]
208
209 seed(1)
210 flute_random_walk_two = []
211 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
212 for i in range(1, 1000):
213     movement = -1 if random() < 0.5 else 1
214     value = flute_random_walk_two[i-1] + movement
215     flute_random_walk_two.append(value)
216 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
217 flute_chord_two = [2, 8, 11, 22, 30, 22, 11, 8, ]
218 flute_notes_two = [flute_chord_two[x] for x in reduceMod7(
219     flute_random_walk_two)]
```

```

204 seed(2)
205 clarinet_random_walk_two = []
206 clarinet_random_walk_two.append(-1 if random() < 0.5 else 1)
207 for i in range(1, 1000):
208     movement = -1 if random() < 0.5 else 1
209     value = clarinet_random_walk_two[i-1] + movement
210     clarinet_random_walk_two.append(value)
211 clarinet_random_walk_two = [abs(x) for x in clarinet_random_walk_two]
212 clarinet_chord_two = [-8, -7, 2, 8, 11, 22, 11, 8, 2, -7, ]
213 clarinet_notes_two = [clarinet_chord_two[x] for x in reduceMod9(
214     clarinet_random_walk_two)]
215
216 seed(3)
217 bassoon_random_walk_two = []
218 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
219 for i in range(1, 1000):
220     movement = -1 if random() < 0.5 else 1
221     value = bassoon_random_walk_two[i-1] + movement
222     bassoon_random_walk_two.append(value)
223 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
224 bassoon_chord_two = [-17, -12, -8, -7, 2, -7, -8, -12, ]
225 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod7(
226     bassoon_random_walk_two)]
227
228 seed(4)
229 horn_random_walk_two = []
230 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
231 for i in range(1, 1000):
232     movement = -1 if random() < 0.5 else 1
233     value = horn_random_walk_two[i-1] + movement
234     horn_random_walk_two.append(value)
235 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
236 horn_chord_two = [-17, -12, -8, -7, 2, -7, -8, -12, ]
237 horn_notes_two = [horn_chord_two[x] for x in reduceMod7(
238     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 = [2, 8, 11, 8, ]
249 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod3(
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

```

```

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
260     seed(7)
261
262     tuba_random_walk_two = []
263     tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
264     for i in range(1, 1000):
265         movement = -1 if random() < 0.5 else 1
266         value = tuba_random_walk_two[i-1] + movement
267         tuba_random_walk_two.append(value)
268     tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
269     tuba_chord_two = [-27, -17, -12, -8, -7, -8, -12, -17, ]
270     tuba_notes_two = [tuba_chord_two[x] for x in reduceMod7(
271         tuba_random_walk_two)]
272
273     seed(8)
274
275     violin1_random_walk_two = []
276     violin1_random_walk_two.append(-1 if random() < 0.5 else 1)
277     for i in range(1, 1000):
278         movement = -1 if random() < 0.5 else 1
279         value = violin1_random_walk_two[i-1] + movement
280         violin1_random_walk_two.append(value)
281     violin1_random_walk_two = [abs(x) for x in violin1_random_walk_two]
282     violin1_chord_two = [2, 8, 11, 22, 30, 37, 39, 37, 30, 22, 11, 8, ]
283     violin1_notes_two = [violin1_chord_two[x] for x in reduceMod11(
284         violin1_random_walk_two)]
285
286     seed(9)
287
288     violin2_random_walk_two = []
289     violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
290     for i in range(1, 1000):
291         movement = -1 if random() < 0.5 else 1
292         value = violin2_random_walk_two[i-1] + movement
293         violin2_random_walk_two.append(value)
294     violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
295     violin2_chord_two = [2, 8, 11, 22, 30, 22, 11, 8, ]
296     violin2_notes_two = [violin2_chord_two[x] for x in reduceMod7(
297         violin2_random_walk_two)]
298
299     seed(10)
300
301     viola_random_walk_two = []
302     viola_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 = viola_random_walk_two[i-1] + movement
306         viola_random_walk_two.append(value)
307     viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
308     viola_chord_two = [-12, -8, -7, 2, 8, 11, 22, 11, 8, 2, -7, -8, ]
309     viola_notes_two = [viola_chord_two[x] for x in reduceMod11(
310         viola_random_walk_two)]
311
312

```

```

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):
307     movement = -1 if random() < 0.5 else 1
308     value = cello_random_walk_two[i-1] + movement
309     cello_random_walk_two.append(value)
310 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
311 cello_chord_two = [-17, -12, -8, -7, 2, 8, 2, -7, -8, -12,]
312 cello_notes_two = [cello_chord_two[x] for x in reduceMod9(
313     cello_random_walk_two)]
314
315 seed(12)
316 bass_random_walk_two = []
317 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
318 for i in range(1, 1000):
319     movement = -1 if random() < 0.5 else 1
320     value = bass_random_walk_two[i-1] + movement
321     bass_random_walk_two.append(value)
322 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
323 bass_chord_two = [-27, -17, -12, -8, -7, 2, -7, -8, -12, -17, ]
324 bass_notes_two = [bass_chord_two[x] for x in reduceMod9(
325     bass_random_walk_two)]
326
327 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
328     talea=abjadext.rmakers.Talea(
329         counts=[6, 2, 4, 2, 6, 1, 12, 8, 10, ],
330         denominator=16,
331         ),
332     beamSpecifier=abjadext.rmakers.BeamSpecifier(
333         beamDivisionsTogether=True,
334         beamRests=False,
335         ),
336     extraCountsPerDivision=[0, 1, 0, -1],
337     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
338         leftClasses=[abjad.Note, abjad.Rest],
339         leftCounts=[1, 0, 1],
340         ),
341     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
342         trivialize=True,
343         extractTrivial=True,
344         rewriteRestFilled=True,
345         ),
346     )
347
348 rmaker_two = abjadext.rmakers.NoteRhythmMaker()
349
350 rmaker_three = abjadext.rmakers.IncisedRhythmMaker(
351     inciseSpecifier=abjadext.rmakers.InciseSpecifier(
352         prefixTalea=[-1],
353         prefixCounts=[0, 1, 1, 0],
354         suffixTalea=[-1],
355         suffixCounts=[1, 0],
356         taleaDenominator=16,

```

```

355     ),
356 )
357
358 attachment_handler_one = AttachmentHandler(
359     starting_dynamic='mp',
360     ending_dynamic='f',
361     hairpin_indicator='<|',
362     articulation='',
363 )
364
365 attachment_handler_two = AttachmentHandler(
366     starting_dynamic='mp',
367     ending_dynamic='mf',
368     hairpin_indicator='--',
369     articulation='tenuto',
370 )
371
372 attachment_handler_three = AttachmentHandler(
373     starting_dynamic='mf',
374     ending_dynamic='f',
375     hairpin_indicator='--',
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,
```

```

$17     attachment_handler=attachment_handler_one,
$18 )
$19 trombonemusicmaker_two = MusicMaker(
$20     rmaker=rmaker_two,
$21     pitches=trombone_notes_two,
$22     continuous=True,
$23     attachment_handler=attachment_handler_two,
$24 )
$25 trombonemusicmaker_three = MusicMaker(
$26     rmaker=rmaker_three,
$27     pitches=trombone_notes_one,
$28     continuous=True,
$29     attachment_handler=attachment_handler_three,
$30 )
$31 #####cello#####
$32 celломusicmaker_one = MusicMaker(
$33     rmaker=rmaker_one,
$34     pitches=cello_scale,
$35     continuous=True,
$36     attachment_handler=attachment_handler_one,
$37 )
$38 celломusicmaker_two = MusicMaker(
$39     rmaker=rmaker_two,
$40     pitches=cello_notes_two,
$41     continuous=True,
$42     attachment_handler=attachment_handler_two,
$43 )
$44 celломusicmaker_three = MusicMaker(
$45     rmaker=rmaker_three,
$46     pitches=cello_notes_one,
$47     continuous=True,
$48     attachment_handler=attachment_handler_three,
$49 )
$50 #####horn#####
$51 hornmusicmaker_one = MusicMaker(
$52     rmaker=rmaker_one,
$53     pitches=horn_scale,
$54     continuous=True,
$55     attachment_handler=attachment_handler_one,
$56 )
$57 hornmusicmaker_two = MusicMaker(
$58     rmaker=rmaker_two,
$59     pitches=horn_notes_two,
$60     continuous=True,
$61     attachment_handler=attachment_handler_two,
$62 )
$63 hornmusicmaker_three = MusicMaker(
$64     rmaker=rmaker_three,
$65     pitches=horn_notes_one,
$66     continuous=True,
$67     attachment_handler=attachment_handler_three,
$68 )
$69 #####tuba#####
$70 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_timespan_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_timestspan_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_timespan_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_timestep in silences:
1329         timestep_list.append(
1330             abjad.AnnotatedTimespan(
1331                 start_offset=silence_timestep.start_offset,
1332                 stop_offset=silence_timestep.stop_offset,
1333                 annotation=MusicSpecifier(
1334                     music_maker=None,
1335                     voice_name=voice_name,
1336                 ),
1337             )
1338         )
1339     timestep_list.sort()
1340
1341 for voice_name, timestep_list in all_timestep_lists.items():
1342     shards = timestep_list.split_at_offsets(bounds)
1343     split_timestep_list = abjad.TimespanList()
1344     for shard in shards:
1345         split_timestep_list.extend(shard)
1346     split_timestep_list.sort()
1347     all_timestep_lists[voice_name] = timestep_list
1348
1349 score = abjad.Score([
1350     abjad.Staff(lilypond_type='TimeSignatureContext', name='Global
1351     Context 1'),
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
1364     Context 2'),
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
1379     Context 3'),
1380     abjad.StaffGroup(

```

```

1371 [
1372     abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1373     lilypond_type='Staff',),
1374         abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1375     lilypond_type='Staff',),
1376         abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1377     lilypond_type='Staff',),
1378         abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1379     lilypond_type='Staff',),
1380         abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1381     lilypond_type='Staff',),
1382     ],
1383     name='Staff Group 3',
1384 )
1385 ],
1386 )
1387
1388 for time_signature in time_signatures:
1389     skip = abjad.Skip(1, multiplier=(time_signature))
1390     abjad.attach(time_signature, skip)
1391     score['Global Context 1'].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 2'].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.
1422   Voice):
1423   for i , shard in enumerate(abjad.mutate(voice[:]).split(
1424     time_signatures)):
1425     time_signature = time_signatures[i]
1426     abjad.mutate(shard).rewrite_meter(time_signature)
1427
1428 for voice in abjad.iterate(score['Staff Group 2']).components(abjad.
1429   Voice):
1430   for i , shard in enumerate(abjad.mutate(voice[:]).split(
1431     time_signatures)):
1432     time_signature = time_signatures[i]
1433     abjad.mutate(shard).rewrite_meter(time_signature)
1434
1435 for voice in abjad.iterate(score['Staff Group 3']).components(abjad.
1436   Voice):
1437   for i , shard in enumerate(abjad.mutate(voice[:]).split(
1438     time_signatures)):
1439     time_signature = time_signatures[i]
1440     abjad.mutate(shard).rewrite_meter(time_signature)
1441
1442 print('Beaming runs ...')
1443
1444 for voice in abjad.select(score).components(abjad.Voice):
1445   for run in abjad.select(voice).runs():
1446     if 1 < len(run):
1447       specifier = abjadext.rmakers.BeamSpecifier(
1448         beam_each_division=False,
1449         )
1450       specifier(run)
1451       abjad.attach(abjad.StartBeam(), run[0])
1452       abjad.attach(abjad.StopBeam(), run[-1])
1453       for leaf in run:
1454         if abjad.Duration(1, 4) <= leaf.written_duration:
1455           continue
1456         previous_leaf = abjad.inspect(leaf).leaf(-1)
1457         next_leaf = abjad.inspect(leaf).leaf(1)
1458         if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
1459             abjad.Duration(1, 4) <= next_leaf.written_duration):
1460           left = previous_leaf.written_duration.flag_count
1461           right = leaf.written_duration.flag_count
1462           beam_count = abjad.BeamCount(
1463             left=left,
1464             right=right,
1465             )
1466           abjad.attach(beam_count, leaf)
1467           continue
1468         if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
1469             abjad.Duration(1, 4) <= previous_leaf.
1470             written_duration):
1471           left = leaf.written_duration.flag_count
1472           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.
1474     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 =
1479                 #1 \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.
1490     Staff):
1491         for selection in abjad.select(staff).components(abjad.Rest).
1492     group_by_contiguity():
1493             start_command = abjad.LilyPondLiteral(
1494                 r'\stopStaff \once \override Staff.StaffSymbol.line-count =
1495                 #1 \startStaff',
1496                 format_slot='before',
1497                 )
1498             stop_command = abjad.LilyPondLiteral(
1499                 r'\stopStaff \startStaff',
1500                 format_slot='after',
1501                 )
1502             abjad.attach(start_command, selection[0])
1503             abjad.attach(stop_command, selection[-1])
1504
1505     for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1506     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 =
1511                 #1 \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])

```

```

1511
1512 print('Stopping Hairpins ...')
1513 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
1514     Staff):
1515     for rest in abjad.iterate(staff).components(abjad.Rest):
1516         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.
1525     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.
1536     Staff):
1537     for rest in abjad.iterate(staff).components(abjad.Rest):
1538         previous_leaf = abjad.inspect(rest).leaf(-1)
1539         if isinstance(previous_leaf, abjad.Note):
1540             abjad.attach(abjad.StopHairpin(), rest)
1541         elif isinstance(previous_leaf, abjad.Chord):
1542             abjad.attach(abjad.StopHairpin(), rest)
1543         elif isinstance(previous_leaf, abjad.Rest):
1544             pass
1545
1546 print('Adding pitch material ...')
1547 def cyc(lst):
1548     count = 0
1549     while True:
1550         yield lst[count%len(lst)]
1551         count += 1
1552
1553 print('Adding attachments ...')
1554 bar_line = abjad.BarLine('||')
1555 metro = abjad.MetronomeMark((1, 4), 60)
1556 markup1 = abjad.Markup(r'\bold { M }')
1557 markup2 = abjad.Markup(r'\bold { N }')
1558 markup3 = abjad.Markup(r'\bold { O }')
1559 markup4 = abjad.Markup(r'\bold { P }')
1560 markup5 = abjad.Markup(r'\bold { Q }')
1561 markup6 = abjad.Markup(r'\bold { R }')
1562 mark1 = abjad.RehearsalMark(markup=markup1)
1563 mark2 = abjad.RehearsalMark(markup=markup2)
1564 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.Trombone(),
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
1650 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
    Staff):
1651     leaf1 = abjad.select(staff).leaves()[0]
1652     abjad.attach(next(instruments1), leaf1)
1653     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.
1714     Staff):
1715     leaf3 = abjad.select(staff).leaves() [22]
1716     abjad.attach(mark3, leaf3)
1717
1718 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1719     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.
1724     Staff):
1725     leaf3 = abjad.select(staff).leaves() [22]
1726     abjad.attach(mark3, leaf3)
1727
1728 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1729     Staff):
1730     leaf4 = abjad.select(staff).leaves() [29]
1731     abjad.attach(mark4, leaf4)
1732
1733 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1734     Staff):
1735     leaf4 = abjad.select(staff).leaves() [29]
1736     abjad.attach(mark4, leaf4)
1737
1738 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1739     Staff):
1740     leaf4 = abjad.select(staff).leaves() [29]
1741     abjad.attach(mark4, leaf4)
1742
1743 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1744     Staff):
1745     leaf5 = abjad.select(staff).leaves() [34]
1746     abjad.attach(mark5, leaf5)
1747
1748 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1749     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.
1754     Staff):
1755     leaf5 = abjad.select(staff).leaves() [34]
1756     abjad.attach(mark5, leaf5)
1757
1758 for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1759     Staff):
1760     leaf6 = abjad.select(staff).leaves() [39]
1761     abjad.attach(mark6, leaf6)
1762
1763 for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1764     Staff):
1765     leaf6 = abjad.select(staff).leaves() [39]
1766     abjad.attach(mark6, leaf6)
1767
1768 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1769     Staff):
1770     leaf6 = abjad.select(staff).leaves() [39]
1771     abjad.attach(mark6, leaf6)

```

```

1755     abjad.attach(mark6, leaf6)
1756
1757 for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1758     Staff):
1759     leaf6 = abjad.select(staff).leaves()[39]
1760     abjad.attach(mark6, leaf6)
1761
1762 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
1763     Staff):
1764     abjad.Instrument.transpose_from_sounding_pitch(staff)
1765
1766 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1767     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/
1773     source/_stylesheets/abjad.ily'],
1774     )
1775
1776 abjad.SegmentMaker.comment_measure_numbers(score)
1777 ######
1778
1779 directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/
1780 Segment_III'
1781 pdf_path = f'{directory}/Segment_III.pdf'
1782 path = pathlib.Path('Segment_III.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.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
29     time_signatures])
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
40 def reduceMod9(rw):
41     return [(x % 10) for x in rw]
42
43 def reduceMod13(rw):
44     return [(x % 14) for x in rw]
45
46 seed(1)
47 flute_random_walk_one = []
48 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
49 for i in range(1, 1000):
    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
67 seed(3)
68 bassoon_random_walk_one = []
69 bassoon_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 = bassoon_random_walk_one[i-1] + movement
73     bassoon_random_walk_one.append(value)
74 bassoon_random_walk_one = [abs(x) for x in bassoon_random_walk_one]
75 bassoon_chord_one = [-19, -8, -5, 2, 12, 2, -5, -8, ]
76 bassoon_notes_one = [bassoon_chord_one[x] for x in reduceMod7(
77     bassoon_random_walk_one)]
78
78 seed(4)
79 horn_random_walk_one = []
80 horn_random_walk_one.append(-1 if random() < 0.5 else 1)
81 for i in range(1, 1000):
82     movement = -1 if random() < 0.5 else 1
83     value = horn_random_walk_one[i-1] + movement
84     horn_random_walk_one.append(value)
85 horn_random_walk_one = [abs(x) for x in horn_random_walk_one]
86 horn_chord_one = [-19, -8, -5, 2, 12, 18, 12, 2, -5, -8, ]
87 horn_notes_one = [horn_chord_one[x] for x in reduceMod9(
88     horn_random_walk_one)]
89
89 seed(5)
90 trumpet_random_walk_one = []
91 trumpet_random_walk_one.append(-1 if random() < 0.5 else 1)
92 for i in range(1, 1000):
93     movement = -1 if random() < 0.5 else 1
94     value = trumpet_random_walk_one[i-1] + movement
95     trumpet_random_walk_one.append(value)
96 trumpet_random_walk_one = [abs(x) for x in trumpet_random_walk_one]
97 trumpet_chord_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]
98 trumpet_notes_one = [trumpet_chord_one[x] for x in reduceMod7(
99     trumpet_random_walk_one)]
```

```

99
100 seed(6)
101 trombone_random_walk_one = []
102 trombone_random_walk_one.append(-1 if random() < 0.5 else 1)
103 for i in range(1, 1000):
104     movement = -1 if random() < 0.5 else 1
105     value = trombone_random_walk_one[i-1] + movement
106     trombone_random_walk_one.append(value)
107 trombone_random_walk_one = [abs(x) for x in trombone_random_walk_one]
108 trombone_chord_one = [-19, -8, -5, 2, -5, -8, ]
109 trombone_notes_one = [trombone_chord_one[x] for x in reduceMod5(
110     trombone_random_walk_one)]
111
112 seed(7)
113 tuba_random_walk_one = []
114 tuba_random_walk_one.append(-1 if random() < 0.5 else 1)
115 for i in range(1, 1000):
116     movement = -1 if random() < 0.5 else 1
117     value = tuba_random_walk_one[i-1] + movement
118     tuba_random_walk_one.append(value)
119 tuba_random_walk_one = [abs(x) for x in tuba_random_walk_one]
120 tuba_chord_one = [-27, -19, -8, -5, 2, -5, -8, -19, ]
121 tuba_notes_one = [tuba_chord_one[x] for x in reduceMod7(
122     tuba_random_walk_one)]
123
124 seed(8)
125 violin1_random_walk_one = []
126 violin1_random_walk_one.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 = violin1_random_walk_one[i-1] + movement
130     violin1_random_walk_one.append(value)
131 violin1_random_walk_one = [abs(x) for x in violin1_random_walk_one]
132 violin1_chord_one = [-5, 2, 12, 18, 20, 25, 34, 35, 34, 25, 20, 18, 12,
133     2, ]
134 violin1_notes_one = [violin1_chord_one[x] for x in reduceMod13(
135     violin1_random_walk_one)]
136
137 seed(9)
138 violin2_random_walk_one = []
139 violin2_random_walk_one.append(-1 if random() < 0.5 else 1)
140 for i in range(1, 1000):
141     movement = -1 if random() < 0.5 else 1
142     value = violin2_random_walk_one[i-1] + movement
143     violin2_random_walk_one.append(value)
144 violin2_random_walk_one = [abs(x) for x in violin2_random_walk_one]
145 violin2_chord_one = [-5, 2, 12, 18, 20, 18, 12, 2, ]
146 violin2_notes_one = [violin2_chord_one[x] for x in reduceMod7(
147     violin2_random_walk_one)]

```

```

148     movement = -1 if random() < 0.5 else 1
149     value = viola_random_walk_one[i-1] + movement
150     viola_random_walk_one.append(value)
151 viola_random_walk_one = [abs(x) for x in viola_random_walk_one]
152 viola_chord_one = [-8, -5, 2, 12, 18, 12, 2, -5, ]
153 viola_notes_one = [viola_chord_one[x] for x in reduceMod7(
    viola_random_walk_one)]
154
155 seed(11)
156 cello_random_walk_one = []
157 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
158 for i in range(1, 1000):
159     movement = -1 if random() < 0.5 else 1
160     value = cello_random_walk_one[i-1] + movement
161     cello_random_walk_one.append(value)
162 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
163 cello_chord_one = [-19, -8, -5, 2, 12, 2, -5, -8]
164 cello_notes_one = [cello_chord_one[x] for x in reduceMod7(
    cello_random_walk_one)]
165
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, 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, ]
185 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, ]
186 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, ]
187 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, ]

```

```

188 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, ]
189
190 seed(1)
191 flute_random_walk_two = []
192 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
193 for i in range(1, 1000):
194     movement = -1 if random() < 0.5 else 1
195     value = flute_random_walk_two[i-1] + movement
196     flute_random_walk_two.append(value)
197 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
198 flute_chord_two = [4, 12, 18, 22, 23, 22, 18, 12, ]
199 flute_notes_two = [flute_chord_two[x] for x in reduceMod7(
    flute_random_walk_two)]
200
201 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(
    clarinet_random_walk_two)]
211
212 seed(3)
213 bassoon_random_walk_two = []
214 bassoon_random_walk_two.append(-1 if random() < 0.5 else 1)
215 for i in range(1, 1000):
216     movement = -1 if random() < 0.5 else 1
217     value = bassoon_random_walk_two[i-1] + movement
218     bassoon_random_walk_two.append(value)
219 bassoon_random_walk_two = [abs(x) for x in bassoon_random_walk_two]
220 bassoon_chord_two = [-17, -10, -7, 4, 12, 4, -7, -10, ]
221 bassoon_notes_two = [bassoon_chord_two[x] for x in reduceMod7(
    bassoon_random_walk_two)]
222
223 seed(4)
224 horn_random_walk_two = []
225 horn_random_walk_two.append(-1 if random() < 0.5 else 1)
226 for i in range(1, 1000):
227     movement = -1 if random() < 0.5 else 1
228     value = horn_random_walk_two[i-1] + movement
229     horn_random_walk_two.append(value)
230 horn_random_walk_two = [abs(x) for x in horn_random_walk_two]
231 horn_chord_two = [-17, -10, -7, 4, -7, -10, ]
232 horn_notes_two = [horn_chord_two[x] for x in reduceMod5(
    horn_random_walk_two)]
233

```

```

234 seed(5)
235 trumpet_random_walk_two = []
236 trumpet_random_walk_two.append(-1 if random() < 0.5 else 1)
237 for i in range(1, 1000):
238     movement = -1 if random() < 0.5 else 1
239     value = trumpet_random_walk_two[i-1] + movement
240     trumpet_random_walk_two.append(value)
241 trumpet_random_walk_two = [abs(x) for x in trumpet_random_walk_two]
242 trumpet_chord_two = [4, 12, 18, 12, ]
243 trumpet_notes_two = [trumpet_chord_two[x] for x in reduceMod3(
244     trumpet_random_walk_two)]
245
246 seed(6)
247 trombone_random_walk_two = []
248 trombone_random_walk_two.append(-1 if random() < 0.5 else 1)
249 for i in range(1, 1000):
250     movement = -1 if random() < 0.5 else 1
251     value = trombone_random_walk_two[i-1] + movement
252     trombone_random_walk_two.append(value)
253 trombone_random_walk_two = [abs(x) for x in trombone_random_walk_two]
254 trombone_chord_two = [-17, -10, -7, 4, -7, -10, ]
255 trombone_notes_two = [trombone_chord_two[x] for x in reduceMod5(
256     trombone_random_walk_two)]
257
258 seed(7)
259 tuba_random_walk_two = []
260 tuba_random_walk_two.append(-1 if random() < 0.5 else 1)
261 for i in range(1, 1000):
262     movement = -1 if random() < 0.5 else 1
263     value = tuba_random_walk_two[i-1] + movement
264     tuba_random_walk_two.append(value)
265 tuba_random_walk_two = [abs(x) for x in tuba_random_walk_two]
266 tuba_chord_two = [-27, -17, -10, -7, 4, -7, -10, -17, ]
267 tuba_notes_two = [tuba_chord_two[x] for x in reduceMod7(
268     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 = [4, 12, 18, 22, 23, 32, 37, 39, 37, 32, 23, 22, 18,
279     12, ]
280 violin1_notes_two = [violin1_chord_two[x] for x in reduceMod13(
281     violin1_random_walk_two)]
282
283 seed(9)
284 violin2_random_walk_two = []
285 violin2_random_walk_two.append(-1 if random() < 0.5 else 1)
286 for i in range(1, 1000):
287     movement = -1 if random() < 0.5 else 1

```

```

283     value = violin2_random_walk_two[i-1] + movement
284     violin2_random_walk_two.append(value)
285 violin2_random_walk_two = [abs(x) for x in violin2_random_walk_two]
286 violin2_chord_two = [4, 12, 18, 22, 23, 32, 23, 22, 18, 12, ]
287 violin2_notes_two = [violin2_chord_two[x] for x in reduceMod9(
288     violin2_random_walk_two)]
289
290 seed(10)
291 viola_random_walk_two = []
292 viola_random_walk_two.append(-1 if random() < 0.5 else 1)
293 for i in range(1, 1000):
294     movement = -1 if random() < 0.5 else 1
295     value = viola_random_walk_two[i-1] + movement
296     viola_random_walk_two.append(value)
297 viola_random_walk_two = [abs(x) for x in viola_random_walk_two]
298 viola_chord_two = [-10, -7, 4, 12, 18, 12, 4, -7, ]
299 viola_notes_two = [viola_chord_two[x] for x in reduceMod7(
300     viola_random_walk_two)]
301
302 seed(11)
303 cello_random_walk_two = []
304 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
305 for i in range(1, 1000):
306     movement = -1 if random() < 0.5 else 1
307     value = cello_random_walk_two[i-1] + movement
308     cello_random_walk_two.append(value)
309 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
310 cello_chord_two = [-17, -10, -7, 4, 12, 4, -7, -10, ]
311 cello_notes_two = [cello_chord_two[x] for x in reduceMod7(
312     cello_random_walk_two)]
313
314 seed(12)
315 bass_random_walk_two = []
316 bass_random_walk_two.append(-1 if random() < 0.5 else 1)
317 for i in range(1, 1000):
318     movement = -1 if random() < 0.5 else 1
319     value = bass_random_walk_two[i-1] + movement
320     bass_random_walk_two.append(value)
321 bass_random_walk_two = [abs(x) for x in bass_random_walk_two]
322 bass_chord_two = [-27, -17, -10, -7, 4, -7, -10, -17, ]
323 bass_notes_two = [bass_chord_two[x] for x in reduceMod7(
324     bass_random_walk_two)]
325
326 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
327     talea=abjadext.rmakers.Talea(
328         counts=[2, 3, 2, 1, 4, 3, 1, 4, 5, 1],
329         denominator=8,
330         ),
331     beamSpecifier=abjadext.rmakers.BeamSpecifier(
332         beamDivisionsTogether=True,
333         beamRests=False,
334         ),
335     extraCountsPerDivision=[1, 1, 0, -1, 0],
336     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
337         )
338 )
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```

333     left_classes=[abjad.Note, abjad.Rest],
334     left_counts=[0, 1, 1],
335     ),
336     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
337         trivialize=True,
338         extract_trivial=True,
339         rewrite_rest_filled=True,
340         ),
341     )
342
343 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
344     talea=abjadext.rmakers.Talea(
345         counts=[2, 1, 3, 1, 4, 1, 1, 5],
346         denominator=16,
347         ),
348     beamSpecifier=abjadext.rmakers.BeamSpecifier(
349         beam_divisions_together=True,
350         beam_rests=False,
351         ),
352     extra_counts_per_division=[0, 1, 0, -1],
353     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
354         left_classes=[abjad.Note, abjad.Rest],
355         left_counts=[1, 1, 0, 0],
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 attachment_handler_one = AttachmentHandler(
388     starting_dynamic='mf',
389     ending_dynamic='ff',
390     hairpin_indicator='<',
391     articulation='',
392 )
393
394 attachment_handler_two = AttachmentHandler(
395     starting_dynamic='ff',
396     ending_dynamic='mf',
397     hairpin_indicator='>',
398     articulation='',
399 )
400
401 attachment_handler_three = AttachmentHandler(
402     starting_dynamic='p',
403     ending_dynamic='pp',
404     hairpin_indicator='--',
405     articulation='tenuto',
406 )
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(
518     rmaker=rmaker_three,
519     pitches=viola_notes_one,
520     continuous=True,
521     attachment_handler=attachment_handler_three,
522 )
523 #####bassoon#####
524 bassoonmusicmaker_one = MusicMaker(
525     rmaker=rmaker_one,
526     pitches=bassoon_scale,
527     continuous=True,
528     attachment_handler=attachment_handler_one,
529 )
530 bassoonmusicmaker_two = MusicMaker(
531     rmaker=rmaker_two,
532     pitches=bassoon_notes_two,
533     continuous=True,
534     attachment_handler=attachment_handler_two,
535 )
536 bassoonmusicmaker_three = MusicMaker(
537     rmaker=rmaker_three,
538     pitches=bassoon_notes_one,
539     continuous=True,
540     attachment_handler=attachment_handler_three,
541 )
542 #####trombone#####
543 trombonemusicmaker_one = MusicMaker(
544     rmaker=rmaker_one,
545     pitches=trombone_scale,
546     continuous=True,
547     attachment_handler=attachment_handler_one,
548 )
```

```
549 trombonemusicmaker_two = MusicMaker(
550     rmaker=rmaker_two,
551     pitches=trombone_notes_two,
552     continuous=True,
553     attachment_handler=attachment_handler_two,
554 )
555 trombonemusicmaker_three = MusicMaker(
556     rmaker=rmaker_three,
557     pitches=trombone_notes_one,
558     continuous=True,
559     attachment_handler=attachment_handler_three,
560 )
561 #####cello#####
562 celломusicmaker_one = MusicMaker(
563     rmaker=rmaker_one,
564     pitches=cello_scale,
565     continuous=True,
566     attachment_handler=attachment_handler_one,
567 )
568 celломusicmaker_two = MusicMaker(
569     rmaker=rmaker_two,
570     pitches=cello_notes_two,
571     continuous=True,
572     attachment_handler=attachment_handler_two,
573 )
574 celломusicmaker_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_timestspan_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
1383     Context 1'),
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
1396     Context 2'),
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
1411     Context 3'),
1412     abjad.StaffGroup(

```

```
1403
1404     [
1405         abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
1406         lilypond_type='Staff'),
1407         abjad.Staff([abjad.Voice(name='Voice 9')], name='Staff 9',
1408         lilypond_type='Staff'),
1409         abjad.Staff([abjad.Voice(name='Voice 10')], name='Staff 10',
1410         lilypond_type='Staff'),
1411         abjad.Staff([abjad.Voice(name='Voice 11')], name='Staff 11',
1412         lilypond_type='Staff'),
1413         abjad.Staff([abjad.Voice(name='Voice 12')], name='Staff 12',
1414         lilypond_type='Staff'),
1415     ],
1416     name='Staff Group 3',
1417 )
1418
1419 ],
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
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))
1496                     and
1497                         abjad.Duration(1, 4) <= previous_leaf.
1498                         written_duration):
1499                     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.
1507         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 =
1512 #1 \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     for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1523         Staff):
1524         for selection in abjad.select(staff).components(abjad.Rest).
1525             group_by_contiguity():
1526             start_command = abjad.LilyPondLiteral(
1527                 r'\stopStaff \once \override Staff.StaffSymbol.line-count =
1528 #1 \startStaff',
1529                 format_slot='before',
1530                 )
1531             stop_command = abjad.LilyPondLiteral(
1532                 r'\stopStaff \startStaff',
1533                 format_slot='after',
1534                 )
1535             abjad.attach(start_command, selection[0])
1536             abjad.attach(stop_command, selection[-1])
1537
1538     for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1539         Staff):
1540         for selection in abjad.select(staff).components(abjad.Rest).
1541             group_by_contiguity():
1542             start_command = abjad.LilyPondLiteral(
1543                 r'\stopStaff \once \override Staff.StaffSymbol.line-count =
1544 #1 \startStaff',
1545                 format_slot='before',
1546                 )
1547             stop_command = abjad.LilyPondLiteral(
1548                 r'\stopStaff \startStaff',
1549                 format_slot='after',
1550                 )
1551             abjad.attach(start_command, selection[0])

```

```

1543     abjad.attach(stop_command, selection[-1])
1544
1545 print('Stopping Hairpins ...')
1546 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
    Staff):
1547     for rest in abjad.iterate(staff).components(abjad.Rest):
1548         previous_leaf = abjad.inspect(rest).leaf(-1)
1549         if isinstance(previous_leaf, abjad.Note):
1550             abjad.attach(abjad.StopHairpin(), rest)
1551         elif isinstance(previous_leaf, abjad.Chord):
1552             abjad.attach(abjad.StopHairpin(), rest)
1553         elif isinstance(previous_leaf, abjad.Rest):
1554             pass
1555
1556 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
    Staff):
1557     for rest in abjad.iterate(staff).components(abjad.Rest):
1558         previous_leaf = abjad.inspect(rest).leaf(-1)
1559         if isinstance(previous_leaf, abjad.Note):
1560             abjad.attach(abjad.StopHairpin(), rest)
1561         elif isinstance(previous_leaf, abjad.Chord):
1562             abjad.attach(abjad.StopHairpin(), rest)
1563         elif isinstance(previous_leaf, abjad.Rest):
1564             pass
1565
1566 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
    Staff):
1567     for rest in abjad.iterate(staff).components(abjad.Rest):
1568         previous_leaf = abjad.inspect(rest).leaf(-1)
1569         if isinstance(previous_leaf, abjad.Note):
1570             abjad.attach(abjad.StopHairpin(), rest)
1571         elif isinstance(previous_leaf, abjad.Chord):
1572             abjad.attach(abjad.StopHairpin(), rest)
1573         elif isinstance(previous_leaf, abjad.Rest):
1574             pass
1575
1576 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
1706                               )[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
1712                               )[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
1718                               )[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.
1724                               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.
1728                               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.
1732                               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.
1736                               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.
1740                               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.
1744                               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.
1748         Staff):
1749         leaf3 = abjad.select(staff).leaves()[22]
1750         abjad.attach(mark3, leaf3)
1751
1752     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1753         Staff):
1754         leaf3 = abjad.select(staff).leaves()[22]
1755         abjad.attach(mark3, leaf3)
1756
1757     for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1758         Staff):
1759         leaf3 = abjad.select(staff).leaves()[22]
1760         abjad.attach(mark3, leaf3)
1761
1762     for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1763         Staff):
1764         leaf4 = abjad.select(staff).leaves()[29]
1765         abjad.attach(mark4, leaf4)
1766
1767     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1768         Staff):
1769         leaf4 = abjad.select(staff).leaves()[29]
1770         abjad.attach(mark4, leaf4)
1771
1772     for staff in abjad.iterate(score['Global Context 3']).components(abjad.
1773         Staff):
1774         leaf4 = abjad.select(staff).leaves()[29]
1775         abjad.attach(mark4, leaf4)
1776
1777     for staff in abjad.iterate(score['Global Context 1']).components(abjad.
1778         Staff):
1779         leaf5 = abjad.select(staff).leaves()[34]
1780         abjad.attach(mark5, leaf5)
1781
1782     for staff in abjad.iterate(score['Global Context 2']).components(abjad.
1783         Staff):
1784         leaf5 = abjad.select(staff).leaves()[34]
1785         abjad.attach(mark5, leaf5)
1786
1787     for staff in abjad.iterate(score['Global Context 3']).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.
1792     Staff):
1793     leaf6 = abjad.select(staff).leaves()[39]
1794     abjad.attach(mark6, leaf6)
1795
1796 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
1797     Staff):
1798     abjad.Instrument.transpose_from_sounding_pitch(staff)
1799
1800 for staff in abjad.iterate(score['Staff Group 2']).components(abjad.
1801     Staff):
1802     abjad.Instrument.transpose_from_sounding_pitch(staff)
1803
1804 for staff in abjad.iterate(score['Staff Group 3']).components(abjad.
1805     Staff):
1806     abjad.Instrument.transpose_from_sounding_pitch(staff)
1807
1808 score_file = abjad.LilyPondFile.new(
1809     score,
1810     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/
1811     source/_stylesheets/abjad.ily'],
1812     )
1813
1814 abjad.SegmentMaker.comment_measure_numbers(score)
1815 ######
1816
1817 directory = '/Users/evansdsg2/Scores/tianshu/tianshu/Segments/Segment_IV'
1818
1819 pdf_path = f'{directory}/Segment_IV.pdf'
1820 path = pathlib.Path('Segment_IV.pdf')
1821 if path.exists():
1822     print(f'Removing {pdf_path} ...')
1823     path.unlink()
1824 time_1 = time.time()
1825 print(f'Persisting {pdf_path} ...')
1826 result = abjad.persist(score_file).as_pdf(pdf_path)
1827 print(result[0])
1828 print(result[1])
1829 print(result[2])
1830 success = result[3]
1831 if success is False:
1832     print('LilyPond failed!')
1833 time_2 = time.time()
1834 total_time = time_2 - time_1
1835 print(f'Total time: {total_time} seconds')
1836 if path.exists():
1837     print(f'Opening {pdf_path} ...')
1838     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 \ekmeliStyle 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/
21     tianshu_title.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 }
```

```

31         }
32
33     arranger = \markup \override #'(
34
35         font-name . "Didot"
36
37     ) \fontsize #2.3 {
38
39         "Gregory Rowland Evans"
40
41     }
42
43 }
44
45 \layout {
46
47     \accidentalStyle forget
48
49     \%accidentalStyle modern
50
51     \%accidentalStyle modern-cautionary
52
53     \%accidentalStyle neo-modern
54
55     \%accidentalStyle dodecaphonic
56
57     indent = #5
58
59     %ragged-last = ##t
60
61     %ragged-right = ##t
62
63     %left-margin = #15
64
65 \context {
66
67     \name TimeSignatureContext
68
69     \type Engraver_group
70
71     \numericTimeSignature
72
73     \consists Axis_group_engraver
74
75     \consists Bar_number_engraver
76
77     \consists Time_signature_engraver
78
79     \consists Mark_engraver
80
81     \consists Metronome_mark_engraver
82
83     \override BarNumber.Y-extent = #'(0 . 0)
84
85     \override BarNumber.Y-offset = 0
86
87     \override BarNumber.extra-offset = #'(-4 . 0)
88
89     \%override BarNumber.font-name = "Didot"
90
91     \override BarNumber.stencil = #(
92
93         make-stencil-boxer 0.1 0.7 ly:text-interface::print

```

```

64
65     )
66
67     \override BarNumber.font-size = 1
68
69     \override BarNumber.padding = 4
70
71     \override MetronomeMark.X-extent = #'(0 . 0)
72
73     \override MetronomeMark.Y-extent = #'(0 . 0)
74
75     \override MetronomeMark.break-align-symbols = #'(left-edge)
76
77     \override MetronomeMark.extra-offset = #'(0 . 4)
78
79     \override MetronomeMark.font-size = 3
80
81     \override RehearsalMark.stencil = #(
82
83         make-stencil-circler 0.1 0.7 ly:text-interface::print
84
85     )
86
87     \override RehearsalMark.X-extent = #'(0 . 0)
88
89     \override RehearsalMark.X-offset = 6
90
91     \override RehearsalMark.Y-offset = -2.25
92
93     \override RehearsalMark.break-align-symbols = #'(time-signature)
94
95     \override RehearsalMark.break-visibility = #end-of-line-invisible
96
97     \override RehearsalMark.font-name = "Didot"
98
99     \override RehearsalMark.font-size = 8
100
101    \override RehearsalMark.outside-staff-priority = 500
102
103    \override RehearsalMark.self-alignment-X = #center
104
105        \override TimeSignature.X-extent = #'(0 . 0)
106
107        \override TimeSignature.X-offset = #ly:self-alignment-interface
108        ::x-aligned-on-self
109
110            \override TimeSignature.Y-extent = #'(0 . 0)
111
112            \override TimeSignature.break-align-symbol = ##f
113
114            \override TimeSignature.break-visibility = #end-of-line-
115            invisible
116
117            \override TimeSignature.font-size = #6
118
119            \override TimeSignature.self-alignment-X = #center
120
121    \override TimeSignature.whiteout = ##t
122
123        \override VerticalAxisGroup.default-staff-staff-spacing = #'(
124            (basic-distance . 0) (minimum-distance . 10) (padding . 6) (
125                stretchability . 0)

```

```

94     )
95
96 \context {
97     \Score
98     \remove Bar_number_engraver
99     \remove Mark_engraver
100    \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-rods
115    \override TupletNumber.text = #tuplet-number::calc-fraction-text
116    proportionalNotationDuration = #(ly:make-moment 1 42)
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
129     \context {
130         \RhythmicStaff
131         \remove Time_signature_engraver
132     }
133     \context {
134         \StaffGroup
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
23     time_signatures])
24
25 def reduceMod3(rw):
26     return [(x % 4) for x in rw]
27
28 def reduceMod7(rw):
29     return [(x % 8) for x in rw]
30
31 def reduceMod9(rw):
32     return [(x % 10) for x in rw]
33
34 def cyc(lst):
35     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
47     i in lst2]
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 = [1, 2, 9, 10, 18, 27, 18, 10, 9, 2, ]
58 flute_notes_one = [flute_chord_one[x] for x in reduceMod9(
59     flute_random_walk_one)]
60
61 seed(4)
62 saxophone_random_walk_one = []
63 saxophone_random_walk_one.append(-1 if random() < 0.5 else 1)
64 for i in range(1, 1000):
65     movement = -1 if random() < 0.5 else 1
66     value = saxophone_random_walk_one[i-1] + movement
67     saxophone_random_walk_one.append(value)
68 saxophone_random_walk_one = [abs(x) for x in saxophone_random_walk_one]
69 saxophone_chord_one = [-8, 1, 2, 9, 10, 9, 2, 1, ]
70 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod7(
71     saxophone_random_walk_one)]
72
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 = [-24, -17, -16, -8, 1, 2, 1, -8, -16, -17, ]
82 cello_notes_one = [cello_chord_one[x] for x in reduceMod9(
83     cello_random_walk_one)]
84
85 seed(1)
86 flute_random_walk_two = []
87 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
88 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(
91     flute_random_walk_two)]
92
93 seed(4)
94 saxophone_random_walk_two = []
95 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
96 for i in range(1, 1000):
97     movement = -1 if random() < 0.5 else 1
98     value = saxophone_random_walk_two[i-1] + movement
99     saxophone_random_walk_two.append(value)
100 saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
101 saxophone_chord_two = [-8, 2, 10, 2, ]
102 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod3(
103     saxophone_random_walk_two)]
104
105 seed(8)
106 cello_random_walk_two = []
107 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
108 for i in range(1, 1000):
109     movement = -1 if random() < 0.5 else 1
110     value = cello_random_walk_two[i-1] + movement
111     cello_random_walk_two.append(value)
112 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
113 cello_chord_two = [-24, -16, 1, -16, ]
114 cello_notes_two = [cello_chord_two[x] for x in reduceMod3(
115     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 = [2, 10, 27, 10, ]
126 flute_notes_three = [flute_chord_three[x] for x in reduceMod3(
127     flute_random_walk_three)]
128
129 seed(4)
130 saxophone_random_walk_three = []
131 saxophone_random_walk_three.append(-1 if random() < 0.5 else 1)
132 for i in range(1, 1000):
133     movement = -1 if random() < 0.5 else 1
134     value = saxophone_random_walk_three[i-1] + movement
135     saxophone_random_walk_three.append(value)
136 saxophone_random_walk_three = [abs(x) for x in
137     saxophone_random_walk_three]
138 saxophone_chord_three = [1, 2, 10, 2, ]

```

```

134     saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod3(
135         saxophone_random_walk_three)]
136
136 seed(8)
137 cello_random_walk_three = []
138 cello_random_walk_three.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 = 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
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     rewrite_rest_filled=True,
187     ),
188 )
189
190 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
191     denominators=[16, 16, 8, 16, 16, 8],
192     extra_counts_per_division=[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 attachment_handler_one = AttachmentHandler(
209     starting_dynamic='mf',
210     ending_dynamic='p',
211     hairpin_indicator='--',
212     articulation='accent',
213     )
214
215 attachment_handler_two = AttachmentHandler(
216     starting_dynamic='mf',
217     ending_dynamic='ff',
218     hairpin_indicator='<',
219     articulation='tenuto',
220     )
221
222 attachment_handler_three = AttachmentHandler(
223     starting_dynamic='ff',
224     ending_dynamic='mp',
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 celломusicmaker_one = MusicMaker(
269     rmaker=rmaker_one,
270     pitches=cello_notes_one,
271     continuous=True,
272     attachment_handler=attachment_handler_one,
273 )
274 celломusicmaker_two = MusicMaker(
275     rmaker=rmaker_two,
276     pitches=cello_notes_two,
277     continuous=True,
278     attachment_handler=attachment_handler_two,
279 )
280 celломusicmaker_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
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), cellomusicmaker_one],
366     [(18, 8), (22, 8), cellomusicmaker_two],
367     [(25, 8), (29, 8), cellomusicmaker_three],
368     [(29, 8), (32, 8), cellomusicmaker_one],
369     [(35, 8), (39, 8), cellomusicmaker_two],
370     [(39, 8), (42, 8), cellomusicmaker_three],
371     [(45, 8), (50, 8), cellomusicmaker_one],
372     [(50, 8), (52, 8), cellomusicmaker_two],
373     [(55, 8), (56, 8), cellomusicmaker_three],
374     [(56, 8), (61, 8), cellomusicmaker_one],
375     [(61, 8), (62, 8), cellomusicmaker_two],
376     [(65, 8), (69, 8), cellomusicmaker_three],
377     [(69, 8), (72, 8), cellomusicmaker_one],
378     [(75, 8), (79, 8), cellomusicmaker_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
422     Context'),
423     abjad.StaffGroup(
424         [
425             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
426             lilypond_type='Staff'),
427             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
428             lilypond_type='Staff'),
429             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
430             lilypond_type='Staff'),
431         ],
432         name='Staff Group 1',
433     ),
434 ],
435 )
436
437 for time_signature in time_signatures:
438     skip = abjad.Skip(1, multiplier=(time_signature))
439     abjad.attach(time_signature, skip)
440     score['Global Context'].append(skip)
441
442 print('Making containers ...')
443
444 def make_container(music_maker, durations):
445     selections = music_maker(durations)
446     container = abjad.Container([])
447     container.extend(selections)
448     return container
449
450 def key_function(timespan):
451     return timespan.annotation.music_maker or silence_maker
452
453 for voice_name, timespan_list in all_timespan_lists.items():
454     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(
464         time_signatures)):
465         time_signature = time_signatures[i]
466         abjad.mutate(shard).rewrite_meter(time_signature)
467
468 print('Beaming runs ...')
469
470 for voice in abjad.select(score).components(abjad.Voice):
471     for run in abjad.select(voice).runs():
472         if 1 < len(run):
473             specifier = abjadext.rmakers.BeamSpecifier(
474                 beam_each_division=False,
475                 )
476             specifier(run)
477             abjad.attach(abjad.StartBeam(), run[0])
478             abjad.attach(abjad.StopBeam(), run[-1])
479             for leaf in run:
480                 if abjad.Duration(1, 4) <= leaf.written_duration:
481                     continue
482                 previous_leaf = abjad.inspect(leaf).leaf(-1)
483                 next_leaf = abjad.inspect(leaf).leaf(1)
484                 if (isinstance(next_leaf, (abjad.Chord, abjad.Note)) and
485                     abjad.Duration(1, 4) <= next_leaf.written_duration):
486                     left = previous_leaf.written_duration.flag_count
487                     right = leaf.written_duration.flag_count
488                     beam_count = abjad.BeamCount(
489                         left=left,
490                         right=right,
491                         )
492                     abjad.attach(beam_count, leaf)
493                     continue
494                 if (isinstance(previous_leaf, (abjad.Chord, abjad.Note)) and
495                     abjad.Duration(1, 4) <= previous_leaf.written_duration):
496                     left = leaf.written_duration.flag_count
497                     right = next_leaf.written_duration.flag_count
498                     beam_count = abjad.BeamCount(
499                         left=left,
500                         right=right,
501                         )
502                     abjad.attach(beam_count, leaf)

```

```

$02
$03 print('Stopping Hairpins ...')
$04 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
    Staff):
$05     for rest in abjad.iterate(staff).components(abjad.Rest):
$06         previous_leaf = abjad.inspect(rest).leaf(-1)
$07         if isinstance(previous_leaf, abjad.Note):
$08             abjad.attach(abjad.StopHairpin(), rest)
$09         elif isinstance(previous_leaf, abjad.Chord):
$10             abjad.attach(abjad.StopHairpin(), rest)
$11         elif isinstance(previous_leaf, abjad.Rest):
$12             pass
$13
$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):
$68     abjad.Instrument.transpose_from_sounding_pitch(staff)
$69
$70 score_file = abjad.LilyPondFile.new(
$71     score,
$72     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/
$73         source/_stylesheets/abjad.ily'],
$74     )
$75
$76 abjad.SegmentMaker.comment_measure_numbers(score)
$77 ######
$78 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand
$79     /Segments/Segment_I'
$80 pdf_path = f'{directory}/Segment_I.pdf'
$81 path = pathlib.Path('Segment_I.pdf')
$82 if path.exists():
$83     print(f'Removing {pdf_path} ...')
$84     path.unlink()
$85 time_1 = time.time()
$86 print(f'Persisting {pdf_path} ...')
$87 result = abjad.persist(score_file).as_pdf(pdf_path)
$88 print(result[0])
$89 print(result[1])
$90 print(result[2])
$91 success = result[3]
$92 if success is False:
$93     print('LilyPond failed!')
$94 time_2 = time.time()
$95 total_time = time_2 - time_1
$96 print(f'Total time: {total_time} seconds')
$97 if path.exists():
$98     print(f'Opening {pdf_path} ...')
$99     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
23     time_signatures])
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
45     i in 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):
```

```

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 = [0, 11, 14, 17, 18, 22, 18, 17, 14, 11, ]
54 flute_notes_one = [flute_chord_one[x] for x in reduceMod9(
55     flute_random_walk_one)]
56
56 seed(4)
57 saxophone_random_walk_one = []
58 saxophone_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 = saxophone_random_walk_one[i-1] + movement
62     saxophone_random_walk_one.append(value)
63 saxophone_random_walk_one = [abs(x) for x in saxophone_random_walk_one]
64 saxophone_chord_one = [-8, -5, -4, 0, 11, 14, 11, 0, -4, -5, ]
65 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod9(
66     saxophone_random_walk_one)]
67
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
78 seed(1)
79 flute_random_walk_two = []
80 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
81 for i in range(1, 1000):
82     movement = -1 if random() < 0.5 else 1
83     value = flute_random_walk_two[i-1] + movement
84     flute_random_walk_two.append(value)
85 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
86 flute_chord_two = [0, 14, 18, 14, ]
87 flute_notes_two = [flute_chord_two[x] for x in reduceMod3(
88     flute_random_walk_two)]
89
89 seed(4)
90 saxophone_random_walk_two = []
91 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
92 for i in range(1, 1000):
93     movement = -1 if random() < 0.5 else 1
94     value = saxophone_random_walk_two[i-1] + movement
95     saxophone_random_walk_two.append(value)
96 saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
97 saxophone_chord_two = [-8, -4, 11, -4, ]
98 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod3(

```

```

    saxophone_random_walk_two)]
```

99

100 seed(8)

101 cello_random_walk_two = []

102 cello_random_walk_two.append(-1 if random() < 0.5 else 1)

103 for i in range(1, 1000):

104 movement = -1 if random() < 0.5 else 1

105 value = cello_random_walk_two[i-1] + movement

106 cello_random_walk_two.append(value)

107 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]

108 cello_chord_two = [-22, -8, -4, -8,]

109 cello_notes_two = [cello_chord_two[x] for x in reduceMod3(
 cello_random_walk_two)]

110

111 seed(1)

112 flute_random_walk_three = []

113 flute_random_walk_three.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 = 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(
 flute_random_walk_three)]

121

122 seed(4)

123 saxophone_random_walk_three = []

124 saxophone_random_walk_three.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 = saxophone_random_walk_three[i-1] + movement

128 saxophone_random_walk_three.append(value)

129 saxophone_random_walk_three = [abs(x) for x in
 saxophone_random_walk_three]

130 saxophone_chord_three = [-5, -4, 11, -4,]

131 saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod3(
 saxophone_random_walk_three)]

132

133 seed(8)

134 cello_random_walk_three = []

135 cello_random_walk_three.append(-1 if random() < 0.5 else 1)

136 for i in range(1, 1000):

137 movement = -1 if random() < 0.5 else 1

138 value = cello_random_walk_three[i-1] + movement

139 cello_random_walk_three.append(value)

140 cello_random_walk_three = [abs(x) for x in cello_random_walk_three]

141 cello_chord_three = [-11, -5, 0, -5,]

142 cello_notes_three = [cello_chord_three[x] for x in reduceMod3(
 cello_random_walk_three)]

143

144 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
 talea=abjadext.rmakers.Talea(
 counts=[3, 1, 3, 1, 3, 1, 3, 1, 3, 1],

```

147     denominator=16,
148     ),
149     beamSpecifier=abjadext.rmakers.BeamSpecifier(
150         beam_divisions_together=True,
151         beam_rests=False,
152         ),
153     extra_counts_per_division=[1, 0, -1, 0],
154     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
155         left_classes=[abjad.Note, abjad.Rest],
156         left_counts=[1, 0, 1],
157         ),
158     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
159         trivialize=True,
160         extract_trivial=True,
161         rewrite_rest_filled=True,
162         ),
163     )
164
165 rmaker_two = abjadext.rmakers.TaleaRhythmMaker(
166     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         beam_divisions_together=True,
172         beam_rests=False,
173         ),
174     extra_counts_per_division=[0, -1, 0, 1],
175     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
176         left_classes=[abjad.Note, abjad.Rest],
177         left_counts=[1, 0, 1],
178         ),
179     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
180         trivialize=True,
181         extract_trivial=True,
182         rewrite_rest_filled=True,
183         ),
184     )
185
186 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
187     denominators=[8, 16, 8, 16, 8, 16],
188     extra_counts_per_division=[-1, 0, 1, -1, 0, 1, 0, 0],
189     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
190         left_classes=[abjad.Rest],
191         left_counts=[1],
192         right_classes=[abjad.Rest],
193         right_counts=[1],
194         outer_divisions_only=True,
195         ),
196     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
197         trivialize=True,
198         extract_trivial=True,
199         rewrite_rest_filled=True,
200         ),

```

```

201     )
202
203 attachment_handler_one = AttachmentHandler(
204     starting_dynamic='f',
205     ending_dynamic='mp',
206     hairpin_indicator='--',
207     articulation='accent',
208 )
209
210 attachment_handler_two = AttachmentHandler(
211     starting_dynamic='fff',
212     ending_dynamic='mf',
213     hairpin_indicator='|>',
214     articulation='tenuto',
215 )
216
217 attachment_handler_three = AttachmentHandler(
218     starting_dynamic='pp',
219     ending_dynamic='mf',
220     hairpin_indicator='<',
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
418     Context'),
419     abjad.StaffGroup(
420         [
421             abjad.Staff([abjad.Voice(name='Voice 1')], name='Staff 1',
422             lilypond_type='Staff',),
423             abjad.Staff([abjad.Voice(name='Voice 3')], name='Staff 3',
424             lilypond_type='Staff',),
425             abjad.Staff([abjad.Voice(name='Voice 8')], name='Staff 8',
426             lilypond_type='Staff',),
427         ],
428         name='Staff Group 1',
429     ),
430     ],
431 )
432
433 for time_signature in time_signatures:
434     skip = abjad.Skip(1, multiplier=(time_signature))
435     abjad.attach(time_signature, skip)
436     score['Global Context'].append(skip)
437
438 print('Making containers ...')
439
440 def make_container(music_maker, durations):
441     selections = music_maker(durations)
442     container = abjad.Container([])
443     container.extend(selections)
444     return container
445
446 def key_function(timespan):
447     return timespan.annotation.music_maker or silence_maker
448
449 for voice_name, timespan_list in all_timespan_lists.items():
450     for music_maker, grouper in itertools.groupby(
451         timespan_list,
452         key=key_function,
453     ):
454         durations = [timespan.duration for timespan in grouper]
455         container = make_container(music_maker, durations)
456         voice = score[voice_name]
457         voice.append(container)
458
459 print('Splitting and rewriting ...')
460
461 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.
462     Voice):
463     for i, shard in enumerate(abjad.mutate(voice[:]).split(
464         time_signatures)):
465         time_signature = time_signatures[i]
466         abjad.mutate(shard).rewrite_meter(time_signature)
467
468 print('Beaming runs ...')
469
470 for voice in abjad.select(score).components(abjad.Voice):
471

```

```

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))
489                     and
490                     abjad.Duration(1, 4) <= previous_leaf.
491                     written_duration):
492                     left = leaf.written_duration.flag_count
493                     right = next_leaf.written_duration.flag_count
494                     beam_count = abjad.BeamCount(
495                         left=left,
496                         right=right,
497                     )
498                     abjad.attach(beam_count, leaf)
499
500     print('Stopping Hairpins ...')
501     for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
502         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 }')

```

```

§16 markup5 = abjad.Markup(r'\bold { E }')
§17 markup6 = abjad.Markup(r'\bold { F }')
§18 mark1 = abjad.RehearsalMark(markup=markup1)
§19 mark2 = abjad.RehearsalMark(markup=markup2)
§20 mark3 = abjad.RehearsalMark(markup=markup3)
§21 mark4 = abjad.RehearsalMark(markup=markup4)
§22 mark5 = abjad.RehearsalMark(markup=markup5)
§23 mark6 = abjad.RehearsalMark(markup=markup6)

§24
§25 def _apply_numerators_and_tech(staff, nums, tech):
§26     numerators = cyc(nums)
§27     techs = cyc(tech)
§28     for logical_tie in abjad.select(staff).logical_ties(pitched=True):
§29         tech = next(techs)
§30         numerator = next(numerators)
§31         bcp = abjad.BowContactPoint((numerator, 5))
§32         technis = abjad.BowMotionTechnique(tech)
§33         for note in logical_tie:
§34             abjad.attach(bcp, note)
§35             abjad.attach(technis, note)
§36     for run in abjad.select(staff).runs():
§37         abjad.bow_contact_spanner(run, omit_bow_changes=False)

§38
§39 instruments1 = cyc([
§40     abjad.Flute(),
§41     abjad.AltoSaxophone(),
§42     abjad.Cello(),
§43 ])
§44
§45 clefs1 = cyc([
§46     abjad.Clef('treble'),
§47     abjad.Clef('treble'),
§48     abjad.Clef('bass'),
§49 ])
§50
§51 abbreviations1 = cyc([
§52     abjad.MarginMarkup(markup=abjad.Markup('fl.')),
§53     abjad.MarginMarkup(markup=abjad.Markup('sx.')),
§54     abjad.MarginMarkup(markup=abjad.Markup('vc.')),
§55 ])
§56
§57 names1 = cyc([
§58     abjad.StartMarkup(markup=abjad.Markup('Flute')),
§59     abjad.StartMarkup(markup=abjad.Markup('Saxophone')),
§60     abjad.StartMarkup(markup=abjad.Markup('Violoncello')),
§61 ])
§62
§63 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
    Staff):
    leaf1 = abjad.select(staff).leaves()[0]
    abjad.attach(next(instruments1), leaf1)
    abjad.attach(next(abbreviations1), leaf1)
    abjad.attach(next(names1), leaf1)
    abjad.attach(next(clefs1), leaf1)

```

```

569
570 for staff in abjad.select(score['Staff Group 1']).components(abjad.Staff
571 )[0]:
572     leaf1 = abjad.select(staff).leaves()[0]
573     last_leaf = abjad.select(staff).leaves()[-1]
574     #abjad.attach(metro, leaf1)
575     abjad.attach(bar_line, last_leaf)
576
577 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
578 Staff):
579     abjad.Instrument.transpose_from_sounding_pitch(staff)
580
581 score_file = abjad.LilyPondFile.new(
582     score,
583     includes=['first_stylesheet.ily', '/Users/evansdsg2/abjad/docs/
584 source/_stylesheets/abjad.ily'],
585 )
586
587 abjad.SegmentMaker.comment_measure_numbers(score)
588 ######
589
590 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand
591 /Segments/Segment_II'
592 pdf_path = f'{directory}/Segment_II.pdf'
593 path = pathlib.Path('Segment_II.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.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
23     time_signatures])
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
51         i in lst2]
52
53 seed(1)
54 flute_random_walk_one = []
55 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
56 for i in range(1, 1000):
57     movement = -1 if random() < 0.5 else 1
58     value = flute_random_walk_one[i-1] + movement
59     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, 11, 12, 20, 31, 20, 12, 11, ]
60 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
    flute_random_walk_one)]
61
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 = [-10, 2, 11, 12, 1, 2, ]
71 saxophone_notes_one = [saxophone_chord_one[x] for x in reduceMod5(
    saxophone_random_walk_one)]
72
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 = [-18, -10, 2, -10, ]
82 cello_notes_one = [cello_chord_one[x] for x in reduceMod3(
    cello_random_walk_one)]
83
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(
    flute_random_walk_two)]
94
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 = [11, 20, ]
104 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod1(
    saxophone_random_walk_two)]
105
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 = [-18, 2, ]
115 cello_notes_two = [cello_chord_two[x] for x in reduceMod1(
116     cello_random_walk_two)]
117
118 seed(1)
119 flute_random_walk_three = []
120 flute_random_walk_three.append(-1 if random() < 0.5 else 1)
121 for i in range(1, 1000):
122     movement = -1 if random() < 0.5 else 1
123     value = flute_random_walk_three[i-1] + movement
124     flute_random_walk_three.append(value)
125 flute_random_walk_three = [abs(x) for x in flute_random_walk_three]
126 flute_chord_three = [11, 20, ]
127 flute_notes_three = [flute_chord_three[x] for x in reduceMod1(
128     flute_random_walk_three)]
129
130 seed(4)
131 saxophone_random_walk_three = []
132 saxophone_random_walk_three.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 = saxophone_random_walk_three[i-1] + movement
136     saxophone_random_walk_three.append(value)
137 saxophone_random_walk_three = [abs(x) for x in
138     saxophone_random_walk_three]
139 saxophone_chord_three = [2, 12, ]
140 saxophone_notes_three = [saxophone_chord_three[x] for x in reduceMod1(
141     saxophone_random_walk_three)]
142
143 seed(8)
144 cello_random_walk_three = []
145 cello_random_walk_three.append(-1 if random() < 0.5 else 1)
146 for i in range(1, 1000):
147     movement = -1 if random() < 0.5 else 1
148     value = cello_random_walk_three[i-1] + movement
149     cello_random_walk_three.append(value)
150 cello_random_walk_three = [abs(x) for x in cello_random_walk_three]
151 cello_notes_three = [-10, ]
152
153 rmaker_one = abjadext.rmakers.TaleaRhythmMaker(
154     talea=abjadext.rmakers.Talea(
155         counts=[2, 1, 3, 2, 2, 3, 1, ],
156         denominator=16,
157         ),
158     beamSpecifier=abjadext.rmakers.BeamSpecifier(
159         beamDivisionsTogether=True,
160         beamRests=False,

```

```

157     ),
158     extra_counts_per_division=[0, 1, 0, -1],
159     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
160         left_classes=[abjad.Note, abjad.Rest],
161         left_counts=[1, 0, 1],
162         ),
163     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
164         trivialize=True,
165         extract_trivial=True,
166         rewrite_rest_filled=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         beam_divisions_together=True,
177         beam_rests=False,
178         ),
179     extra_counts_per_division=[1, 0, -1, 0],
180     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
181         left_classes=[abjad.Note, abjad.Rest],
182         left_counts=[1, 0, 1],
183         ),
184     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
185         trivialize=True,
186         extract_trivial=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 attachmentHandler_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 celломusicmaker_one = MusicMaker(
269     rmaker=rmaker_one,
270     pitches=cello_notes_one,
271     continuous=True,
272     attachment_handler=attachment_handler_one,
273 )
274 celломusicmaker_two = MusicMaker(
275     rmaker=rmaker_two,
276     pitches=cello_notes_two,
277     continuous=True,
278     attachment_handler=attachment_handler_two,
279 )
280 celломusicmaker_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
421 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',),
    abjad.Staff([abjad.Voice(name='Voice 8')],name='Staff 8',
lilypond_type='Staff',),
],
    name='Staff Group 1',
),
],
),
)
)

431 for time_signature in time_signatures:
432     skip = abjad.Skip(1, multiplier=(time_signature))
433     abjad.attach(time_signature, skip)
434     score['Global Context'].append(skip)

435
436 print('Making containers ...')

437 def make_container(music_maker, durations):
438     selections = music_maker(durations)
439     container = abjad.Container([])
440     container.extend(selections)
441     return container

442
443 def key_function(timespan):
444     return timespan.annotation.music_maker or silence_maker

445
446 for voice_name, timespan_list in all_timespan_lists.items():
447     for music_maker, grouper in itertools.groupby(
448         timespan_list,
449         key=key_function,
450     ):
451         durations = [timespan.duration for timespan in grouper]
452         container = make_container(music_maker, durations)
453         voice = score[voice_name]
454         voice.append(container)

455
456 print('Splitting and rewriting ...')

457
458 for voice in abjad.iterate(score['Staff Group 1']).components(abjad.
459     Voice):
460     for i, shard in enumerate(abjad.mutate(voice[:]).split(
461         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/_stylesheets/abjad.ily'],
585     )
586
587 abjad.SegmentMaker.comment_measure_numbers(score)
588 ######
589
590 directory = '/Users/evansdsg2/Scores/four_ages_of_sand/four_ages_of_sand
    /Segments/Segment_III'
591 pdf_path = f'{directory}/Segment_III.pdf'
592 path = pathlib.Path('Segment_III.pdf')
593 if path.exists():
594     print(f'Removing {pdf_path} ...')
595     path.unlink()
596 time_1 = time.time()
597 print(f'Persisting {pdf_path} ...')
598 result = abjad.persist(score_file).as_pdf(pdf_path)
599 print(result[0])
600 print(result[1])
601 print(result[2])
602 success = result[3]
603 if success is False:
604     print('LilyPond failed!')
605 time_2 = time.time()
606 total_time = time_2 - time_1
607 print(f'Total time: {total_time} seconds')
608 if path.exists():
609     print(f'Opening {pdf_path} ...')
610     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
23     time_signatures])
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
51         i in lst2]
52
53 seed(1)
54 flute_random_walk_one = []
55 flute_random_walk_one.append(-1 if random() < 0.5 else 1)
56 for i in range(1, 1000):
57     movement = -1 if random() < 0.5 else 1
58     value = flute_random_walk_one[i-1] + movement
59     flute_random_walk_one.append(value)
60 flute_random_walk_one = [abs(x) for x in flute_random_walk_one]
61 flute_chord_one = [2, 13, 16, 20, 31, 20, 16, 13, ]
62 flute_notes_one = [flute_chord_one[x] for x in reduceMod7(
63     flute_random_walk_one)]
64
65 seed(4)
66 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
74 seed(8)
75 cello_random_walk_one = []
76 cello_random_walk_one.append(-1 if random() < 0.5 else 1)
77 for i in range(1, 1000):
78     movement = -1 if random() < 0.5 else 1
79     value = cello_random_walk_one[i-1] + movement
80     cello_random_walk_one.append(value)
81 cello_random_walk_one = [abs(x) for x in cello_random_walk_one]
82 cello_chord_one = [-21, -18, -14, -3, 2, -3, -14, -18, ]
83 cello_notes_one = [cello_chord_one[x] for x in reduceMod7(
84     cello_random_walk_one)]
85
86 seed(1)
87 flute_random_walk_two = []
88 flute_random_walk_two.append(-1 if random() < 0.5 else 1)
89 for i in range(1, 1000):
90     movement = -1 if random() < 0.5 else 1
91     value = flute_random_walk_two[i-1] + movement
92     flute_random_walk_two.append(value)
93 flute_random_walk_two = [abs(x) for x in flute_random_walk_two]
94 flute_chord_two = [2, 16, 31, 16, ]
95 flute_notes_two = [flute_chord_two[x] for x in reduceMod3(
96     flute_random_walk_two)]
97
98 seed(4)
99 saxophone_random_walk_two = []
100 saxophone_random_walk_two.append(-1 if random() < 0.5 else 1)
101 for i in range(1, 1000):
102     movement = -1 if random() < 0.5 else 1
103     value = saxophone_random_walk_two[i-1] + movement
104     saxophone_random_walk_two.append(value)
105 saxophone_random_walk_two = [abs(x) for x in saxophone_random_walk_two]
106 saxophone_chord_two = [-3, 13, ]
107 saxophone_notes_two = [saxophone_chord_two[x] for x in reduceMod1(
108     saxophone_random_walk_two)]
109
110 seed(8)
111 cello_random_walk_two = []
112 cello_random_walk_two.append(-1 if random() < 0.5 else 1)
113 for i in range(1, 1000):
114     movement = -1 if random() < 0.5 else 1
115     value = cello_random_walk_two[i-1] + movement
116     cello_random_walk_two.append(value)
117 cello_random_walk_two = [abs(x) for x in cello_random_walk_two]
```



```

163     ),
164     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
165         trivialize=True,
166         extract_trivial=True,
167         rewrite_rest_filled=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         beam_divisions_together=True,
178         beam_rests=False,
179         ),
180     extra_counts_per_division=[1, 0, -1, 0],
181     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
182         left_classes=[abjad.Note, abjad.Rest],
183         left_counts=[1, 0, 1],
184         ),
185     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
186         trivialize=True,
187         extract_trivial=True,
188         rewrite_rest_filled=True,
189         ),
190     )
191
192 rmaker_three = abjadext.rmakers.EvenDivisionRhythmMaker(
193     denominators=[16, 16, 8, 16, 16, 8],
194     extra_counts_per_division=[1, 0, 0, -1, 0, 1, -1, 0],
195     burnishSpecifier=abjadext.rmakers.BurnishSpecifier(
196         left_classes=[abjad.Rest],
197         left_counts=[1],
198         right_classes=[abjad.Rest],
199         right_counts=[1],
200         outer_divisions_only=True,
201         ),
202     tupletSpecifier=abjadext.rmakers.TupletSpecifier(
203         trivialize=True,
204         extract_trivial=True,
205         rewrite_rest_filled=True,
206         ),
207     )
208
209 attachment_handler_one = AttachmentHandler(
210     starting_dynamic='p',
211     ending_dynamic='mp',
212     hairpin_indicator='--',
213     articulation='accent',
214     )
215
216 attachment_handler_two = AttachmentHandler(

```

```

217     starting_dynamic='fff',
218     ending_dynamic='mf',
219     hairpin_indicator='>',
220     articulation='tenuto',
221 )
222
223 attachment_handler_three = AttachmentHandler(
224     starting_dynamic='mp',
225     ending_dynamic='ff',
226     hairpin_indicator='<|',
227     articulation='',
228 )
229
230 ##### oboe#####
231 flutemusicmaker_one = MusicMaker(
232     rmaker=rmaker_one,
233     pitches=flute_notes_one,
234     continuous=True,
235     attachment_handler=attachment_handler_one,
236 )
237 flutemusicmaker_two = MusicMaker(
238     rmaker=rmaker_two,
239     pitches=flute_notes_two,
240     continuous=True,
241     attachment_handler=attachment_handler_two,
242 )
243 flutemusicmaker_three = MusicMaker(
244     rmaker=rmaker_three,
245     pitches=flute_notes_three,
246     continuous=True,
247     attachment_handler=attachment_handler_three,
248 )
249 ##### saxophone#####
250 saxophonemusicmaker_one = MusicMaker(
251     rmaker=rmaker_one,
252     pitches=saxophone_notes_one,
253     continuous=True,
254     attachment_handler=attachment_handler_one,
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   celломusicmaker_two = MusicMaker(
276     rmaker=rmaker_two,
277     pitches=cello_notes_two,
278     continuous=True,
279     attachment_handler=attachment_handler_two,
280   )
281   celломusicmaker_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), cellomusicmaker_one],
366         [(18, 8), (22, 8), cellomusicmaker_two],
367         [(25, 8), (29, 8), cellomusicmaker_three],
368         [(29, 8), (32, 8), cellomusicmaker_one],
369         [(35, 8), (39, 8), cellomusicmaker_two],
370         [(39, 8), (42, 8), cellomusicmaker_three],
371         [(45, 8), (50, 8), cellomusicmaker_one],
372         [(50, 8), (52, 8), cellomusicmaker_two],
373         [(55, 8), (56, 8), cellomusicmaker_three],
374         [(56, 8), (61, 8), cellomusicmaker_one],
375         [(61, 8), (62, 8), cellomusicmaker_two],
376         [(65, 8), (69, 8), cellomusicmaker_three],
377         [(69, 8), (72, 8), cellomusicmaker_one],
378         [(75, 8), (79, 8), cellomusicmaker_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 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
421     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         name='Staff Group 1',
432     ),

```

```

429 ],
430 )
431
432 for time_signature in time_signatures:
433     skip = abjad.Skip(1, multiplier=(time_signature))
434     abjad.attach(time_signature, skip)
435     score['Global Context'].append(skip)
436
437 print('Making containers ...')
438
439 def make_container(music_maker, durations):
440     selections = music_maker(durations)
441     container = abjad.Container([])
442     container.extend(selections)
443     return container
444
445 def key_function(timespan):
446     return timespan.annotation.music_maker or silence_maker
447
448 for voice_name, timespan_list in all_timespan_lists.items():
449     for music_maker, grouper in itertools.groupby(
450         timespan_list,
451         key=key_function,
452     ):
453         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(
462         time_signatures)):
463         time_signature = time_signatures[i]
464         abjad.mutate(shard).rewrite_meter(time_signature)
465
466 print('Beaming runs ...')
467
468 for voice in abjad.select(score).components(abjad.Voice):
469     for run in abjad.select(voice).runs():
470         if 1 < len(run):
471             specifier = abjadext.rmakers.BeamSpecifier(
472                 beam_each_division=False,
473                 )
474             specifier(run)
475             abjad.attach(abjad.StartBeam(), run[0])
476             abjad.attach(abjad.StopBeam(), run[-1])
477             for leaf in run:
478                 if abjad.Duration(1, 4) <= leaf.written_duration:
479                     continue
480                 previous_leaf = abjad.inspect(leaf).leaf(-1)
481                 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)))
492                 and
493                     abjad.Duration(1, 4) <= previous_leaf.
494                     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)

501 print('Stopping Hairpins ...')
502 for staff in abjad.iterate(score['Staff Group 1']).components(abjad.
503     Staff):
504     for rest in abjad.iterate(staff).components(abjad.Rest):
505         previous_leaf = abjad.inspect(rest).leaf(-1)
506         if isinstance(previous_leaf, abjad.Note):
507             abjad.attach(abjad.StopHairpin(), rest)
508         elif isinstance(previous_leaf, abjad.Chord):
509             abjad.attach(abjad.StopHairpin(), rest)
510         elif isinstance(previous_leaf, abjad.Rest):
511             pass

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 def _apply_numerators_and_tech(staff, nums, tech):
528     numerators = cyc(nums)
529     techs = cyc(tech)
530     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/
585     source/_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 "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   "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-
82     invisible
83   \override TimeSignature.font-size = #4
84   \override TimeSignature.self-alignment-X = #center
85   \override VerticalAxisGroup.default-staff-staff-spacing = #'(
86     (basic-distance . 0) (minimum-distance . 10) (padding . 6) (
87       stretchability . 0)
88   )
89   }
90
91 \context {
92   \Score
93   \remove Bar_number_engraver
94   \remove Mark_engraver
95   \accepts TimeSignatureContext
96   \override BarLine.bar-extent = #'(-2 . 2)
97   \override Beam.breakable = ##t
98   \override Beam.concaveness = #10000
99   \override Beam.beam-thickness = #0.8
100  \override Beam.length-fraction = #1.5
101  \override DynamicText.font-size = #-2
102  \override Glissando.breakable = ##t
103  \override SpacingSpanner.strict-grace-spacing = ##t
104  \override SpacingSpanner.strict-note-spacing = ##t
105  \override SpacingSpanner.uniform-stretching = ##t
106  \override StaffGrouper.staff-staff-spacing = #'(
107    (basic-distance . 14) (minimum-distance . 14) (padding . 1)
108    s)
109  \override Stem.thickness = #0.75
110  \override TupletBracket.bracket-visibility = ##t

```

```
108     \override TupletBracket.minimum-length = #3
109     \override TupletBracket.padding = #2
110     \override TupletBracket.springs-and-rods = #ly:spanner::set-
111         spacing-rods
112     \override TupletNumber.text = #tuplet-number::calc-fraction-text
113     proportionalNotationDuration = #(ly:make-moment 1 40)
114     autoBeaming = ##f
115     tupletFullLength = ##t
116 }
117 \context {
118     \Voice
119     \remove Forbid_line_break_engraver
120 }
121 \context {
122     \Staff
123     \remove Time_signature_engraver
124     \hide BarLine
125 }
126 \context {
127     \RhythmicStaff
128     \remove Time_signature_engraver
129 }
130 \context {
131     \StaffGroup
132 }
133
134 \paper {
135     top-margin = 1.5\cm
136     bottom-margin = 1.5\cm
137
138     %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

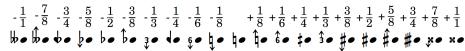
FOREWORD

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
11.12
st.
mf
p
mp
msp.
7.8
15.16
mp
ord.

Cchar -2- Evans

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

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

ff
ord. sp.
mp
sl. 8.8
ff
mf
p
mp

10 $\frac{4}{4}$

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

ff
msp. 5.4
ff
mf
ord.
mf

Cchar ~3~ Evans

13 **3**

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

ff
ord.
sp.
fff

23.24
11.12
13.12

mf

14

st.
p
ord.
sp.
fff
mf

15 **4**

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

ord.
ff
ff
ord.
ff
p

78
9.8
17.16

mf
mf
mf
mp

4

sp.
ord.
ff
17.16

mf

Cchar ~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

Musical score for orchestra and piano, page 21, measures 3 and 4. The score includes parts for B.H., vc.I, L.H., B.H., ve.II, L.H., and piano. Measure 3 starts with a piano solo. Measure 4 begins with a dynamic **ff**. Various instruments play eighth-note patterns, often starting with a piano note followed by a woodwind or brass entry. The piano part features sustained notes and eighth-note chords. Dynamics include **ord.**, **ff**, **mf**, **sp.**, **st.**, **sl.**, **8va...**, and **mp**.

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

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

ff
mf
ff
ff
mf
p
mp

ord.
sp.
7.8
ord.
31.32
st.

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

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

ff
ff
ff
ff
ff
ff

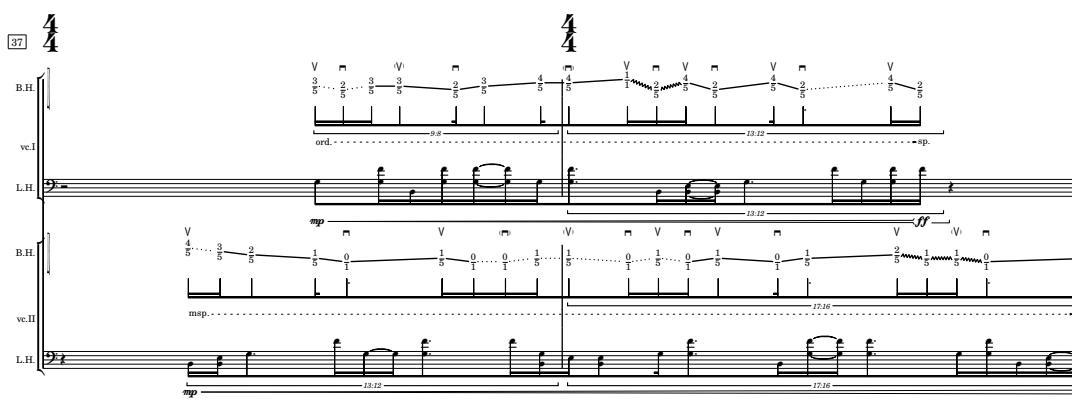
msp.
ord.
15.16
mp
ff
ff
ff

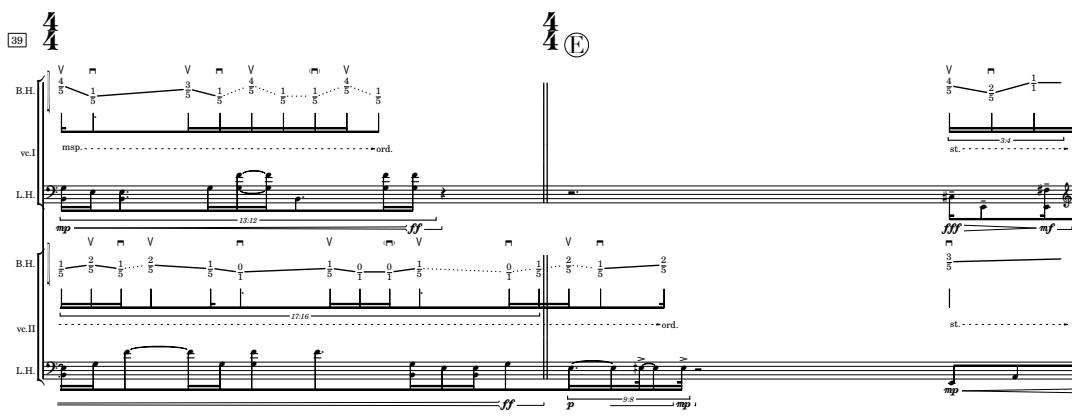
ff
ff
ff
ff
ff
ff

Cchar ~7~ Evans

Musical score for orchestra and piano, page 29, measures 29-30. The score includes parts for Bassoon (B.H.), Trombone (ve.I), Trombone (ve.II), Bassoon (L.H.), and Piano (L.H.). Measure 29 starts with a dynamic *ff*. The piano part has a sustained note with a fermata. Measures 29-30 show various rhythmic patterns and dynamics (e.g., *ord.*, *msp.*, *11:12*, *mp*, *mf*). Measure 30 ends with a dynamic *ff*.

Musical score for orchestra and piano, page 21, measures 3 and 5. The score includes parts for B.H., ve.I, L.H., B.H., ve.II, and L.H. Measure 3 (4/4 time) shows B.H. and ve.I playing eighth-note patterns. L.H. has sixteenth-note patterns. B.H. and ve.II play eighth-note patterns. L.H. has sixteenth-note patterns. Measure 5 (17/16 time) shows B.H. and ve.I playing eighth-note patterns. L.H. has sixteenth-note patterns. B.H. and ve.II play eighth-note patterns. L.H. has sixteenth-note patterns. The piano part in measure 5 includes a melodic line with various dynamics and markings like *msp.*, *17/16*, *ord.*, *sp.*, *msp.*, *12.22*, *ff*, *ord.*, *9.8*, and *2.5*.

[37] 

[38] 

[41]

5

B.H.

vc.I

L.H.

B.H.

vc.II

L.H.

ord.

mp

78

ff

sp.

78

ff

13.12

ff

11.12

4

[42]

5

B.H.

vc.I

L.H.

B.H.

vc.II

L.H.

msp.

9.8

mf

ord.

msp.

9.8

mf

Sev.

mp

ff

78

st.

ord.

78

st.

mp

p

>

mp

4

[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 ord. 17.16
L.H. 17.16
B.H. V
vc.II 17.16
L.H. 17.16 mf

[47] 5

B.H. 3 5
vc.I 5d sp.
L.H. 3.2
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

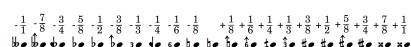
Tiānshū is the name of an art installation in the form of a book by artist Xu Bing filled with meaningless glyphs in the style of traditional Chinese characters. The term tiān shū, which can be translated to mean "divine writing," originally referred to religious texts but is now used to mean "gibberish."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

天書

Tianshu

for twelve players

Gregory Rowland Evans

J = 108

Flute

Clarinet

Bassoon

Horn

Trumpet

Trombone

Tuba

Violin I

Violin II

Viola

Violoncello

Contrabass

Gregory Rowland Evans

4 3 4 4

fl.

cl.

bassn.

hr.

trp.

trmb.

th.

vln.I

vln.II

vla.

vc.

ch.

fff *mf*

78 *mp* *ff* *p*

ffff *mf*

11.12 *mp*

mp *ff* *p*

ffff *mf*

mp *78* *mf*

mp *ff* *p*

ffff *mf*

mp *ff* *p*

-2-

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

fl. \textit{mp} \textit{ff}

cl. \textit{mp}

bassn. \textit{ff}

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

hr. \textit{mp}

trp. \textit{mp} \textit{ff}

trmb. \textit{ff}

th. \textit{mp}

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

vln.I 12:12 \textit{mp} \textit{ff}

vin.II \textit{mp}

vla. \textit{mp}

vc. 3:4 \textit{ff}

cb. \textit{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

19

fl.

cl.

bassn.

19

hr.

trp.

trmb.

th.

19

vln.I

vln.II

vla.

vc.

cb.

fff $\xrightarrow{3-4}$ $\xrightarrow{5-6}$ $\xrightarrow{7-8}$ $\xrightarrow{9-10}$ $\xrightarrow{11-12}$ $\xrightarrow{12-13}$ $\xrightarrow{3-4}$

$\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{mp} \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{11-12}$ \xrightarrow{mf} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

$\xrightarrow{8va\dots}$ $\xrightarrow{5-6}$ \xrightarrow{ff} \xrightarrow{mp} \xrightarrow{ff} $\xrightarrow{9.8}$ \xrightarrow{mf} \xrightarrow{ff} \xrightarrow{p}

[23]

fl.

cl.

bassoon

[23]

hr.

trp.

trmb.

th.

[23]

vn.I

vn.II

vla.

vc.

cb.

[26] 

29

3

fl. *mp*

cl. *mp*

bsn.

29

3 (D)

hr.

trp. *mp*

trmb.

tb.

29

3 (D)

vln.I *mp*

vln.II *mp*

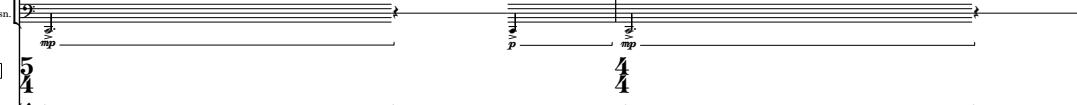
vla. *mp*

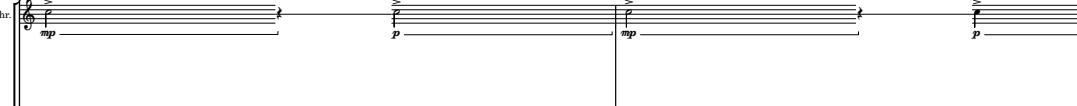
vc.

cb.

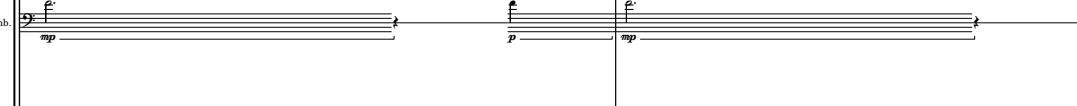
[32] 

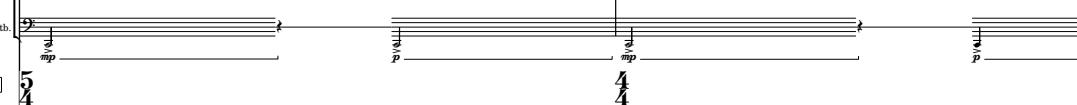




[32] 

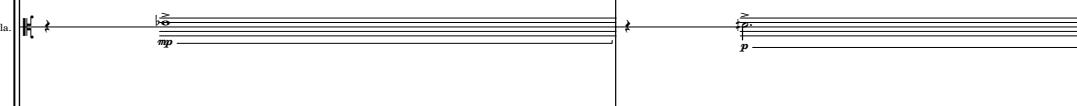


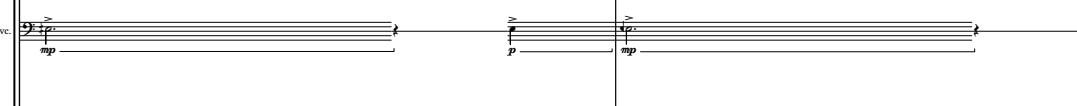


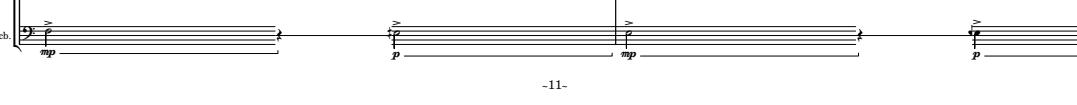


[32] 









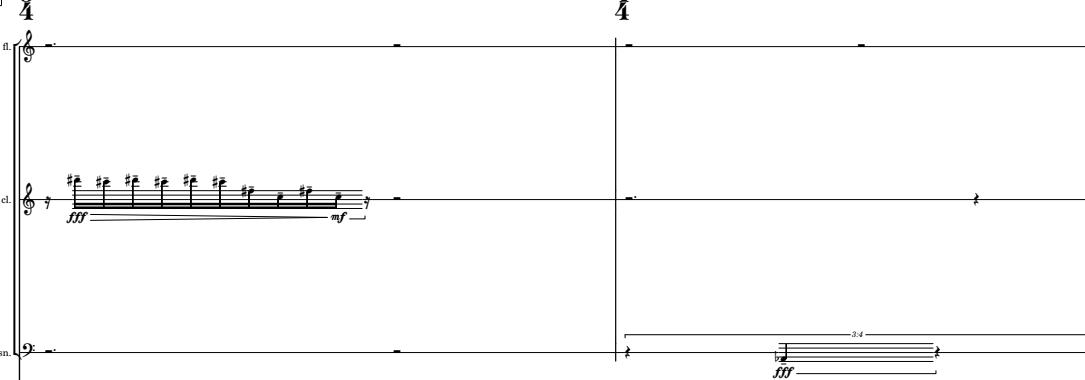
[36]

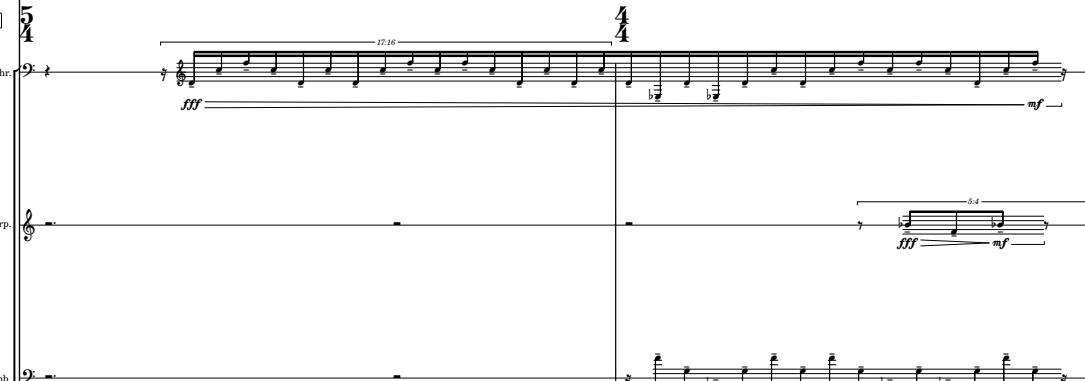
[36]

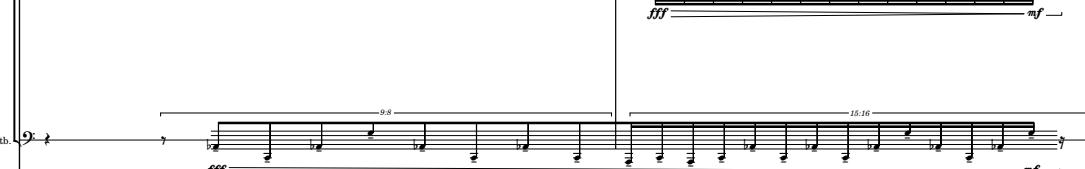
[36]

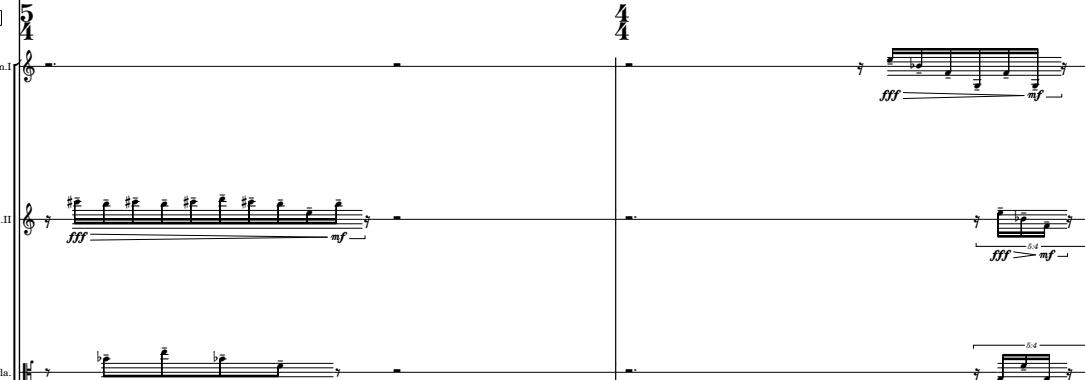
[36]

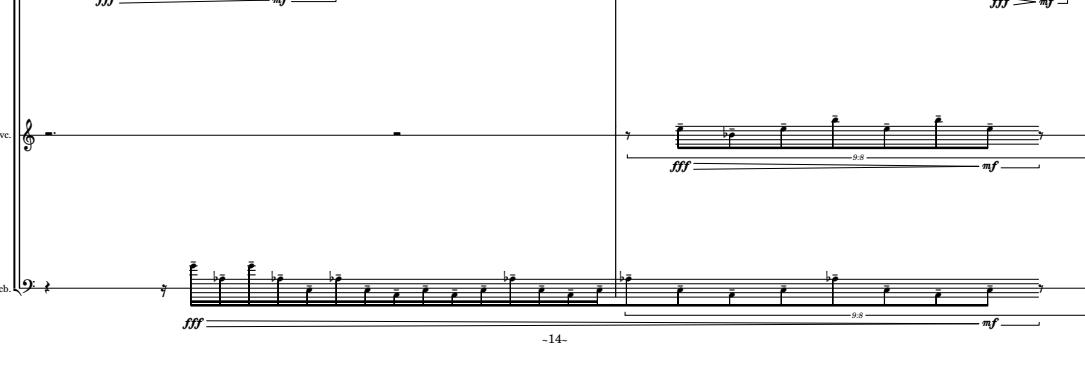
[36]

[38] 

[38] 

[38] 

[38] 

[38] 

[38] 

[40] $\frac{4}{4}$ (F) $\frac{5}{4}$

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

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

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

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

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

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

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

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

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

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

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

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

Flute: mp ff ff
Clarinet: mp ff
Bassoon: mp ff
Horn: mp s.d. ff
Trumpet: mp ff
Trombone: mp ff
Tuba: mp ff
Violin I: mp ff ff
Violin II: mp ff
Viola: mp ff
Cello: mp ff
Double Bass: mp s.s. ff

-15-

44

fl. *mp* *ff*

cl. *ff*

bassn.

44

hr. *mp* *ff*

trp. *mp* *ff*

trmb.

tb.

44

vln.I *mp* *ff*

vln.II *ff*

vla.

vc. *mp* *ff*

cb.

47

fl.

el.

bassn.

47

hr.

trp.

trmb.

thb.

vln.I

vln.II

vla.

vc.

cb.

3

5

3

5

5.4

13.12

mp

ff

15.16

5.4

13.12

mp

ff

15.16

8.8

mp

ff

15.16

13.12

mp

ff

15.16

7.8

mp

ff

15.16

15.16

mp

ff

Sva.....

15.16

mp

ff

7.8

mp

ff

Sva.....

17.16

mp

ff

13.12

mp

ff

15.16

mp

ff

17.16

mp

ff

[50] $\frac{5}{4}$ $\downarrow = 90$

fl.

cl.

bssn.

50

5
4

12.12

2
4

7.8

mf ! p

hr.

trp.

trmb.

tb.

vln.I

vln.II

vla.

vc.

ch.

19.20

mp

15.16

6.5

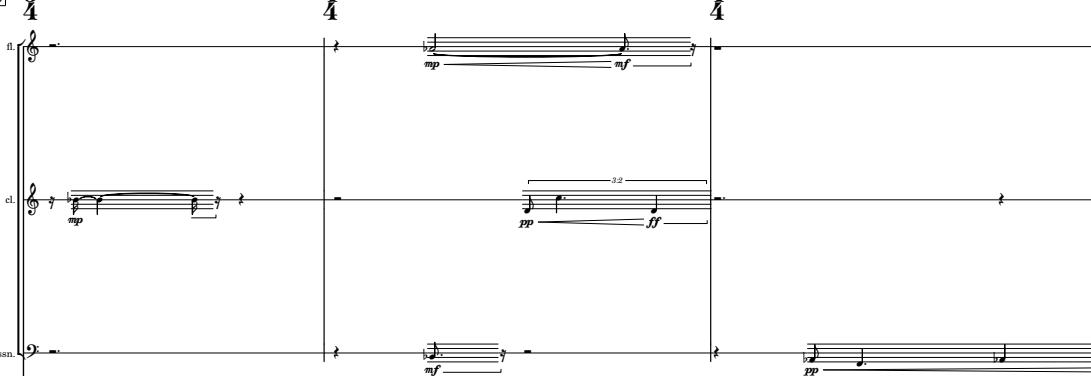
p

p

12.16

mf

-19-

[53] 

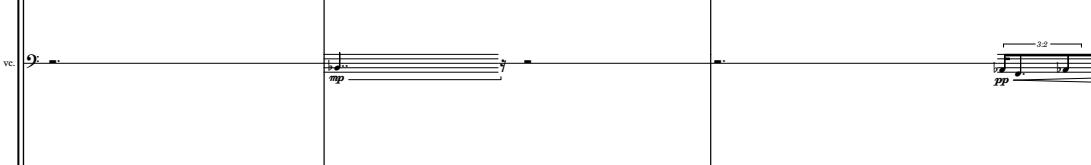
[53] 

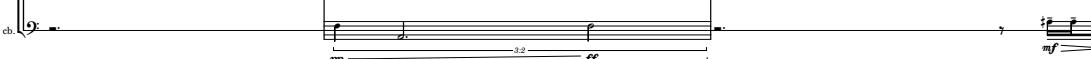
[53] 

[53] 

[53] 

[53] 

[53] 

[53] 

56

4

fl. *pp* *ff*

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

bassn. *ff* *3:2*

56

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

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

trmb.

tb. *p*

56

vln.I *pp* *ff*

vln.II *p* *3:2*

vla. *p* *11:12*

vc. *ff* *3:2*

ch. *p* *3:2*

4 (G)

mf *13:12*

mf *11:12*

mf

mf *8:6* *7:6* *p*

mp

mf *p*

mf *p*

[58]

[58]

[59]

[63]  

cl.   

bassn.   

[63]   

trp.  

trmb.   

th.   

[63]   

vln.II  

vla.   

vc.   

ch.   

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

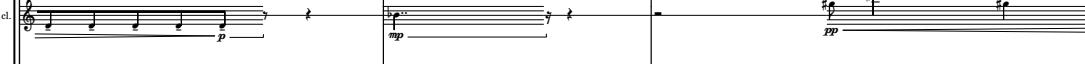
vla. $\frac{2}{4}$ $\frac{3}{4}$ $\frac{4}{4}$ $\frac{3}{4}$

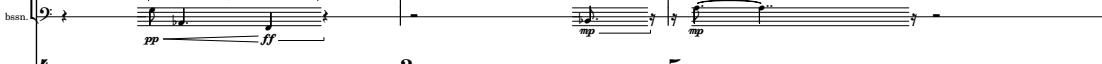
vc. $\frac{2}{4}$ $\frac{3}{4}$ $\frac{4}{4}$ $\frac{3}{4}$

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

72   

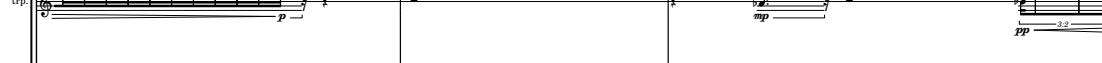
fl. 

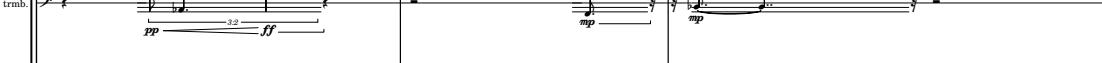
cl. 

bsn. 

72   

hr. 

trp. 

trmb. 

th. 

72   

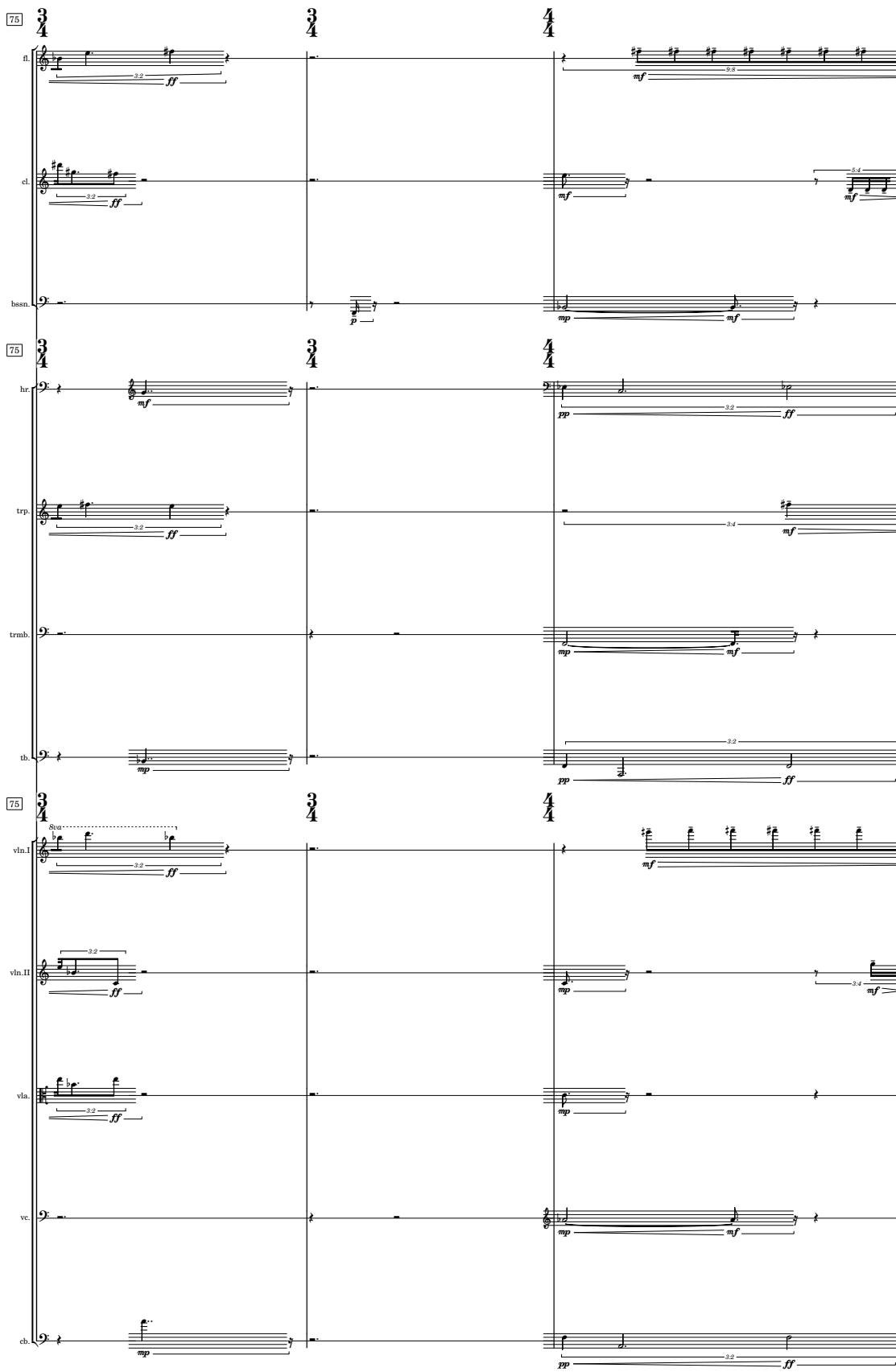
vln.I 

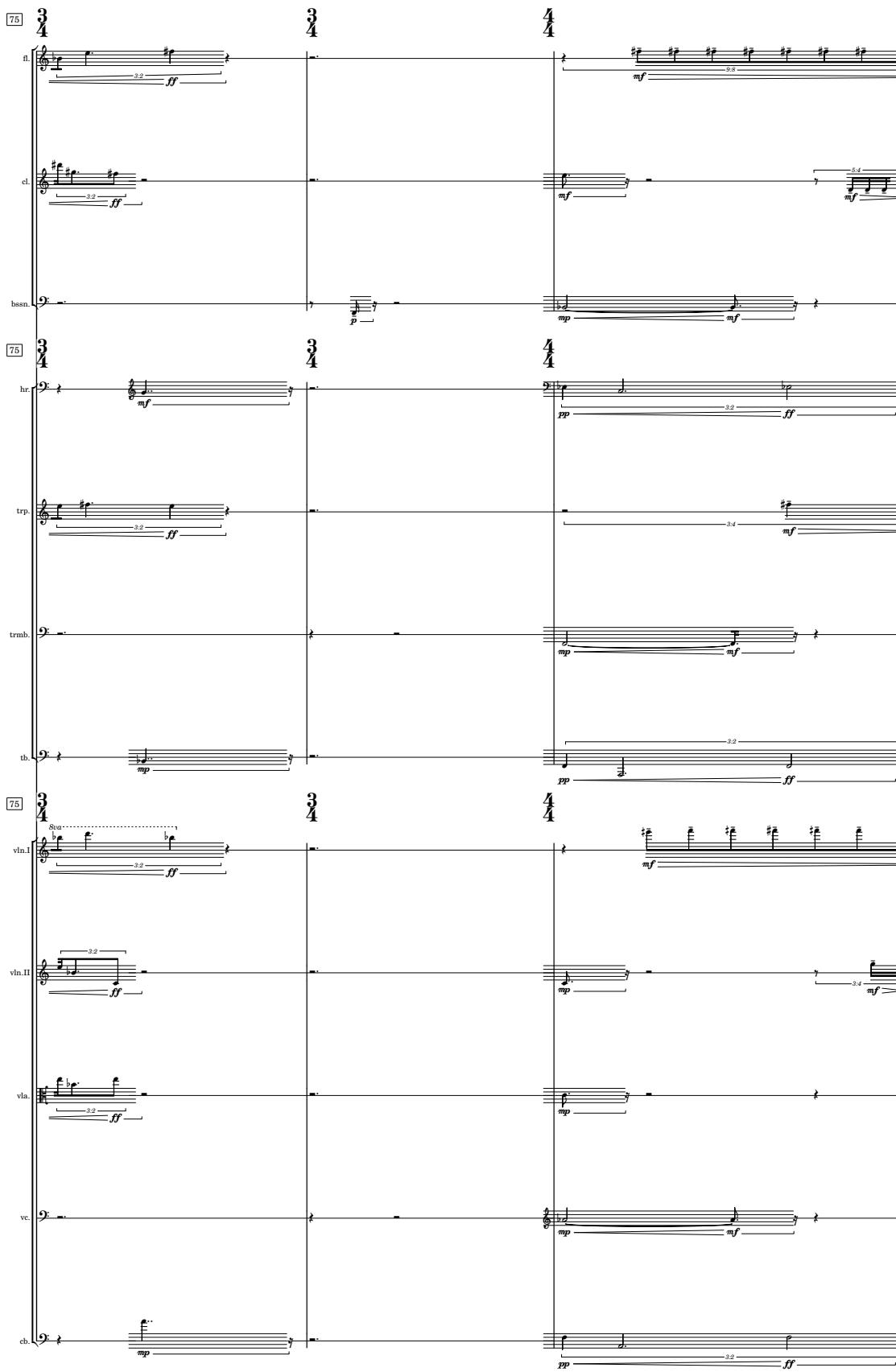
vln.II 

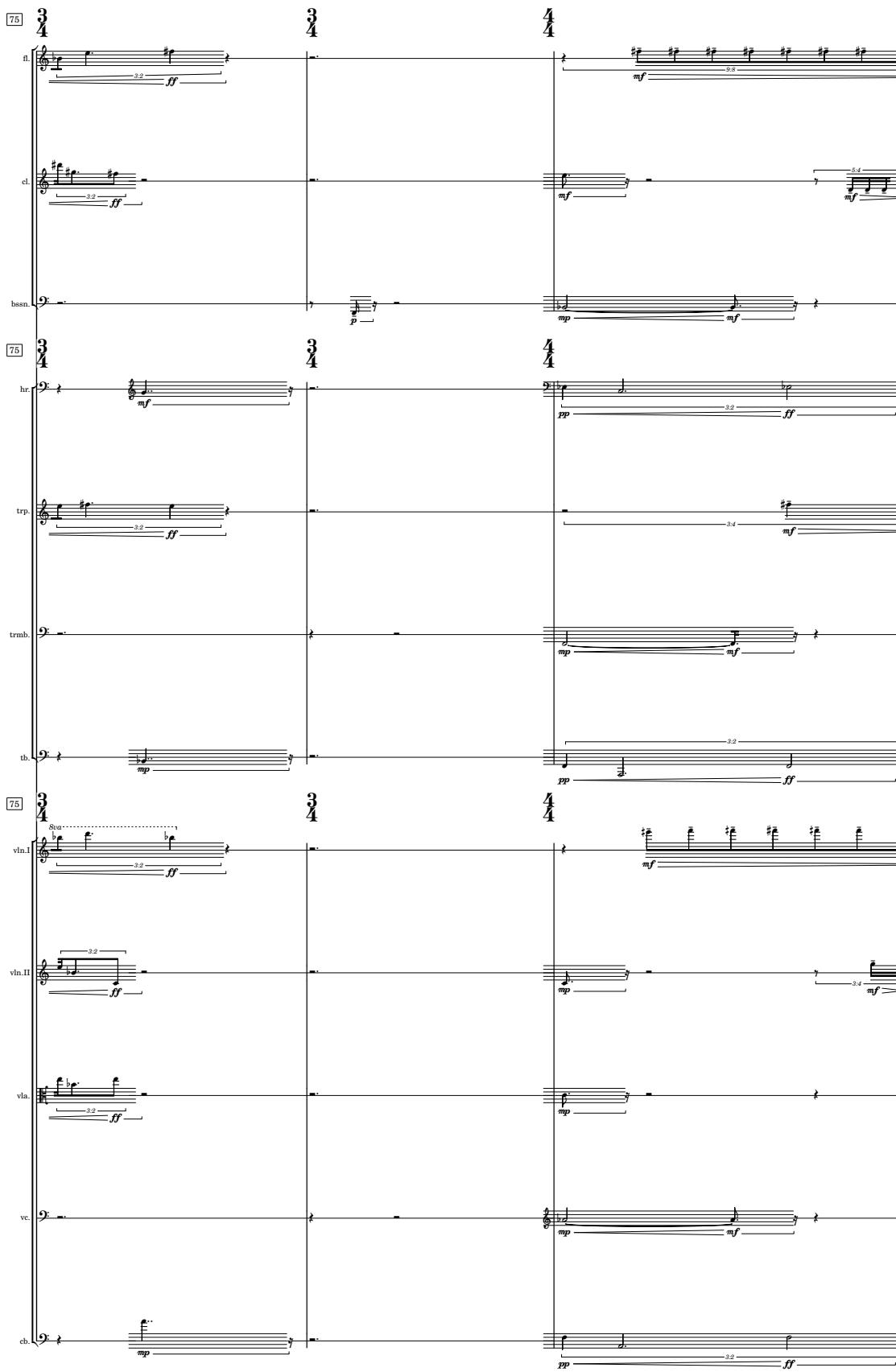
vla. 

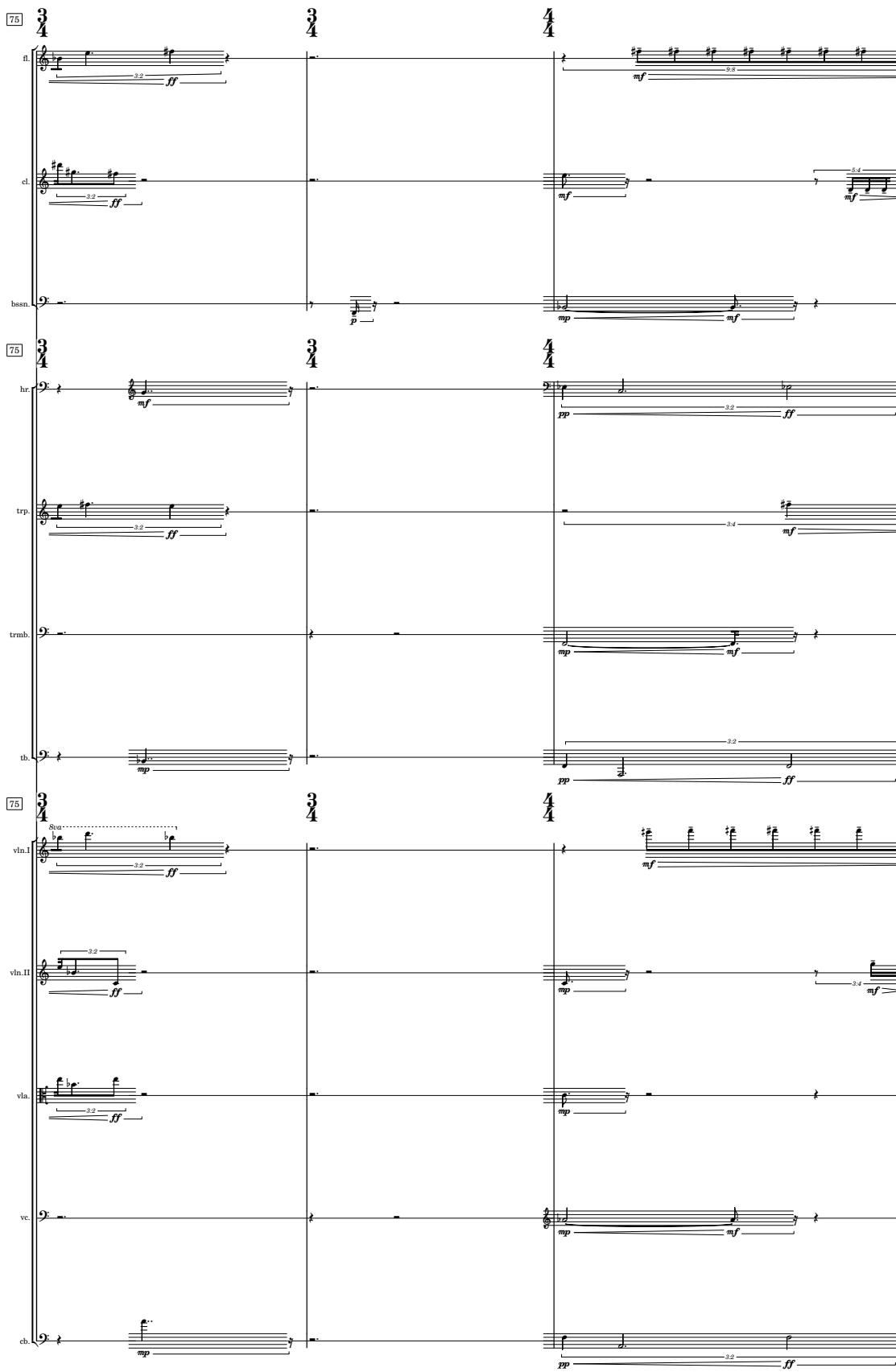
vc. 

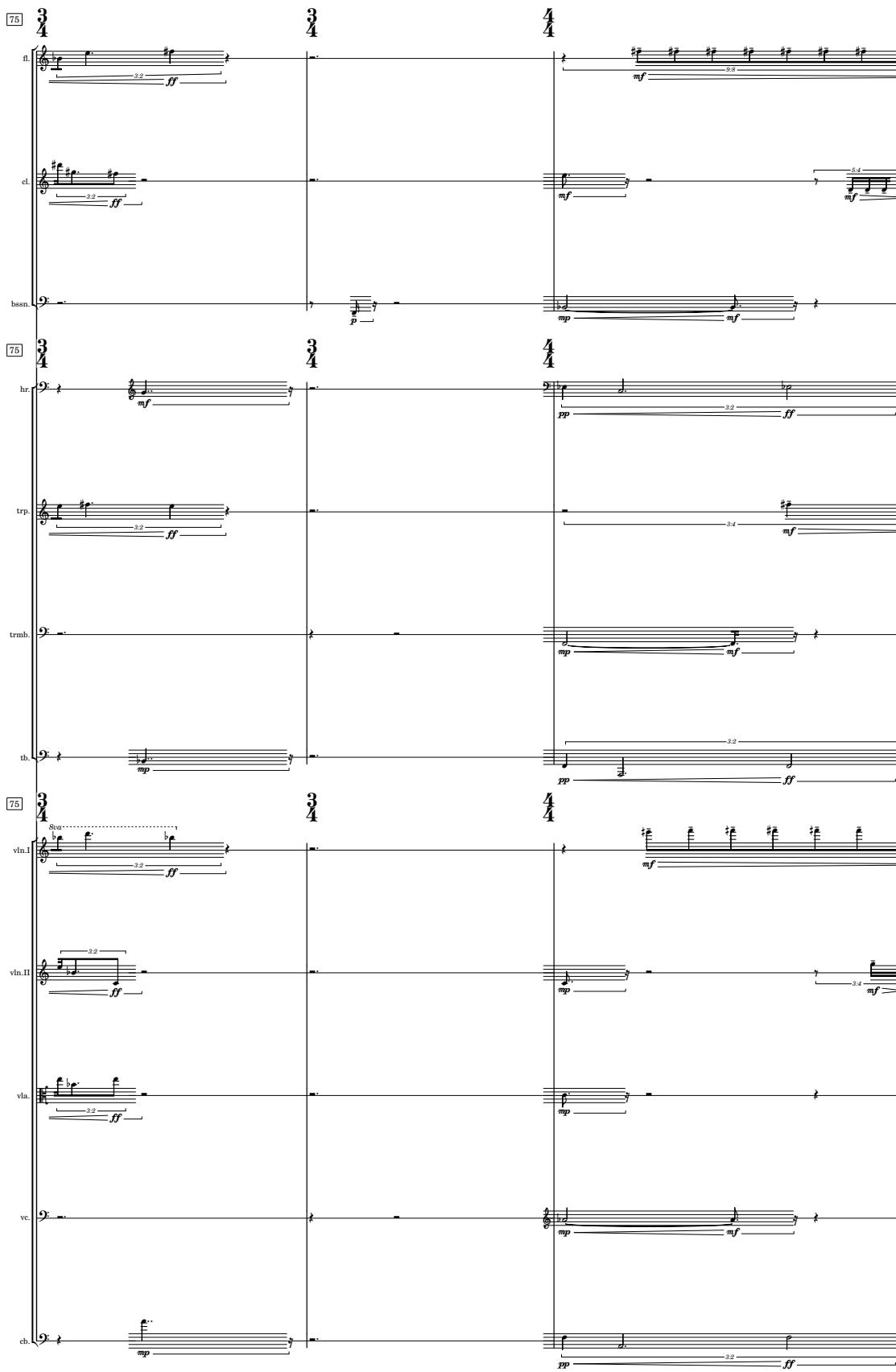
ch. 

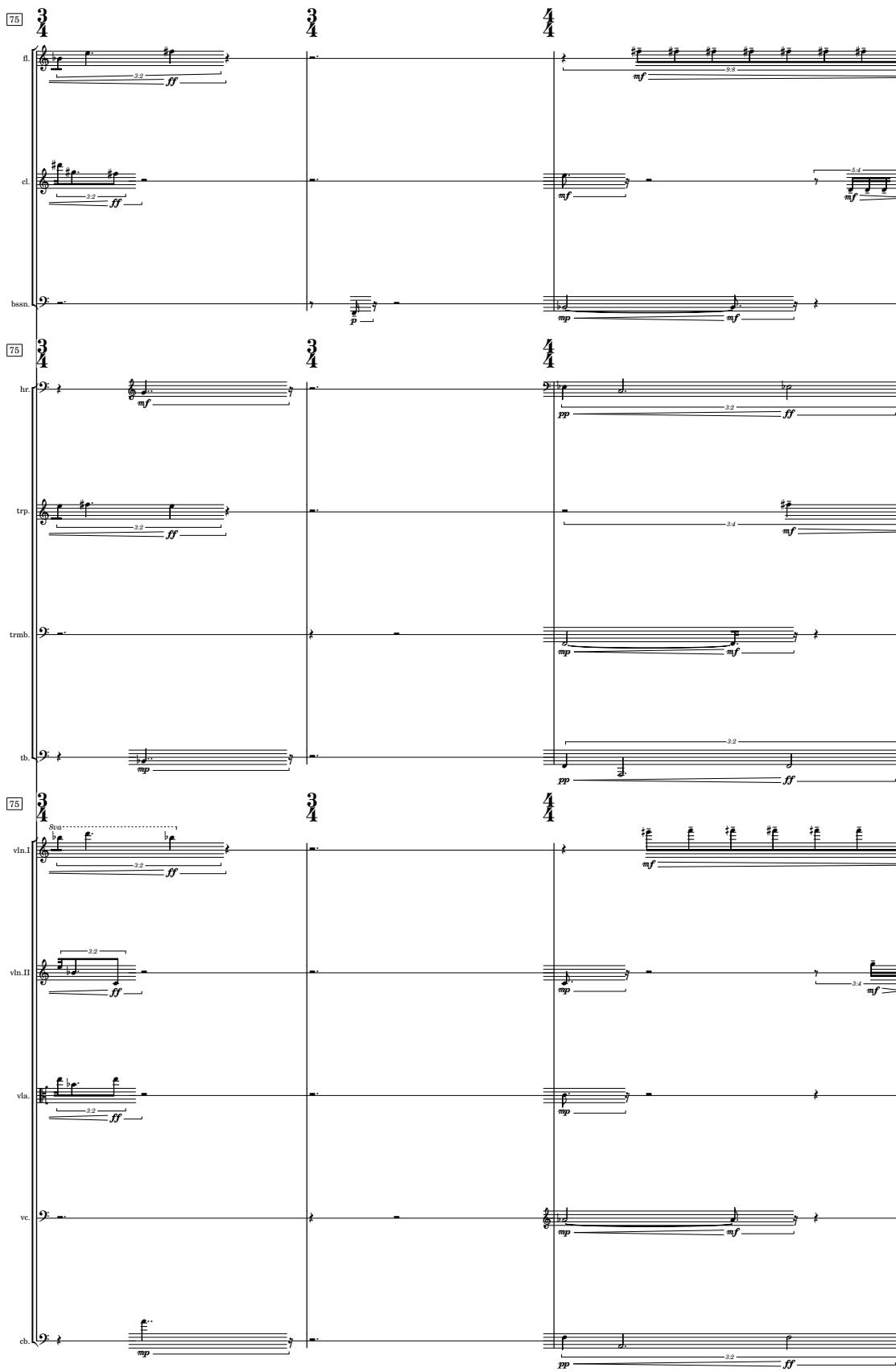
75 

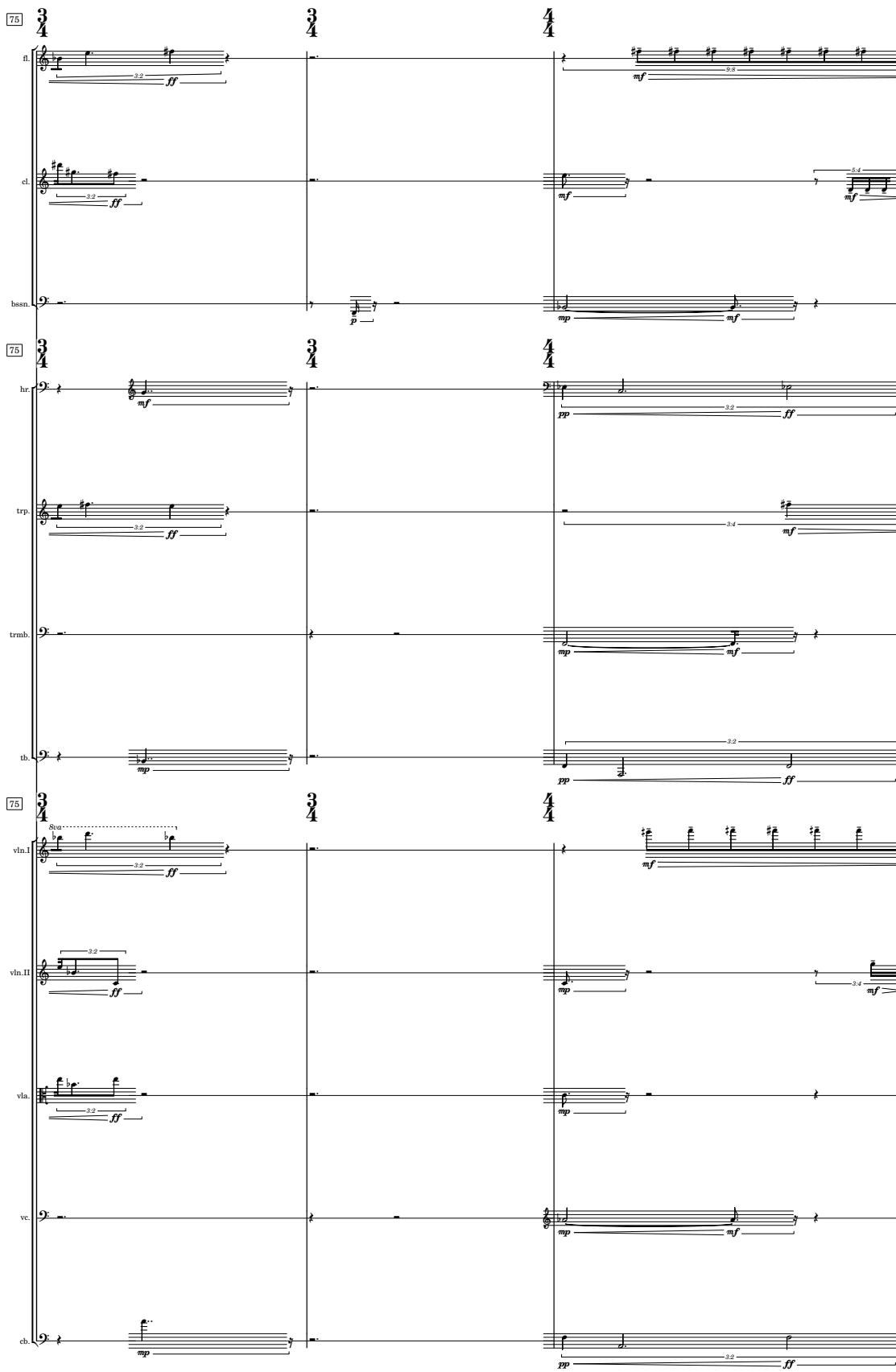
76 

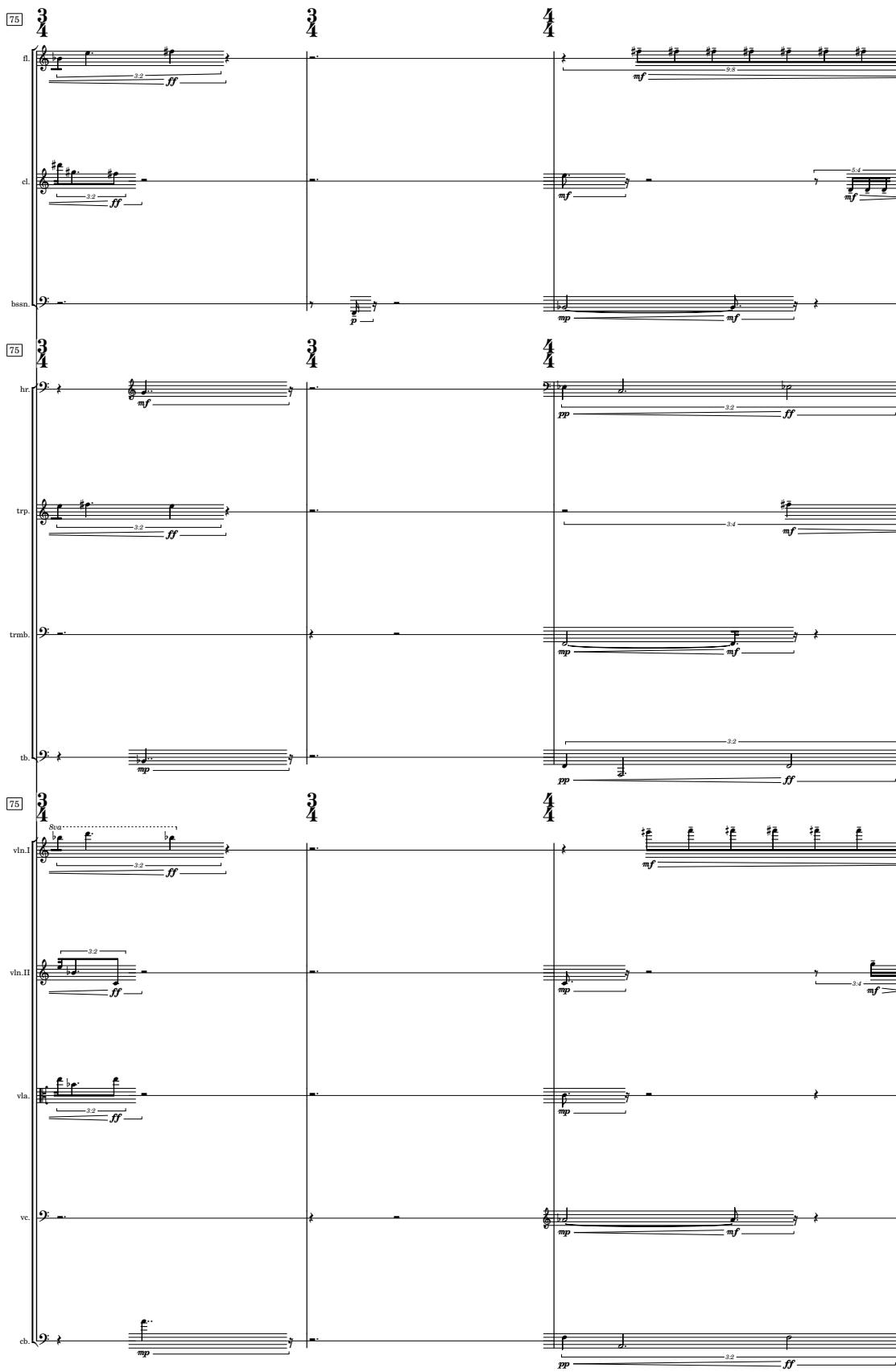
77 

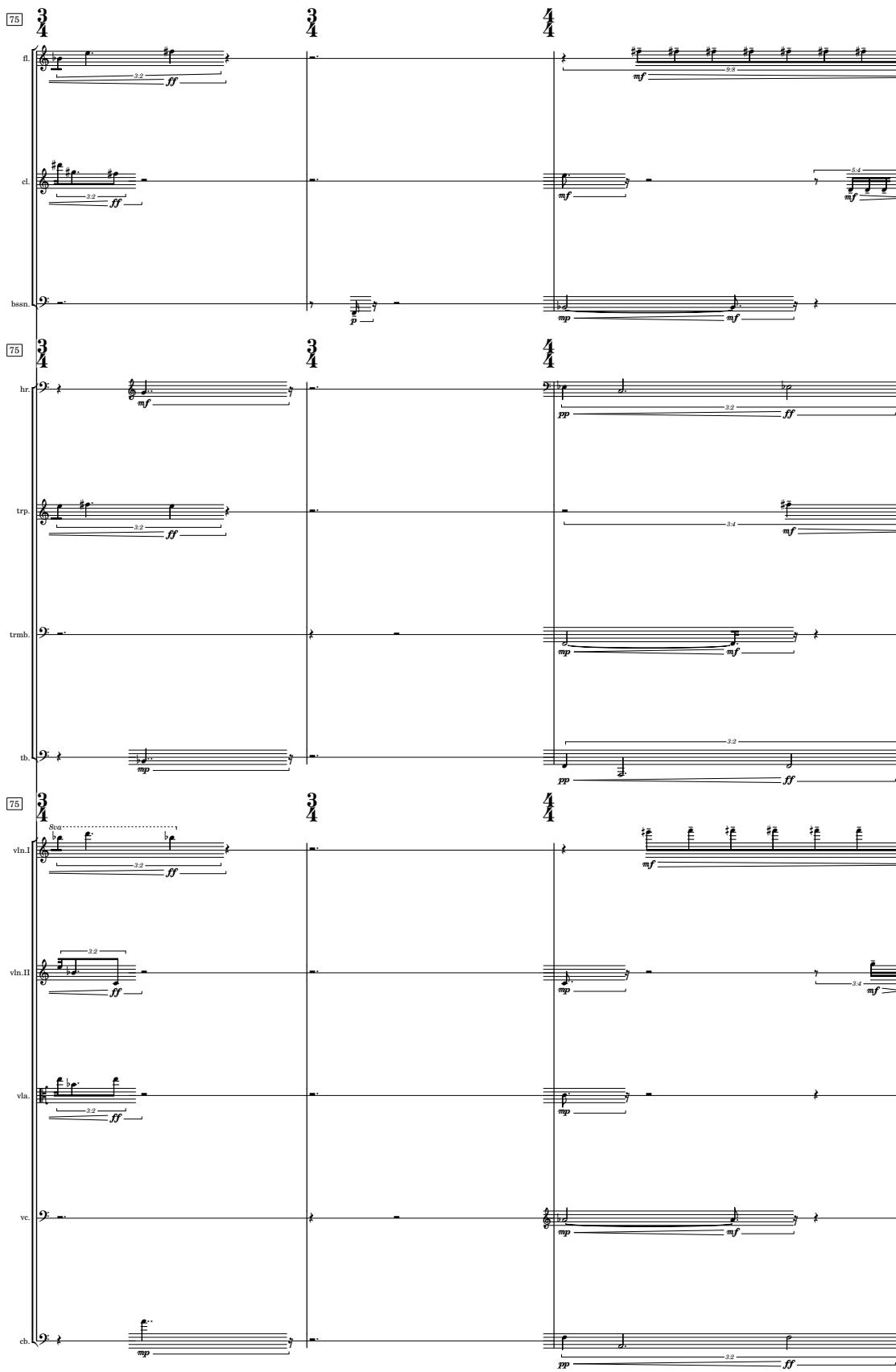
78 

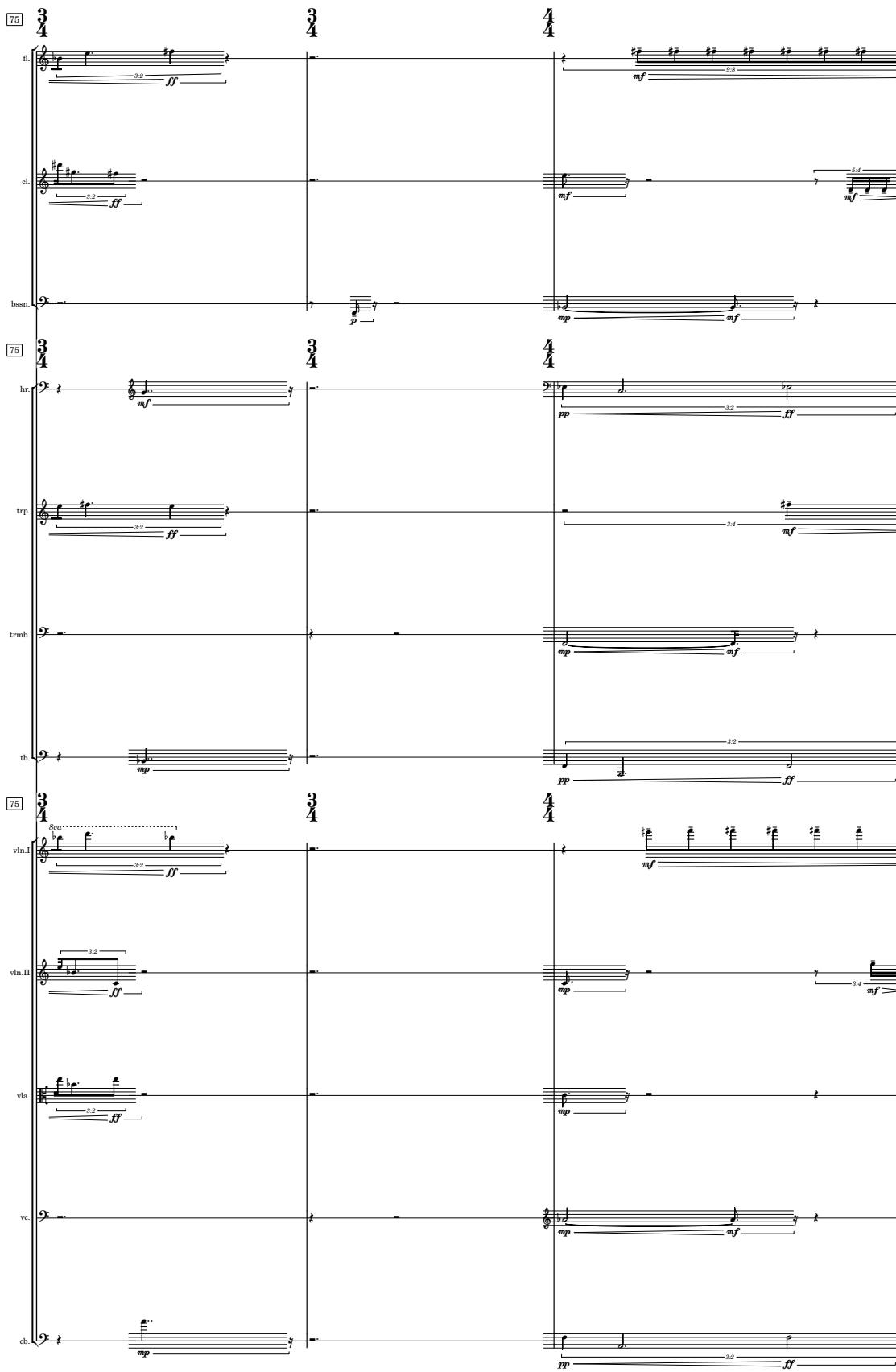
79 

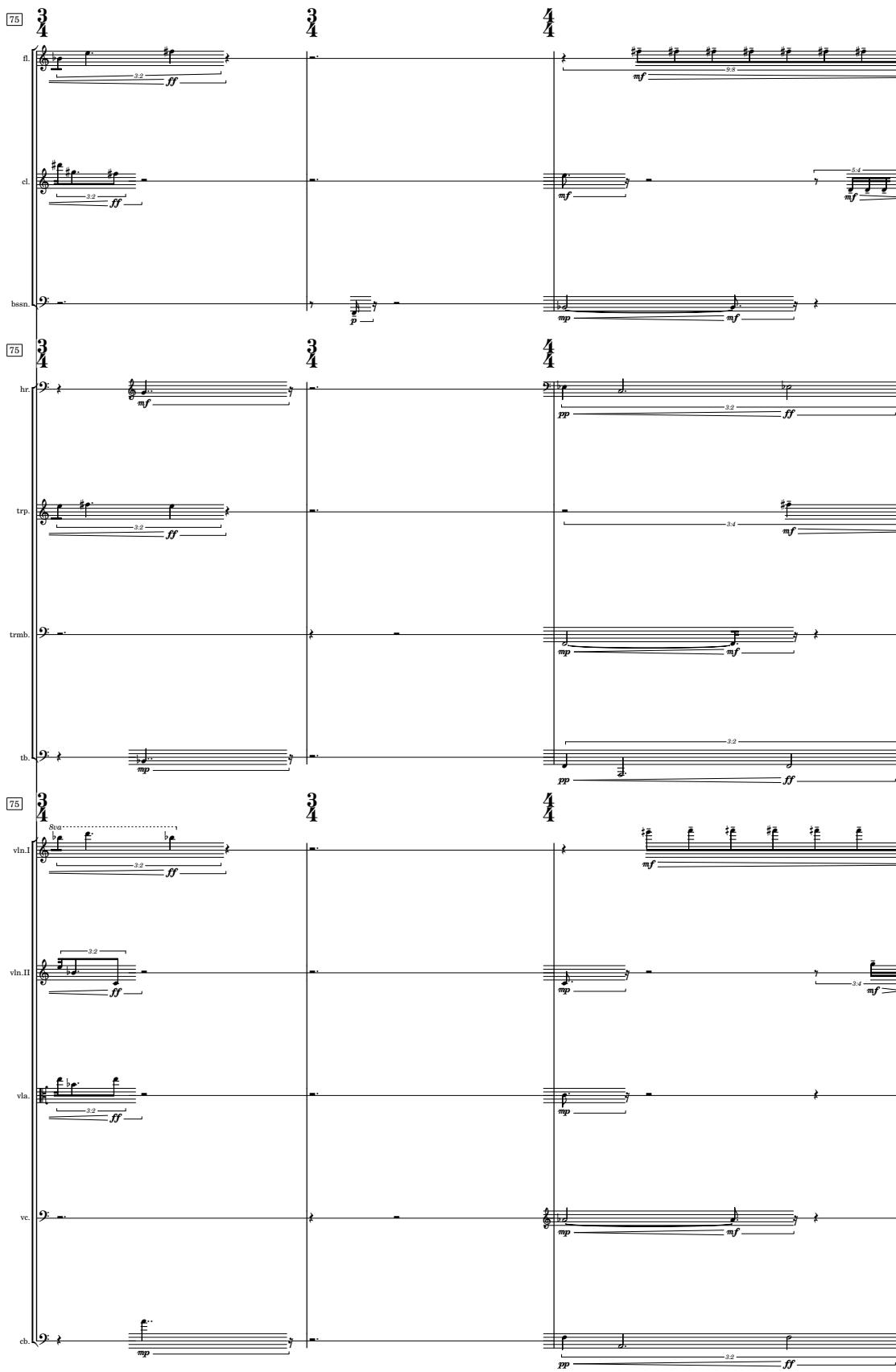
80 

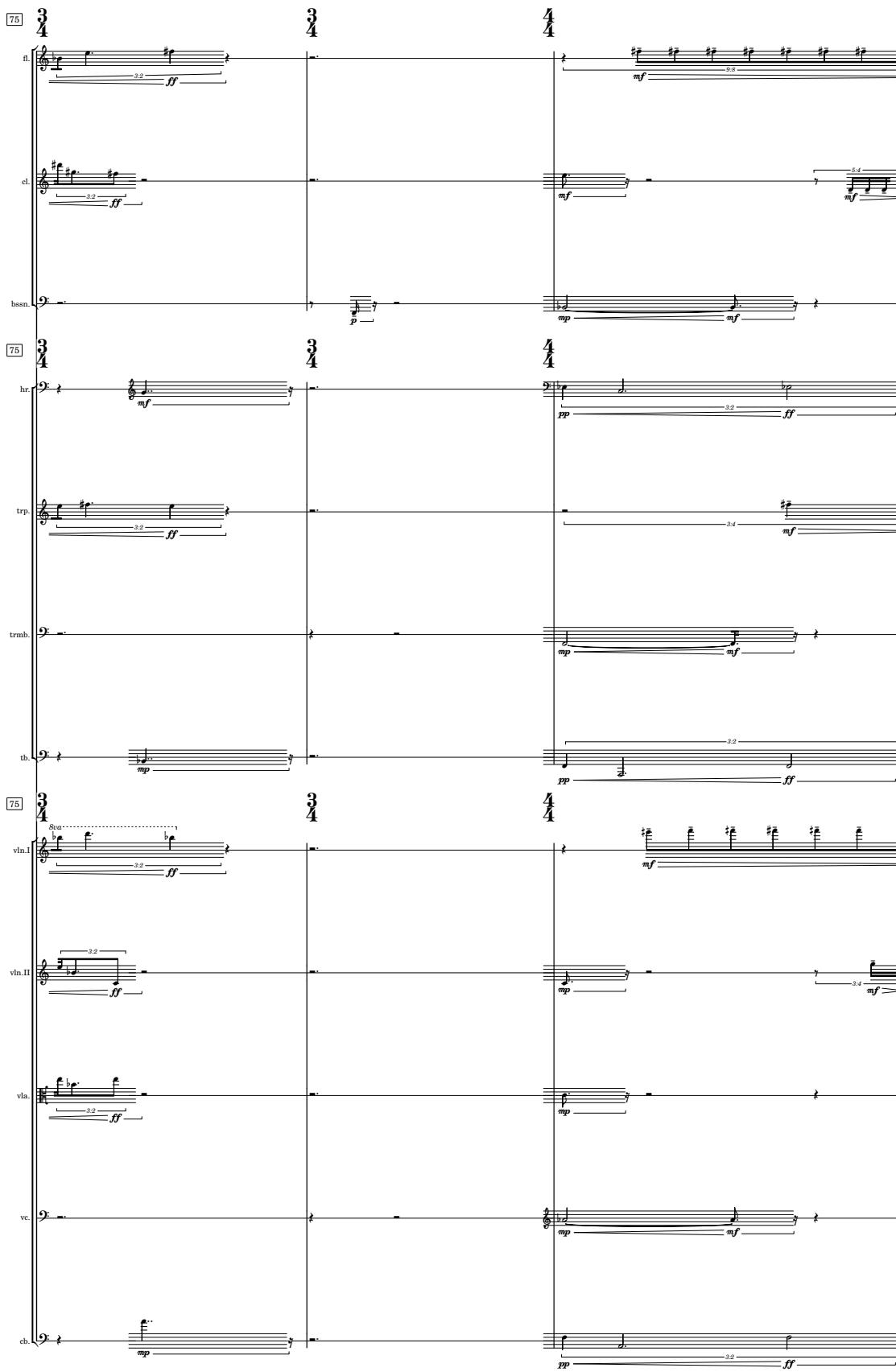
81 

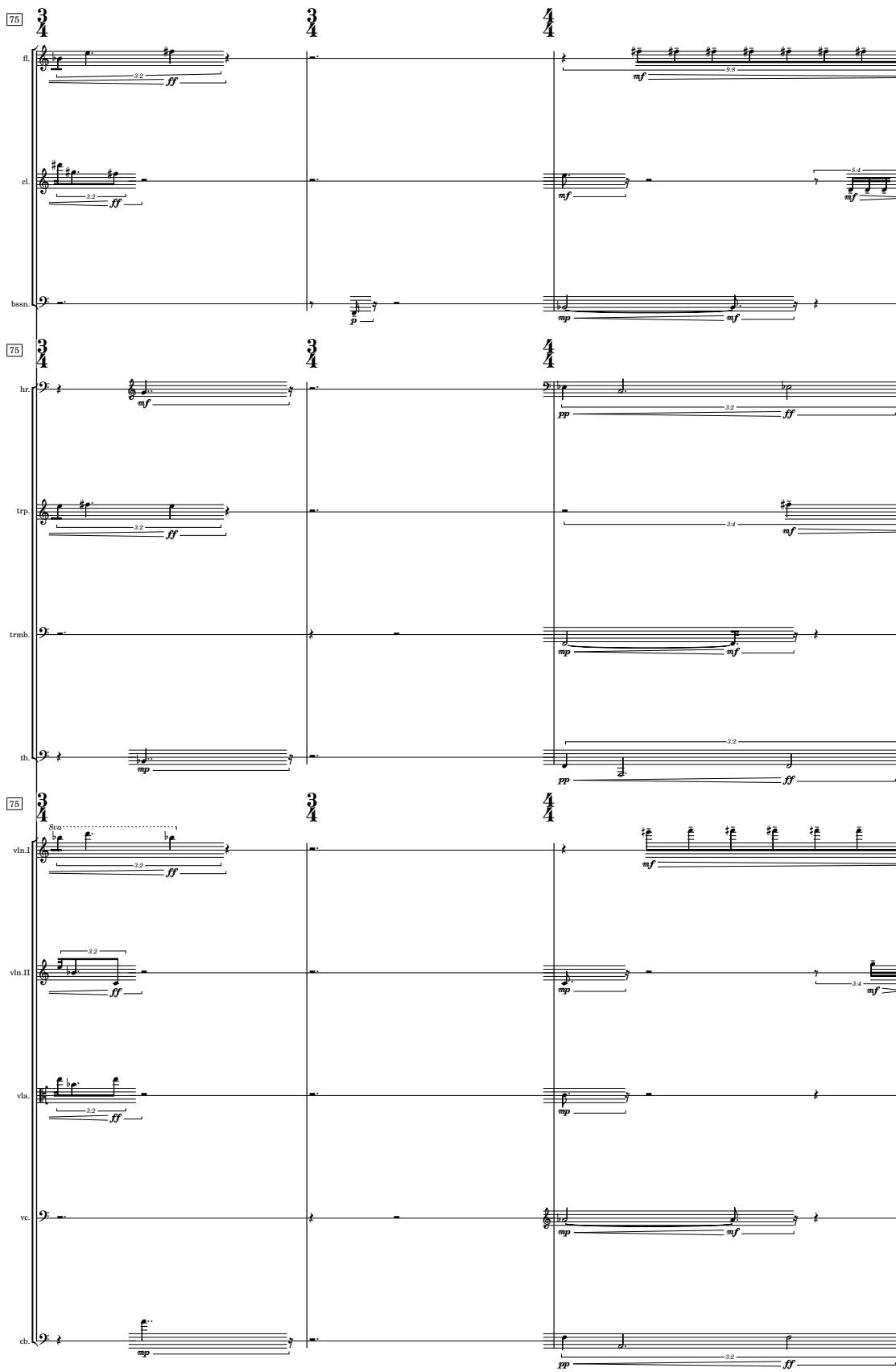
82 

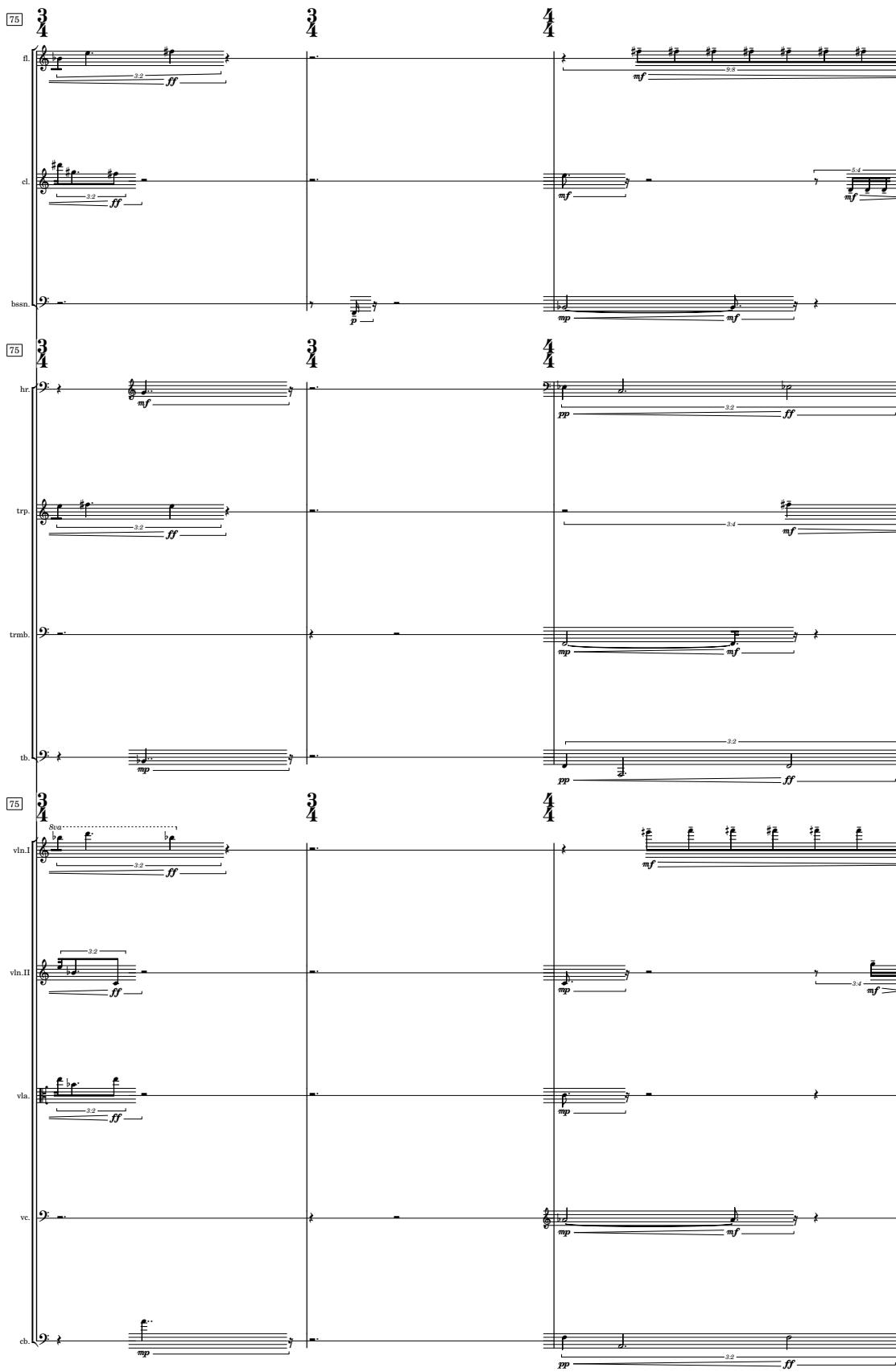
83 

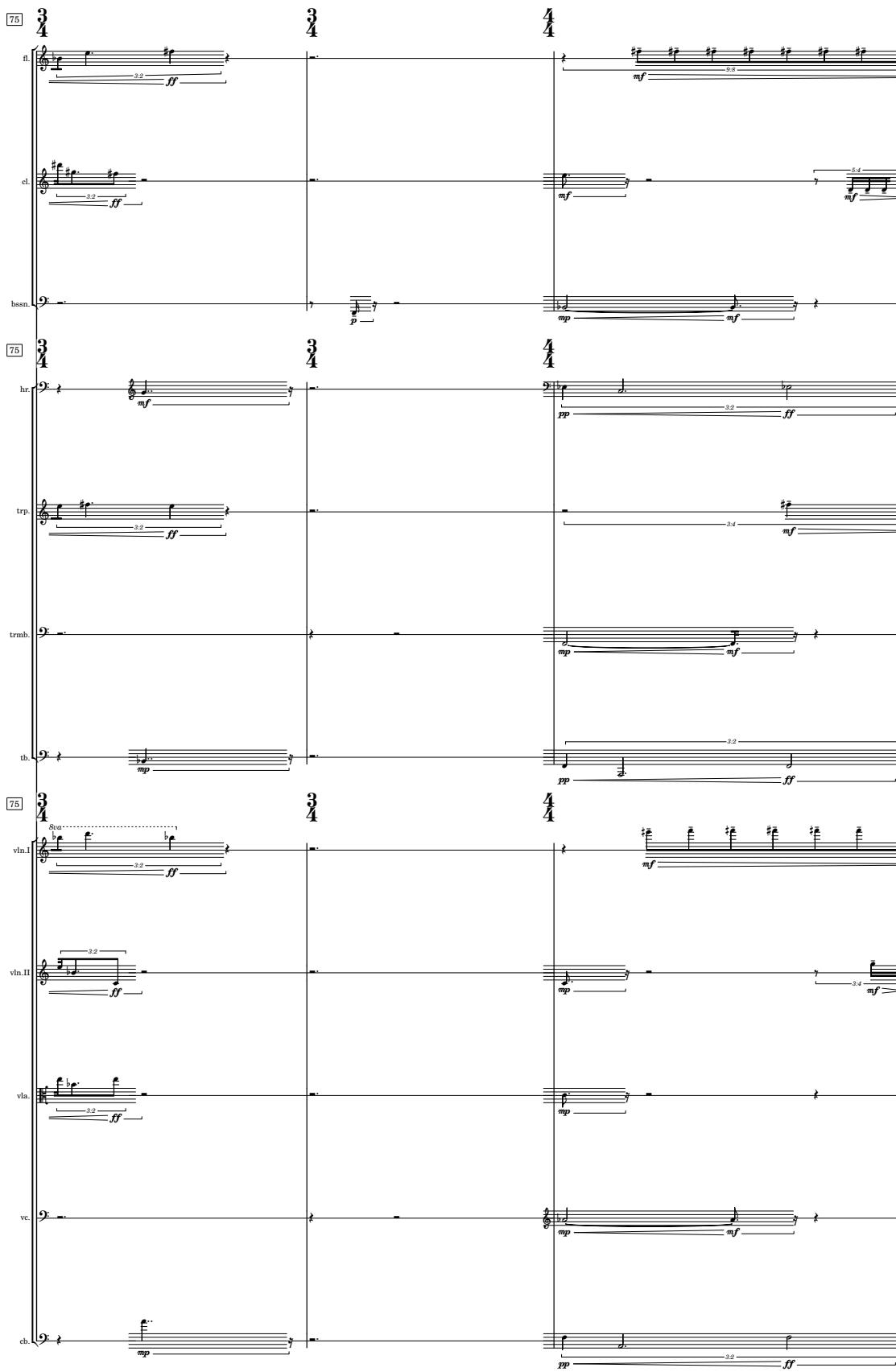
84 

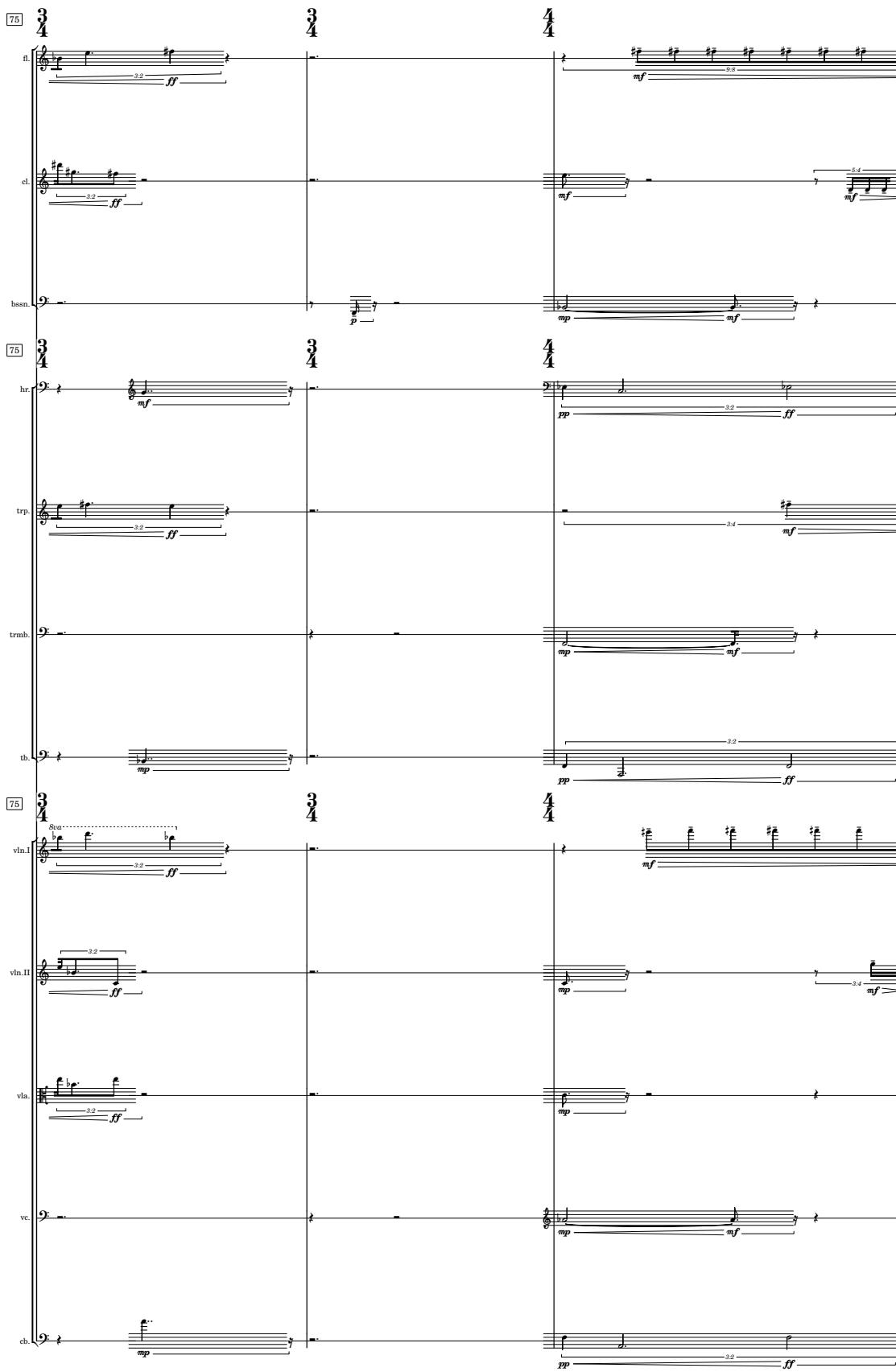
85 

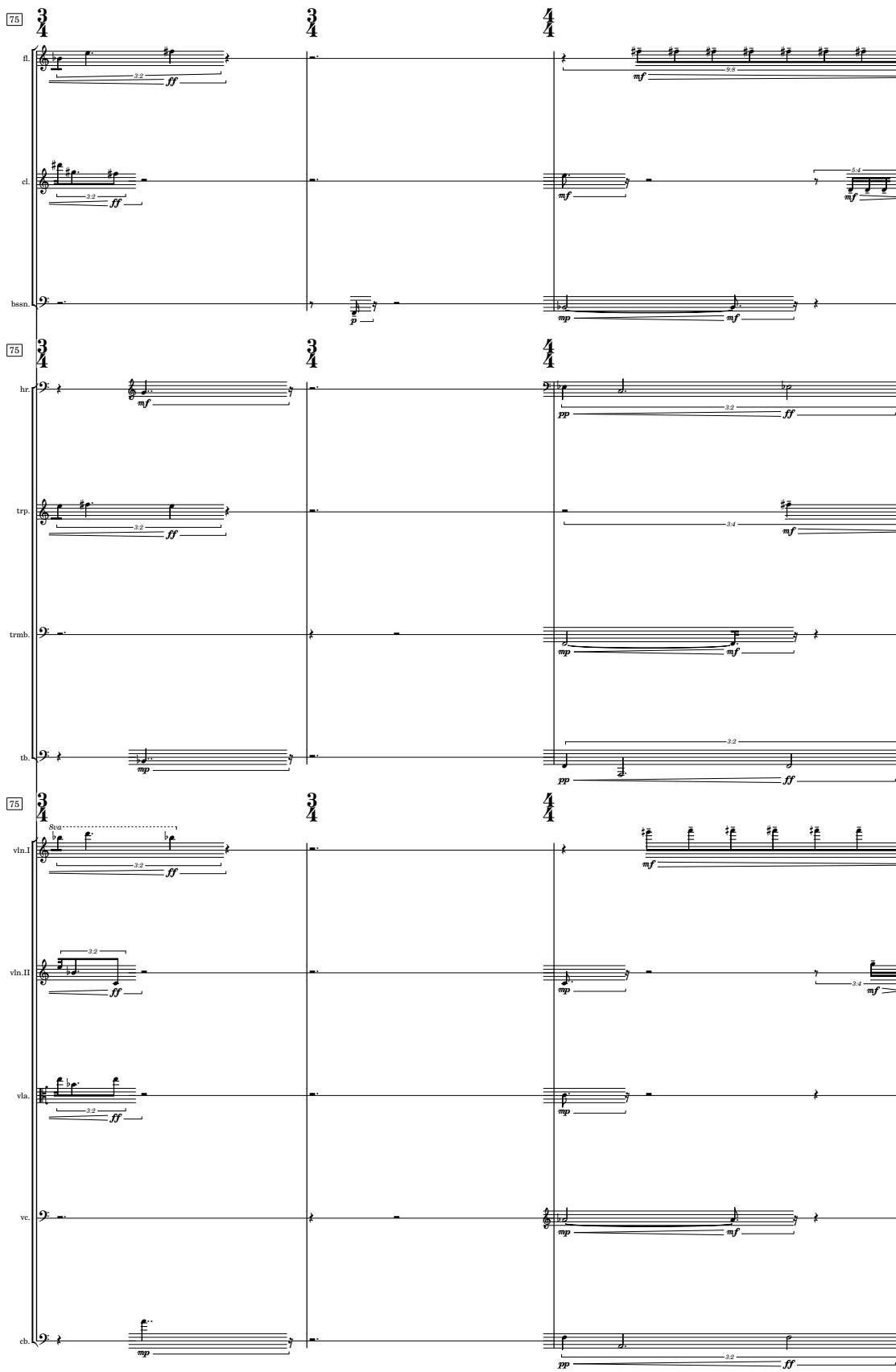
86 

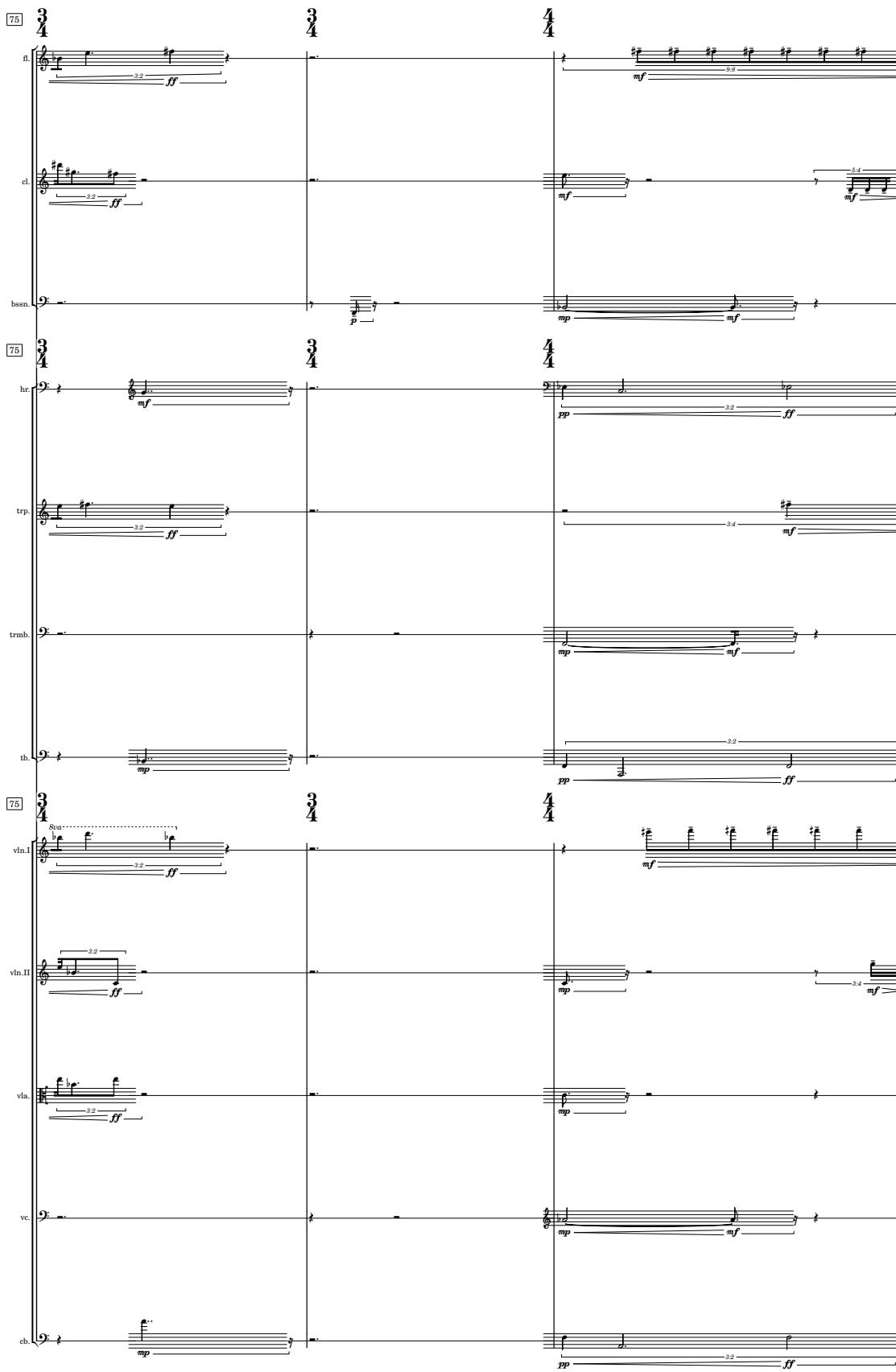
87 

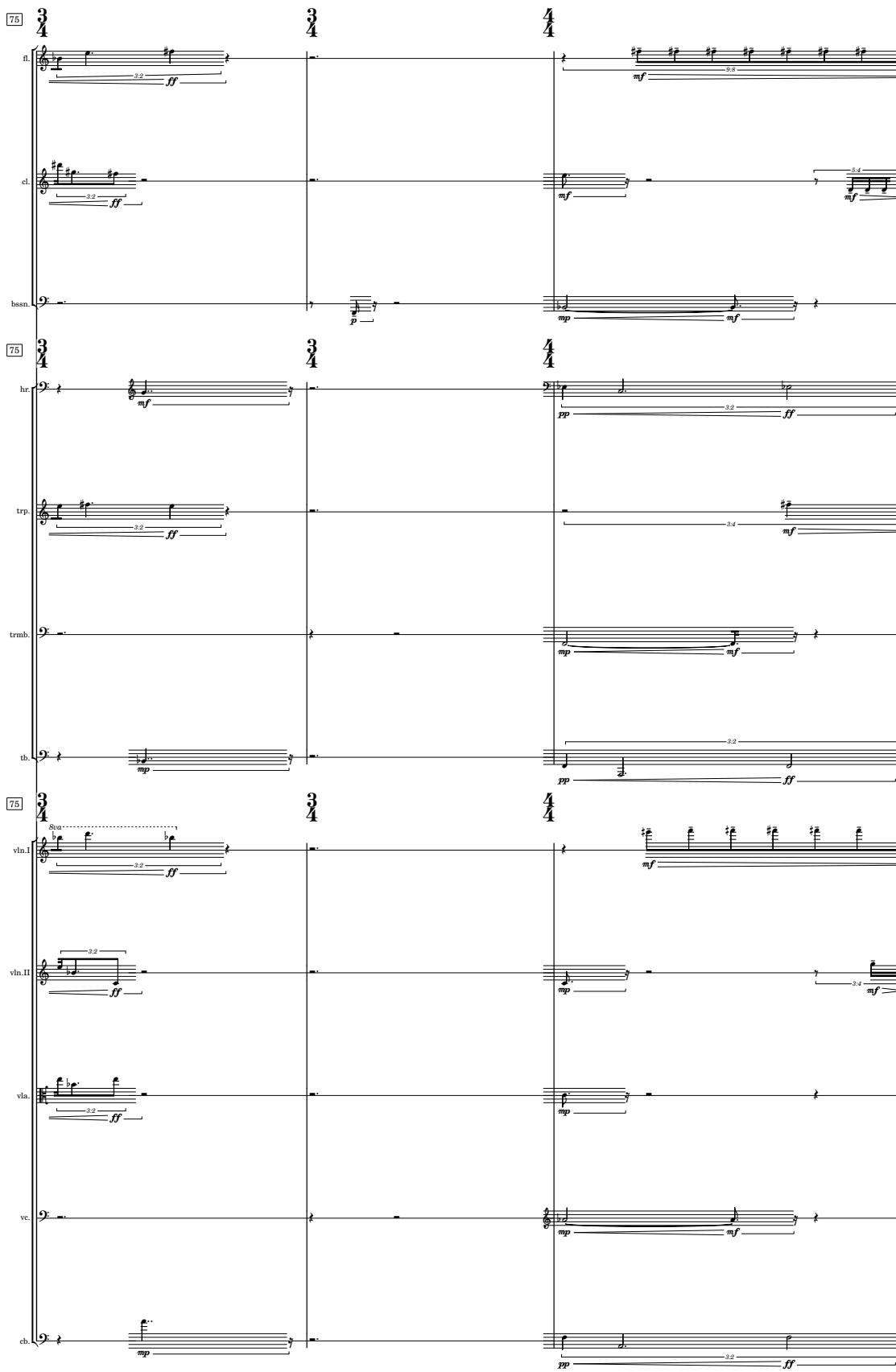
88 

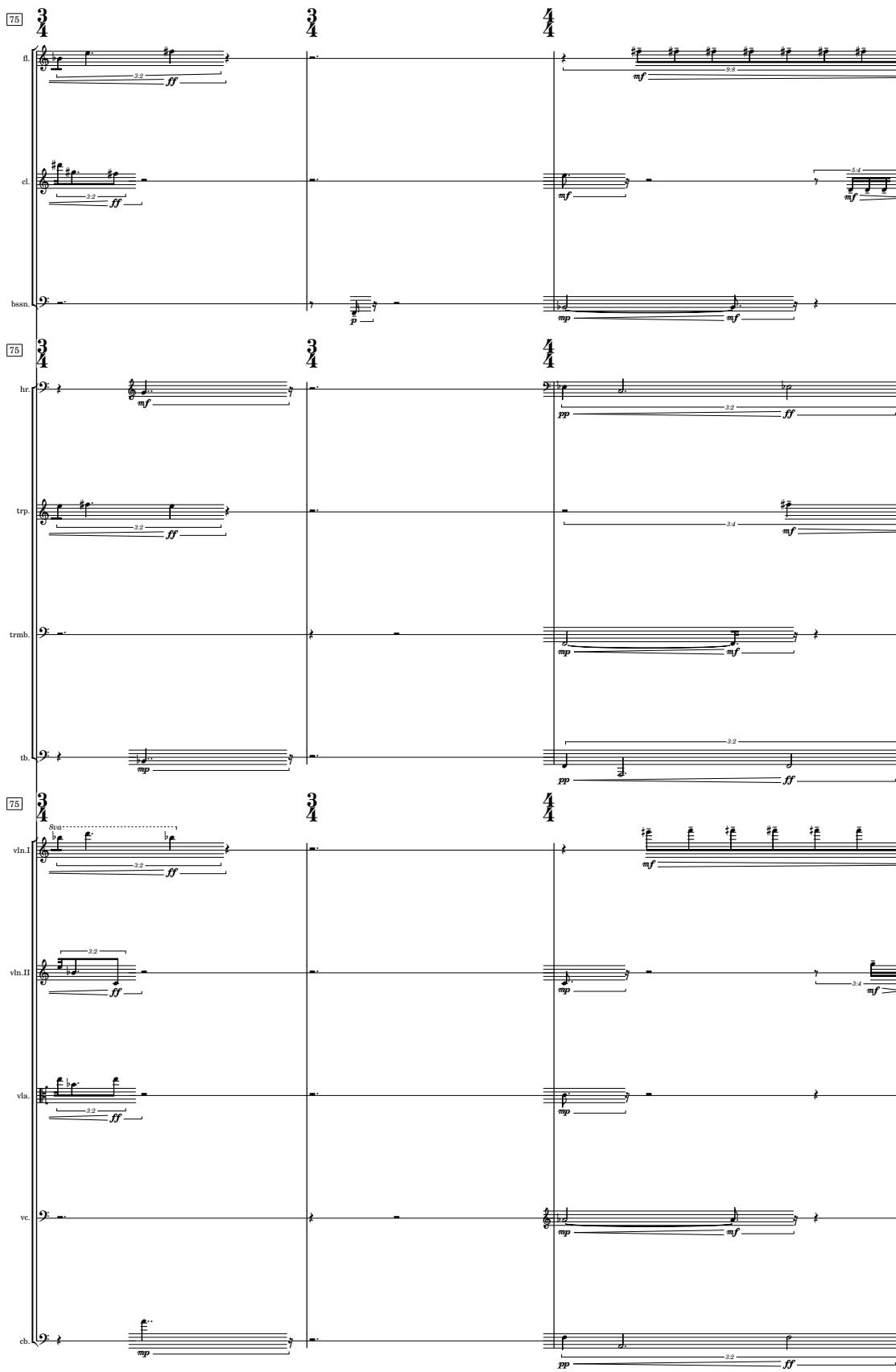
89 

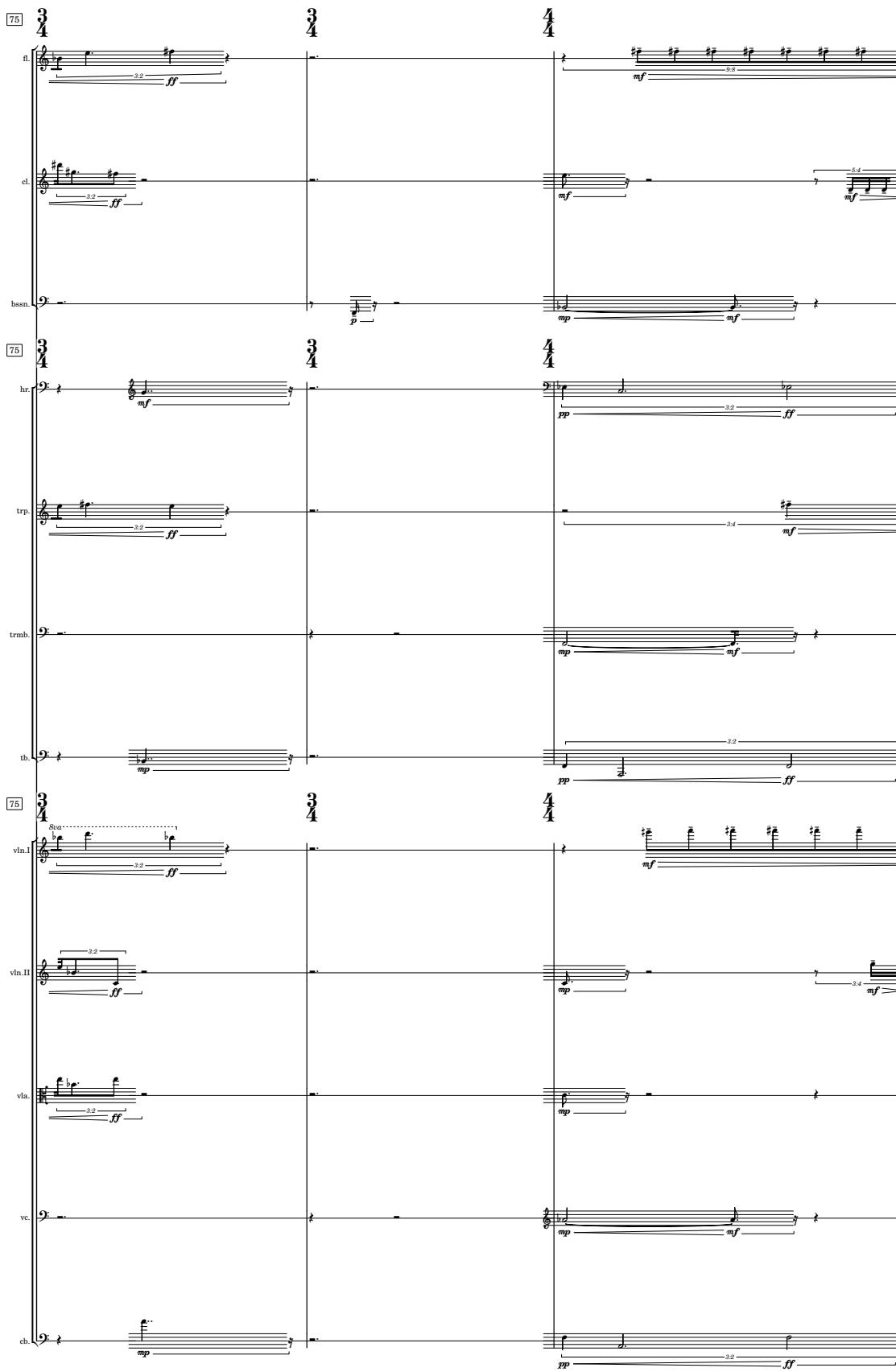
90 

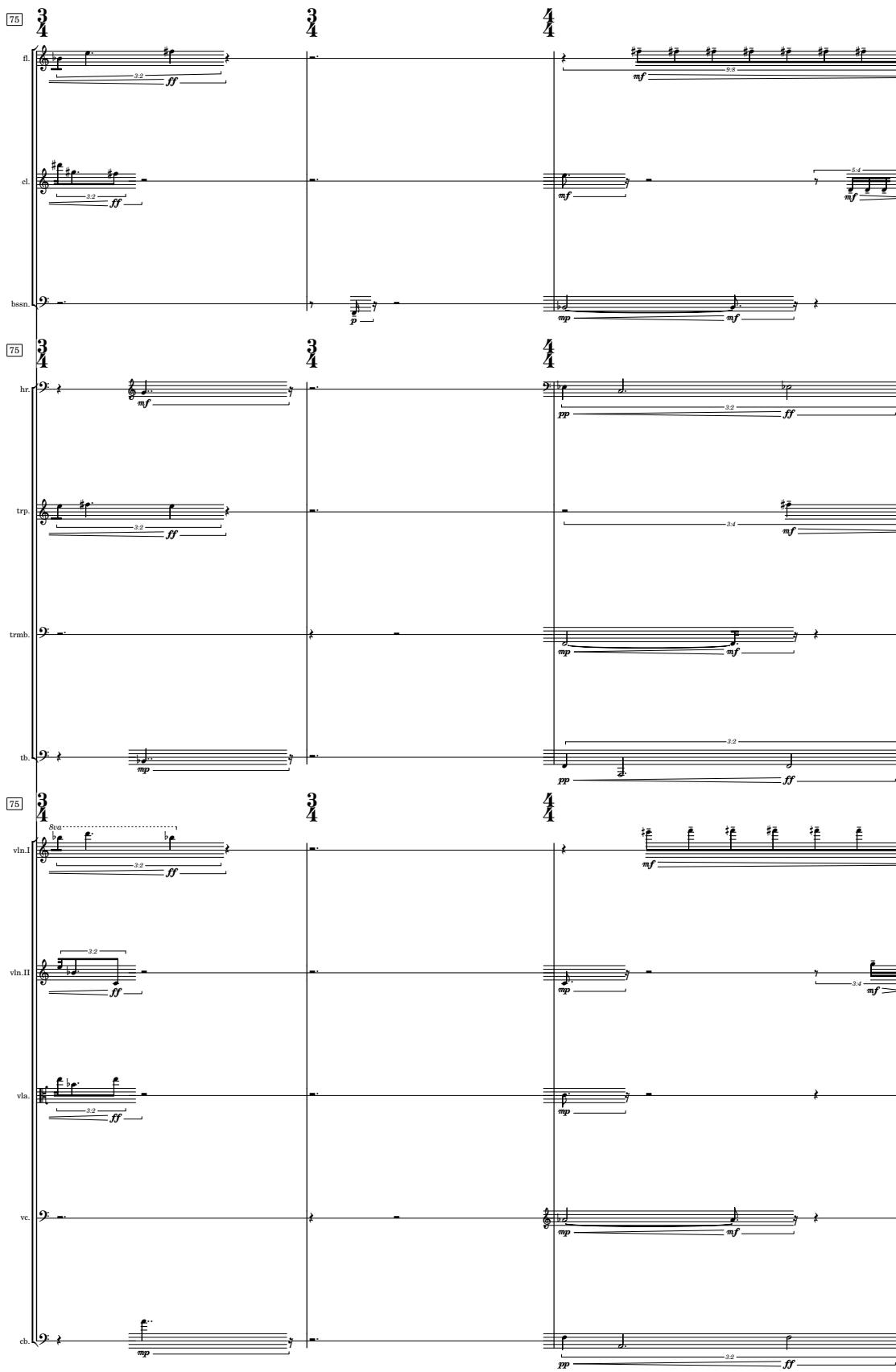
91 

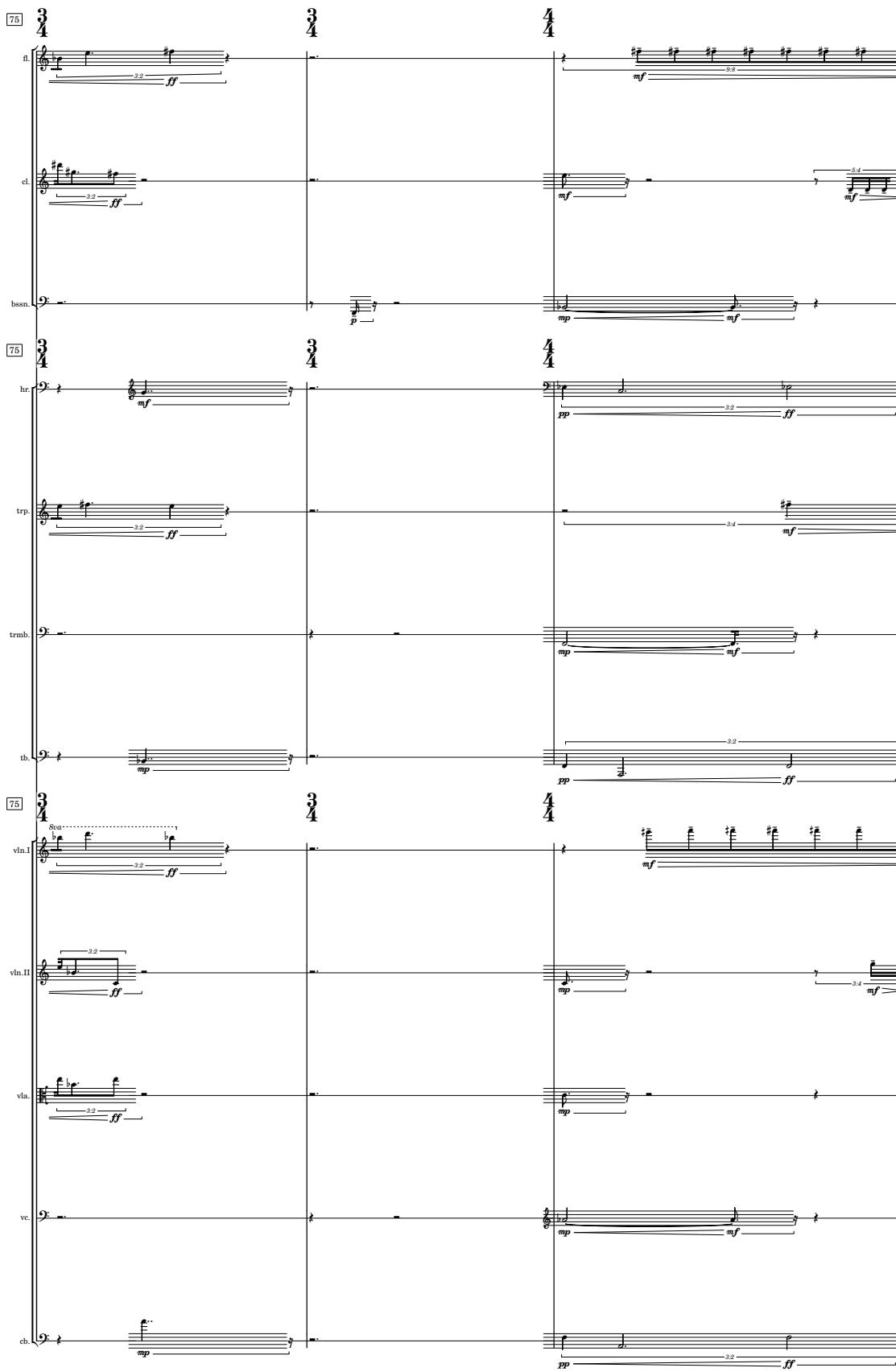
92 

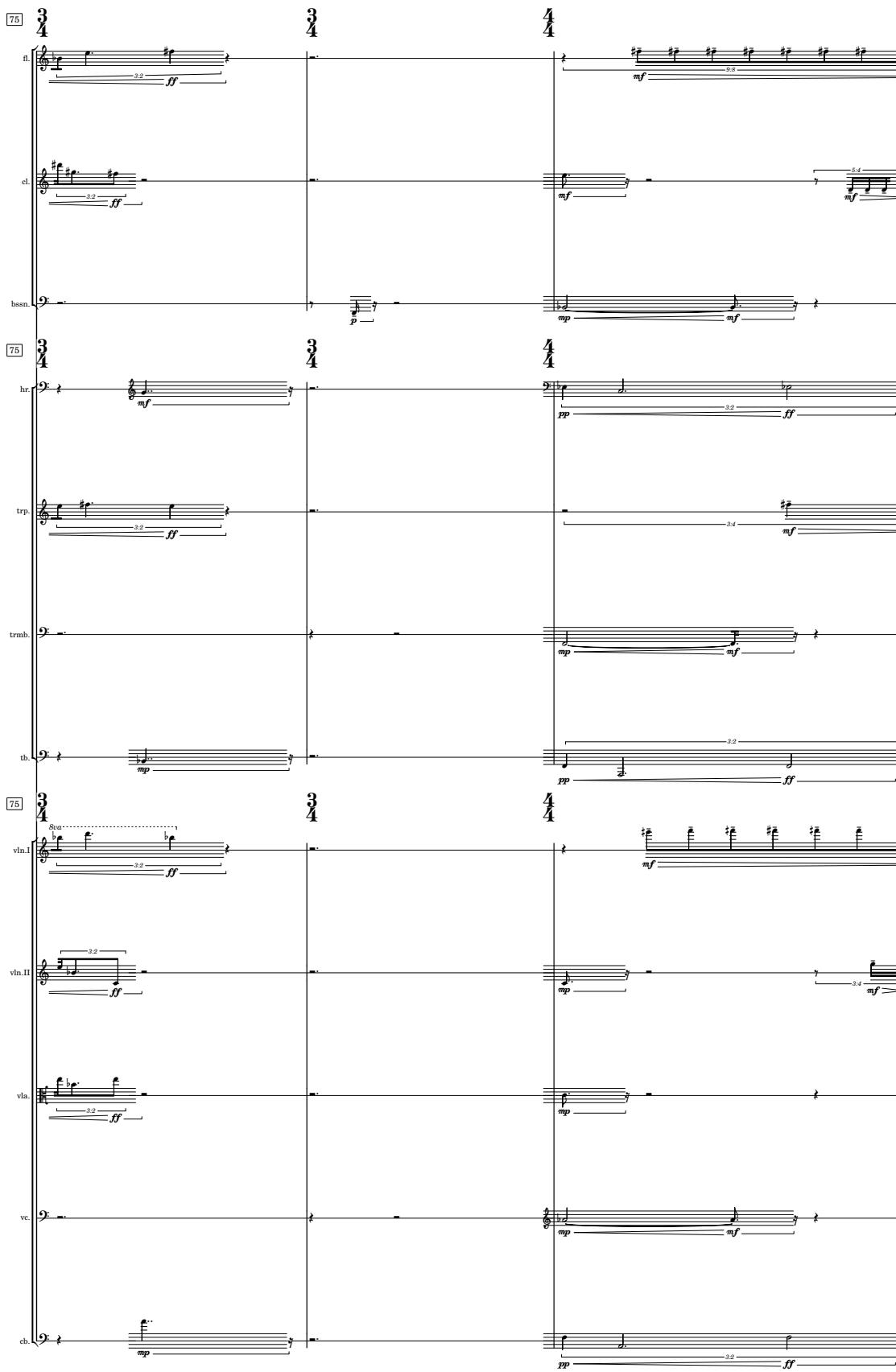
93 

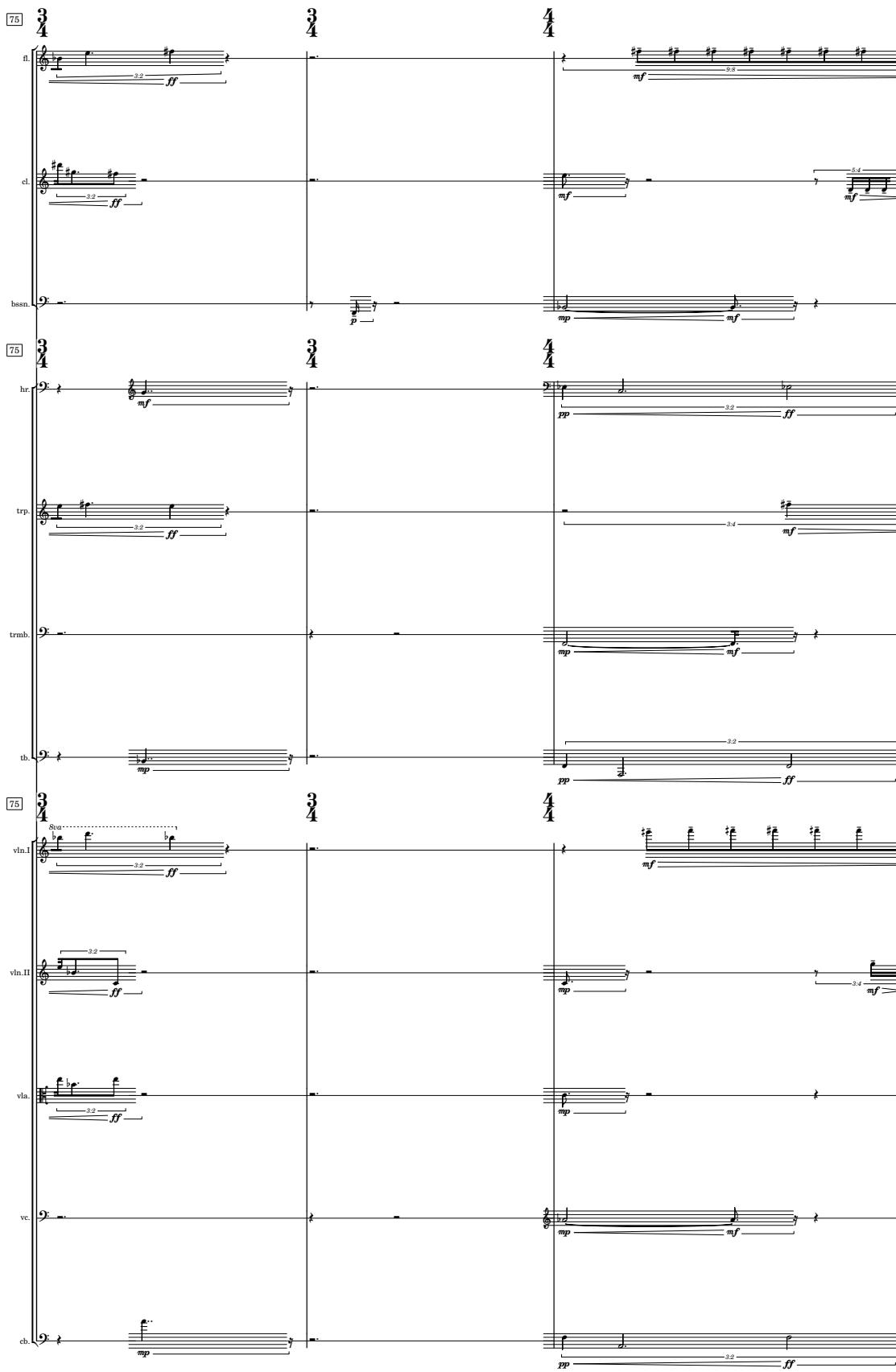
94 

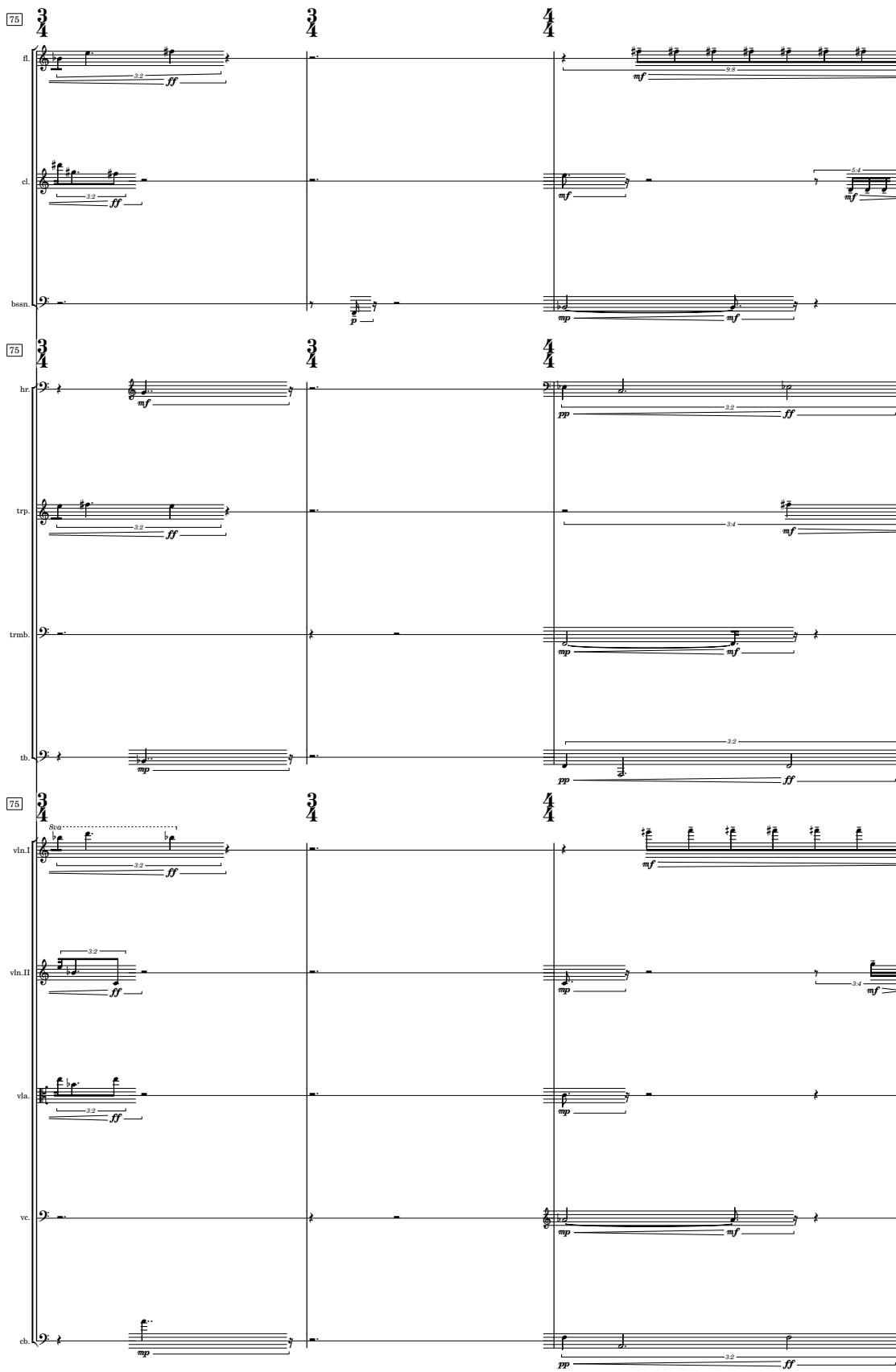
95 

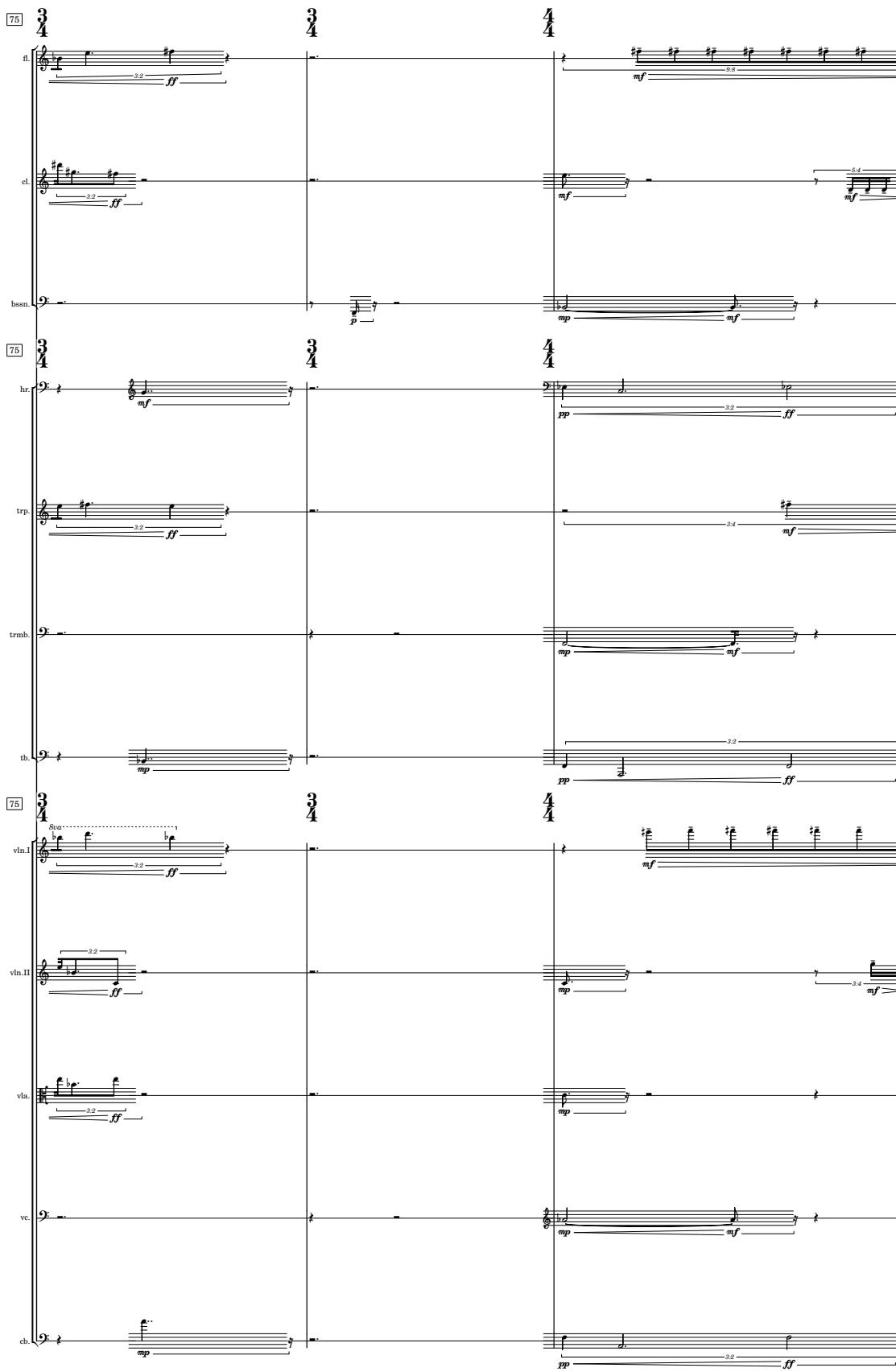
96 

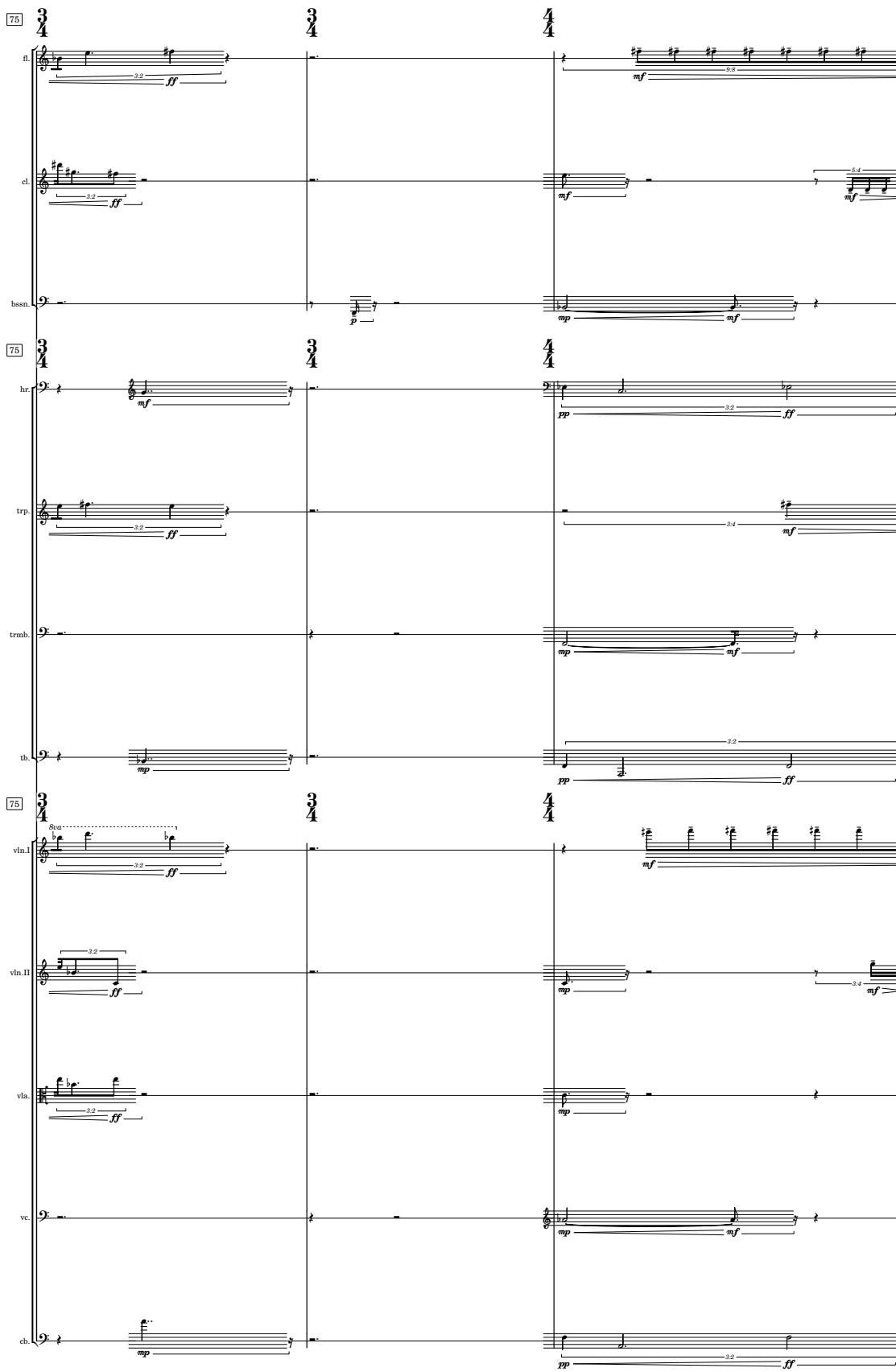
97 

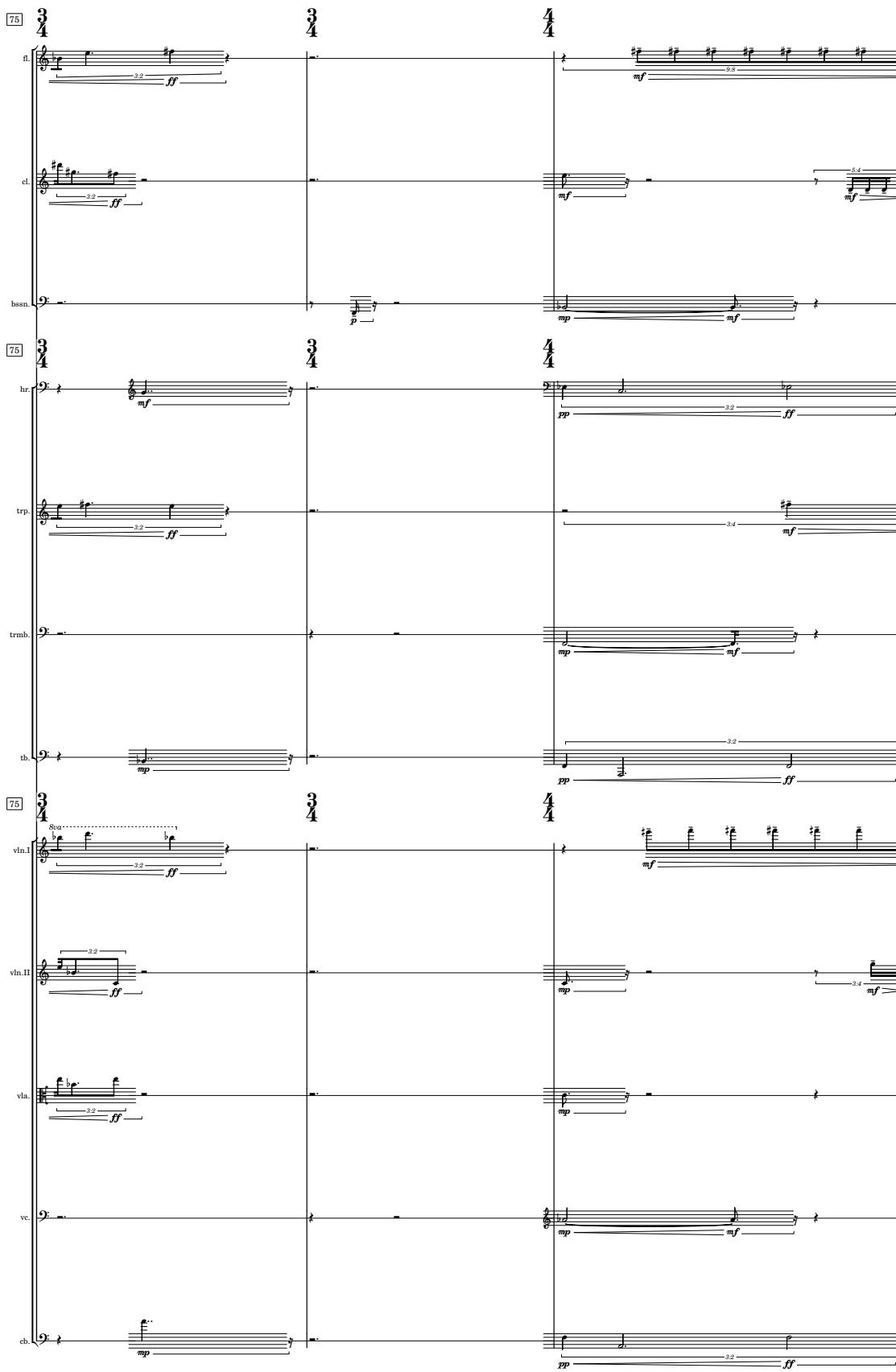
98 

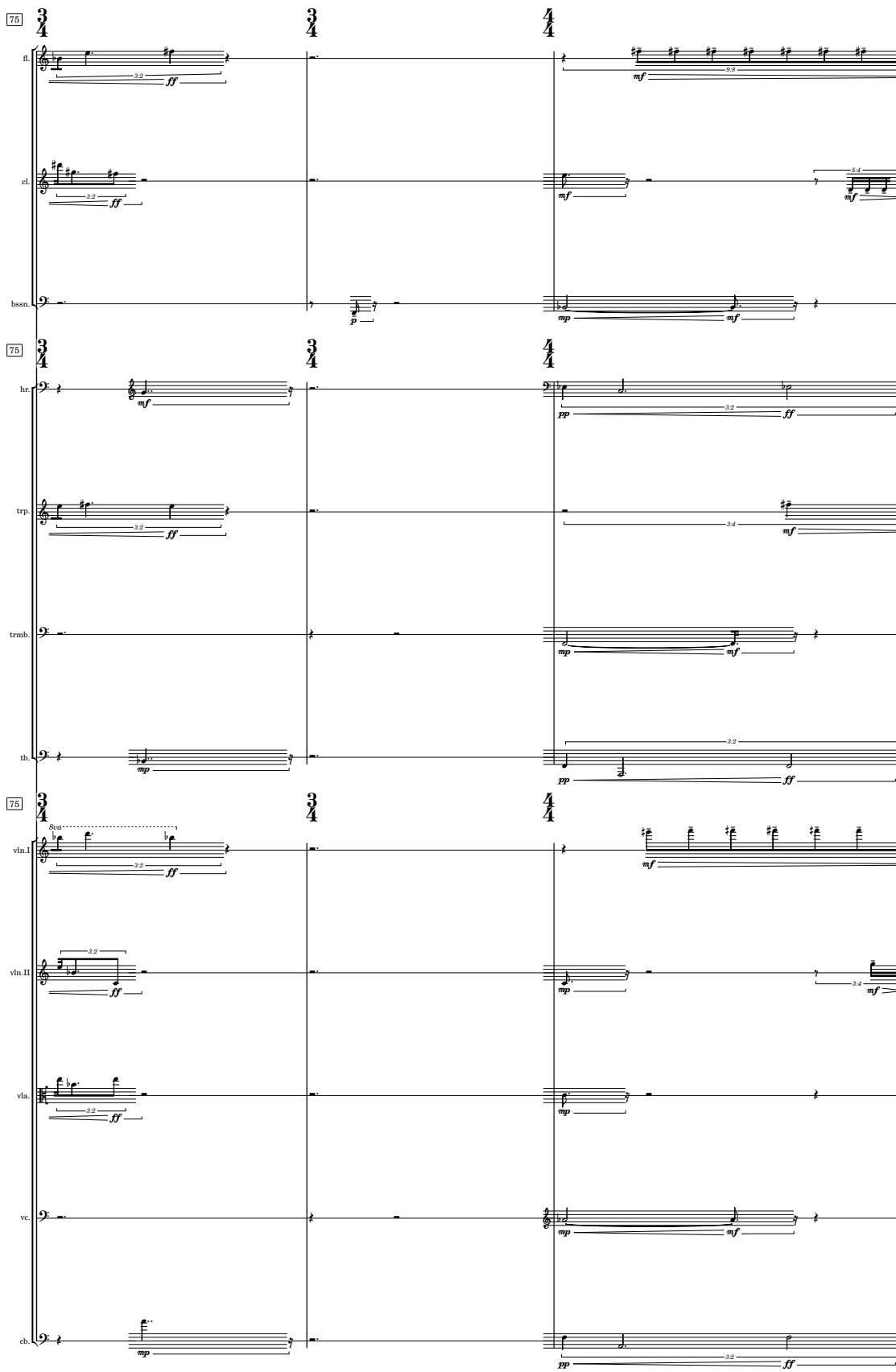
99 

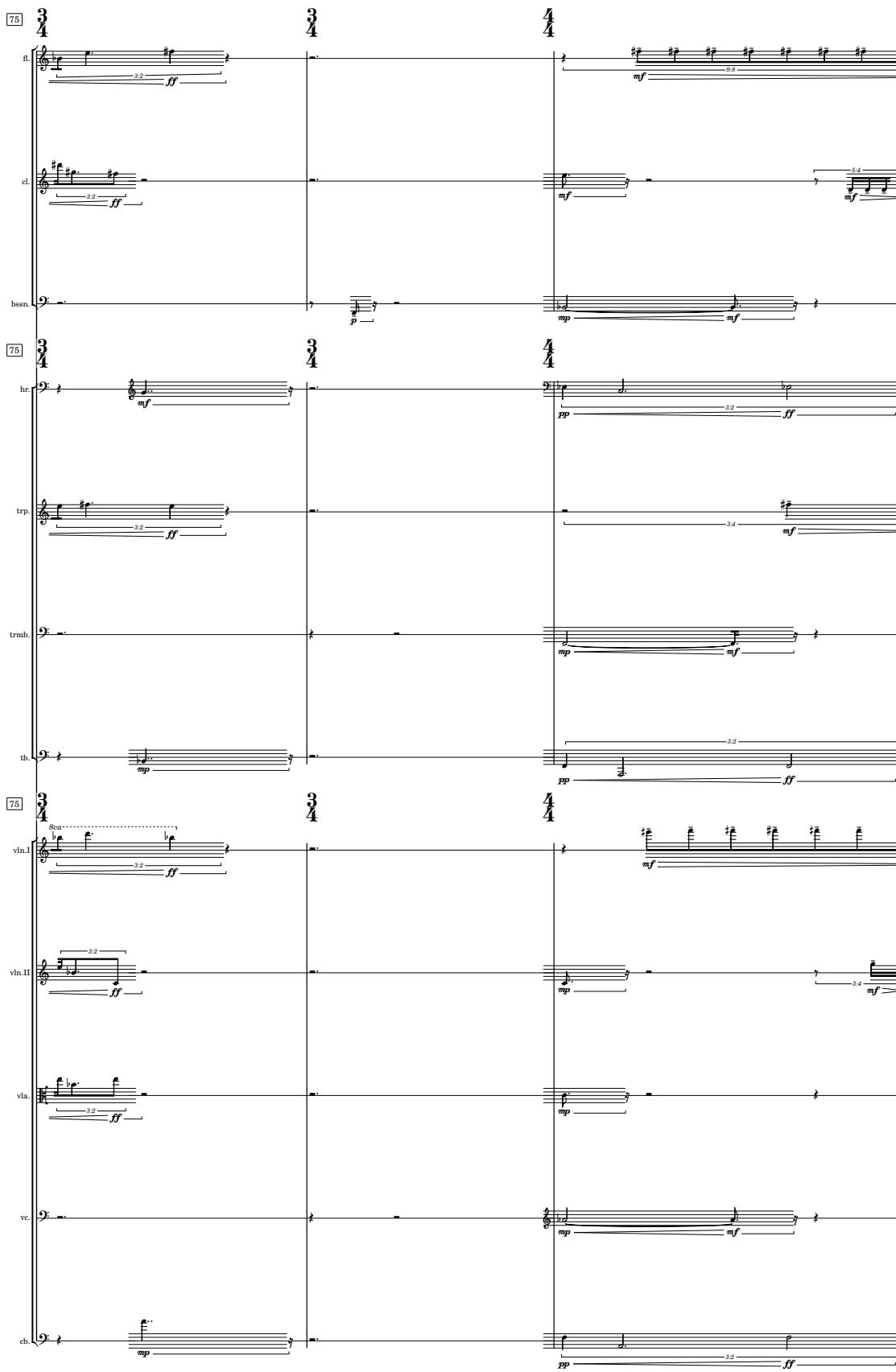
100 

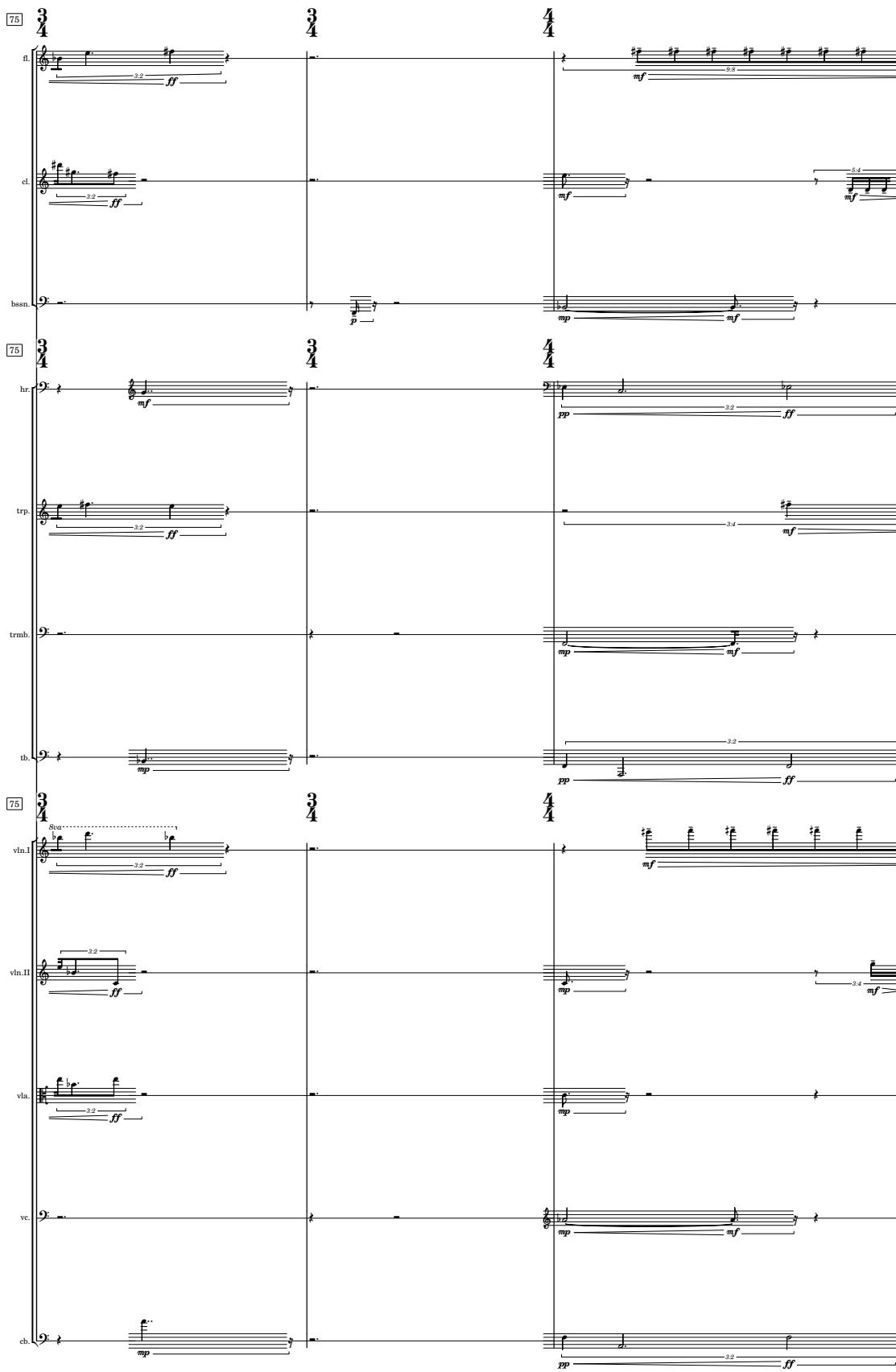
101 

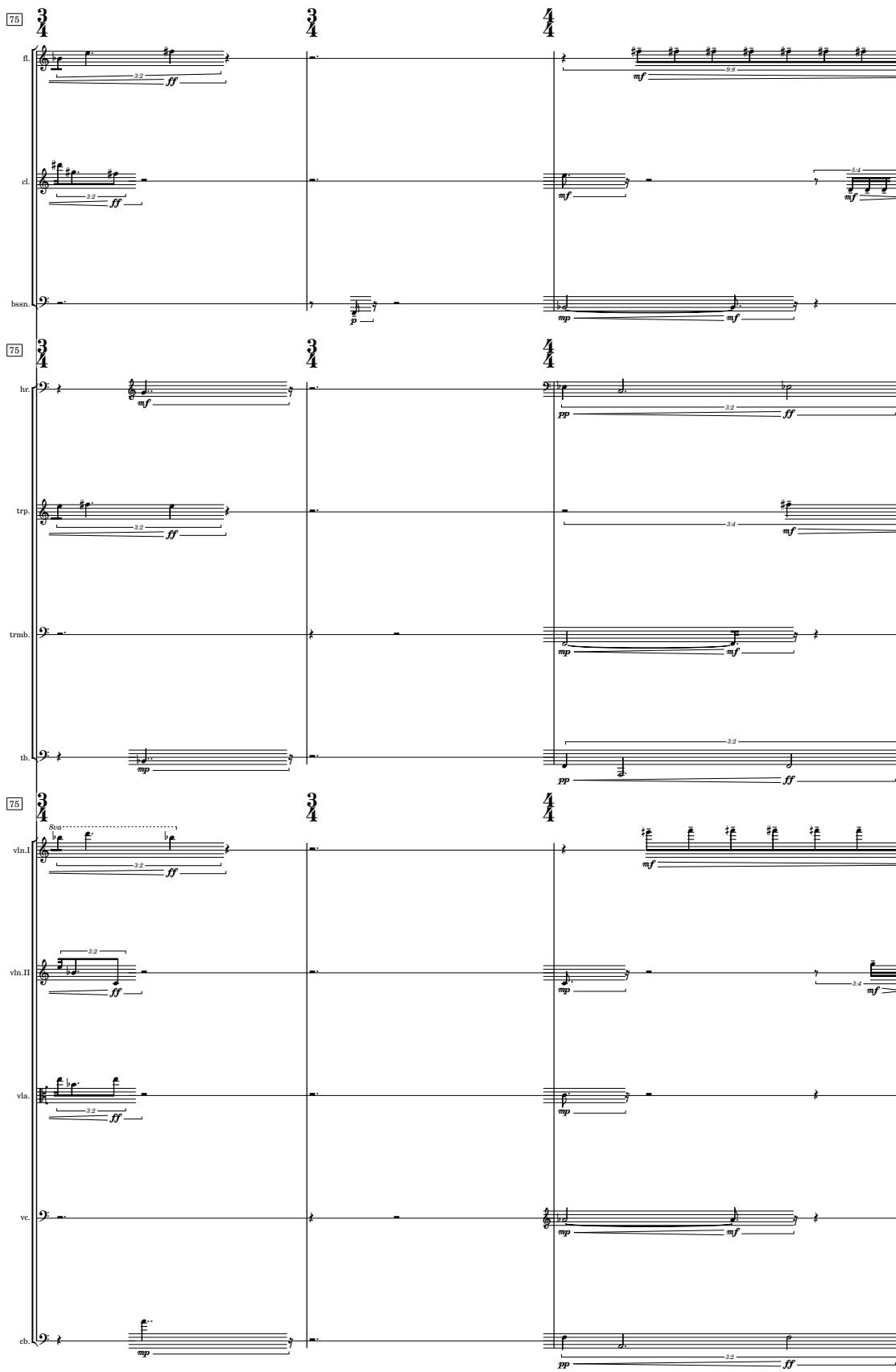
102 

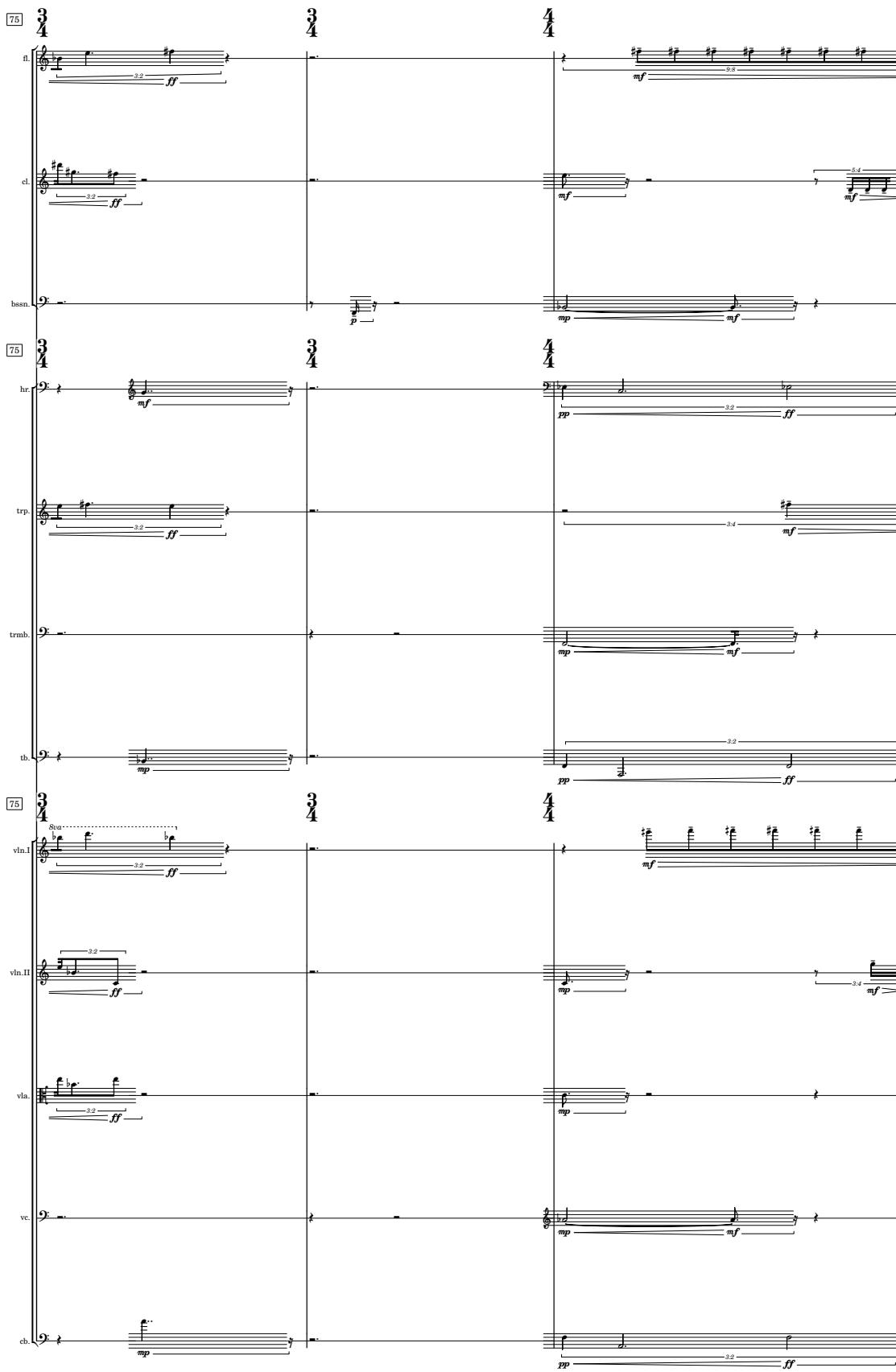
103 

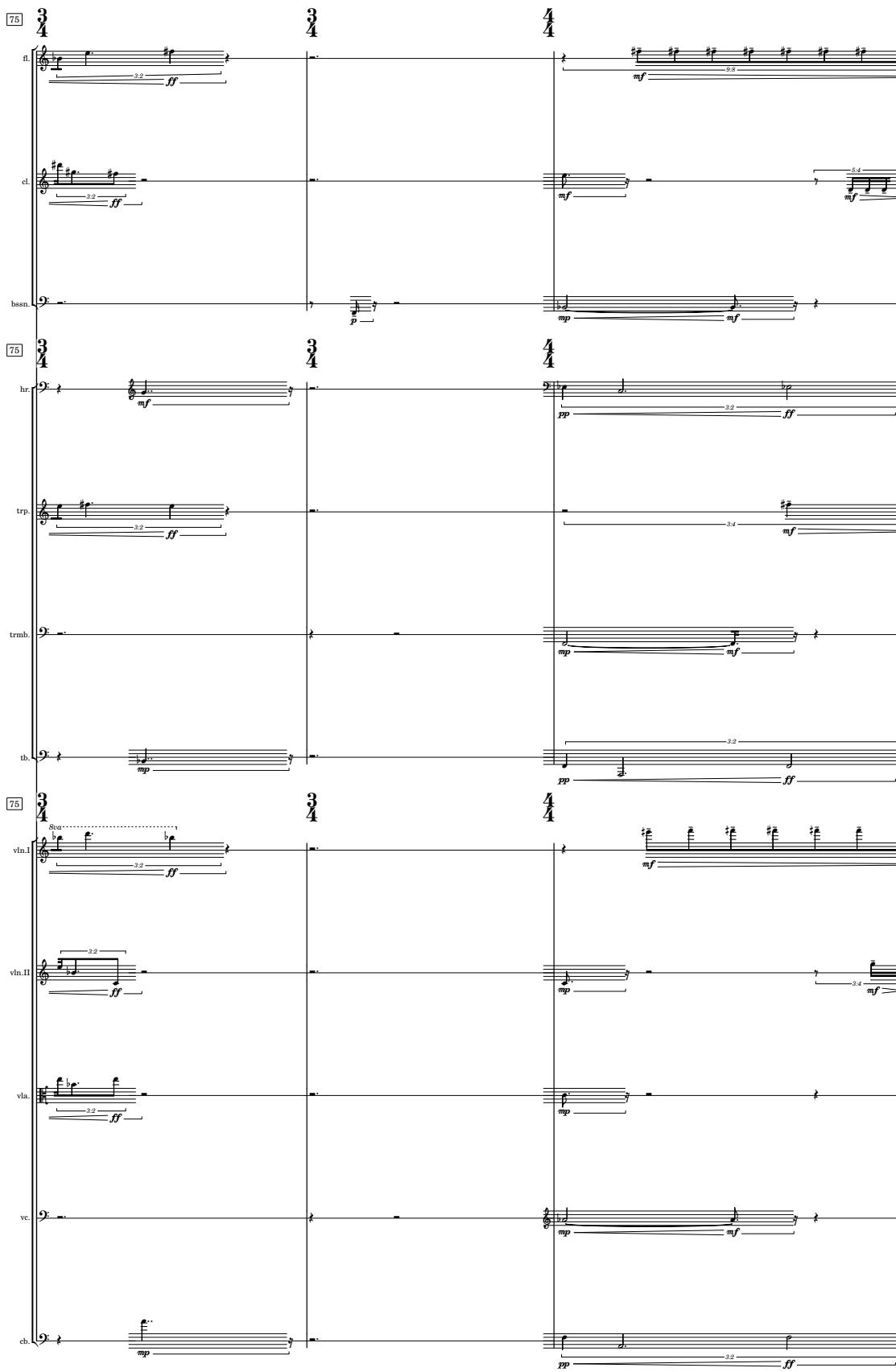
104 

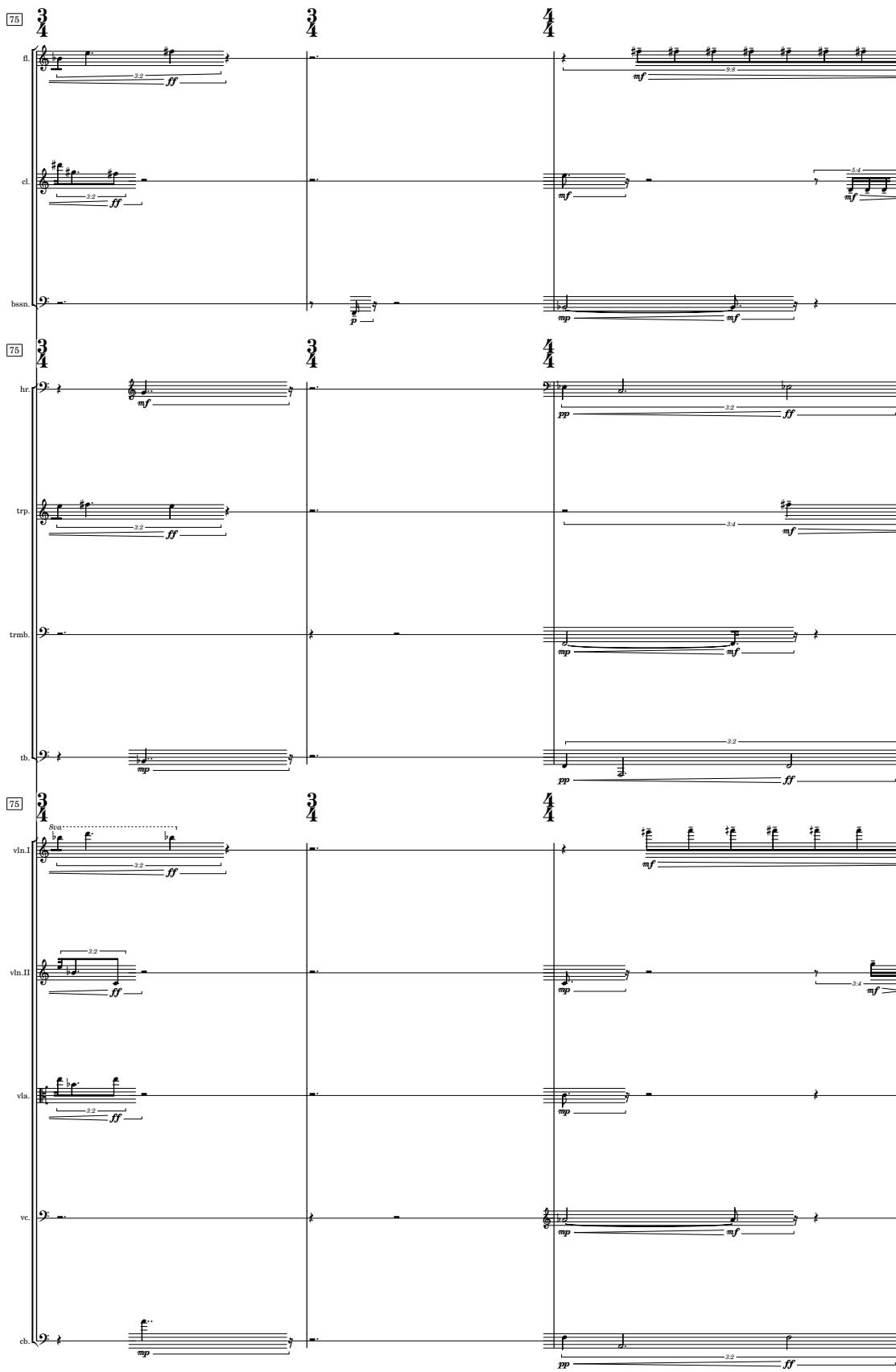
105 

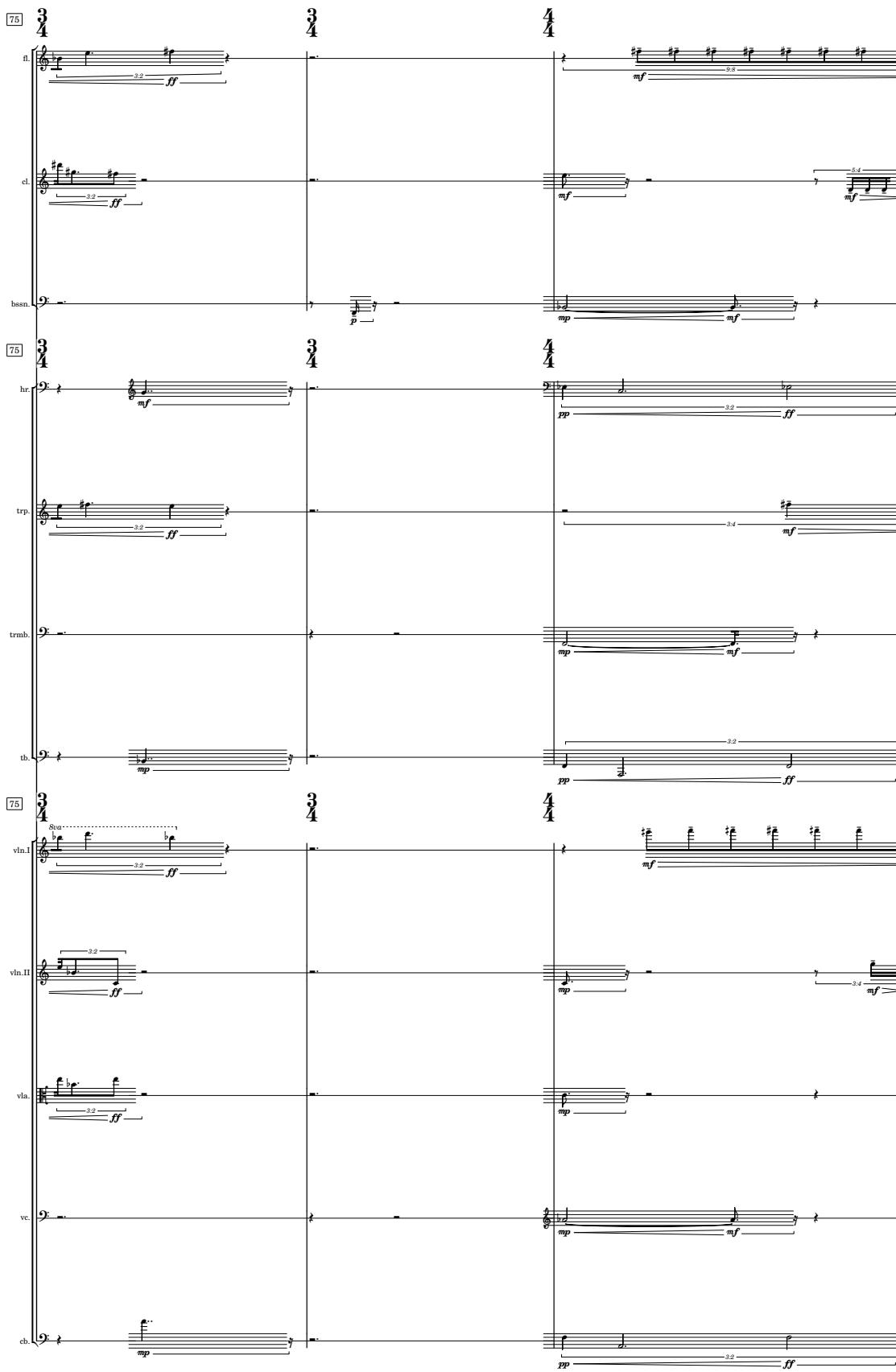
106 

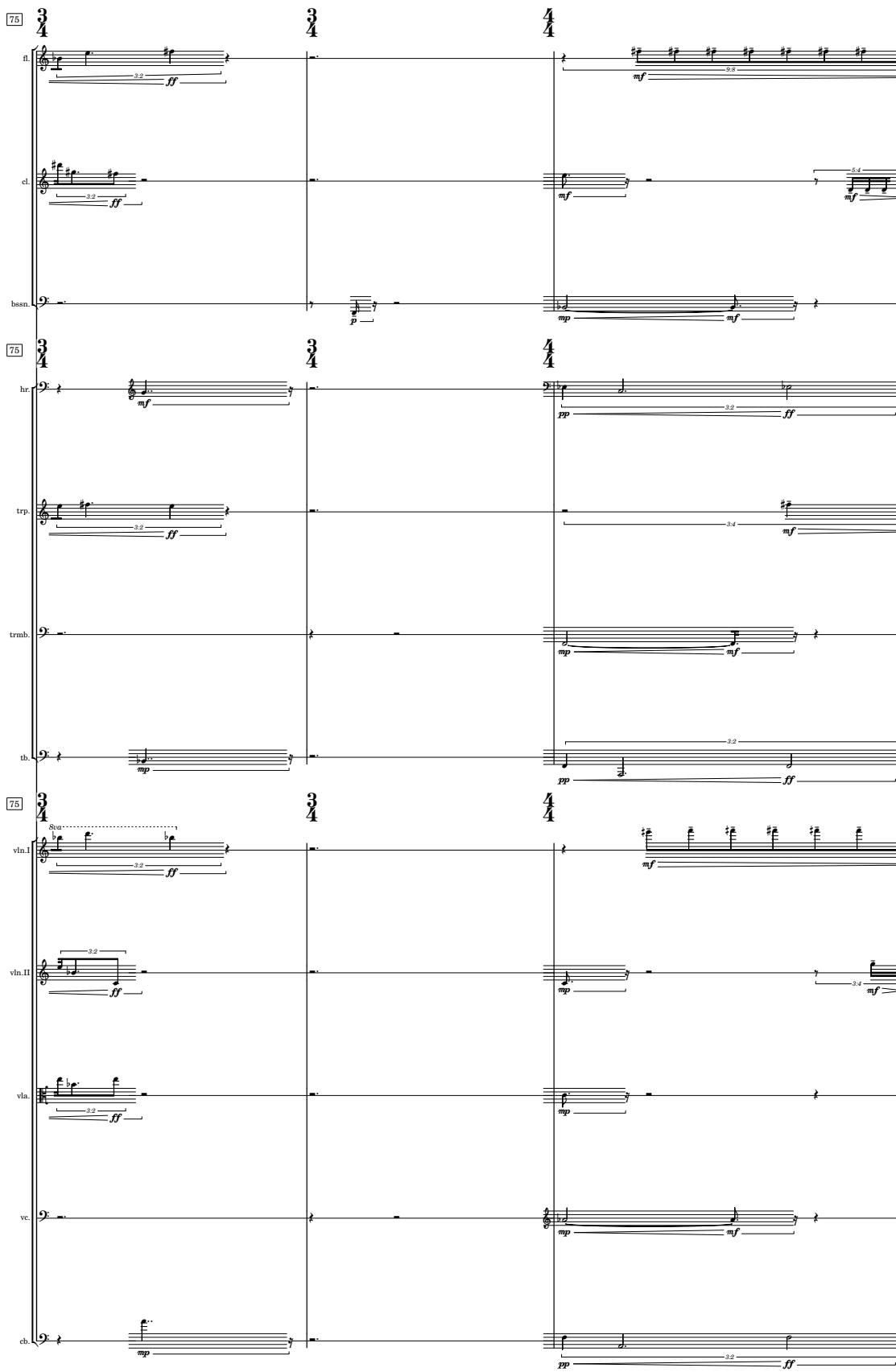
107 

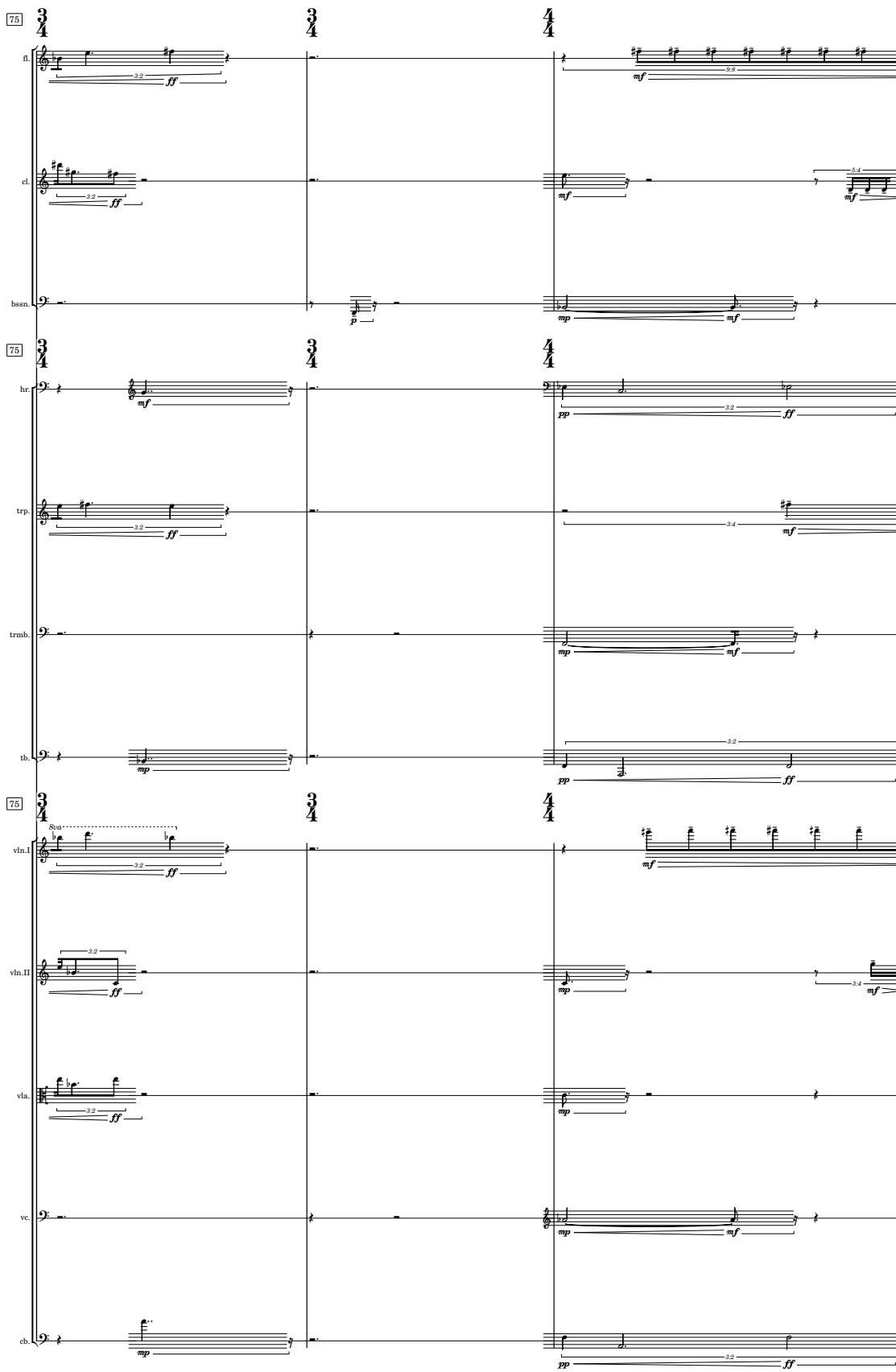
108 

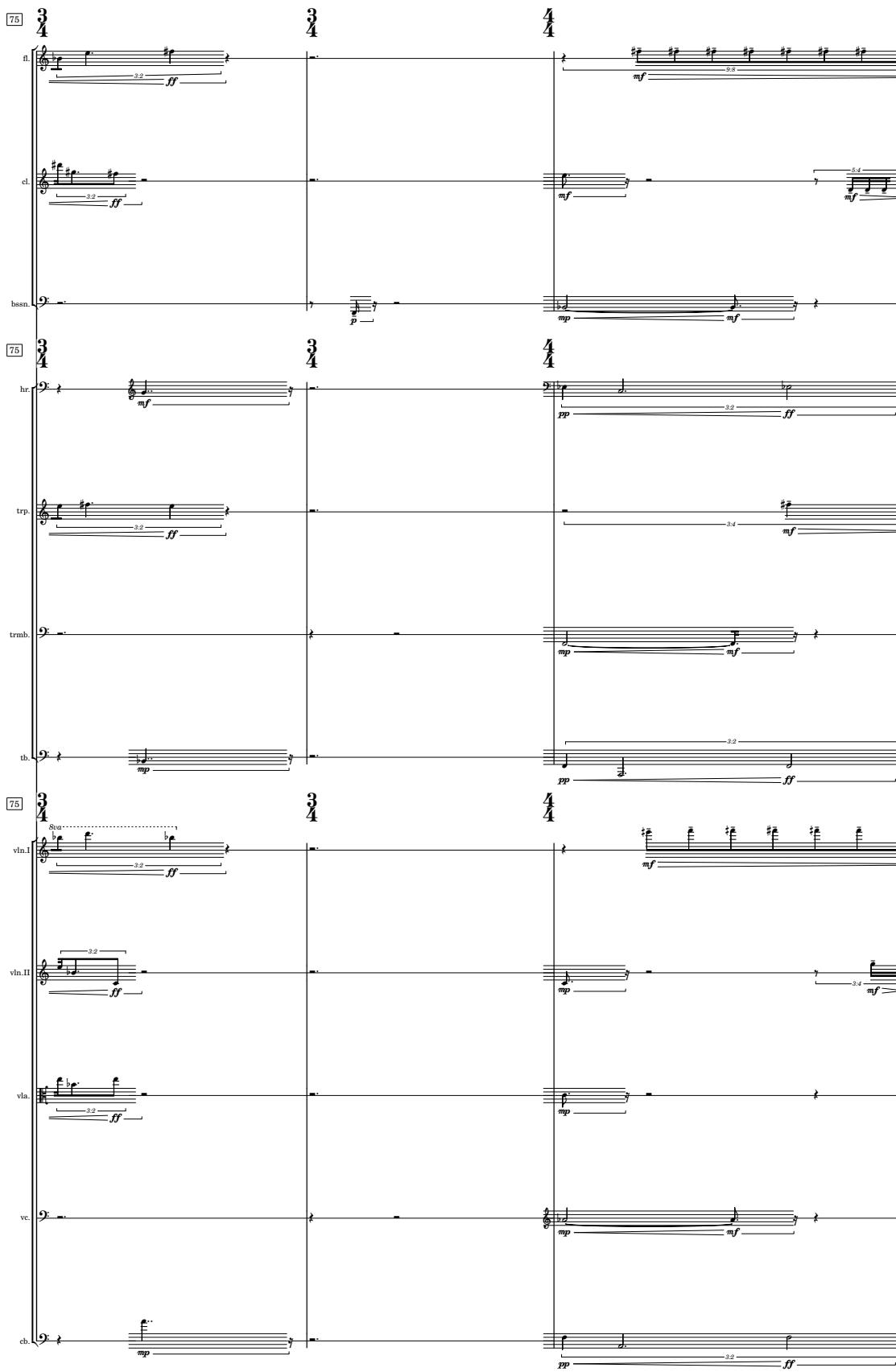
109 

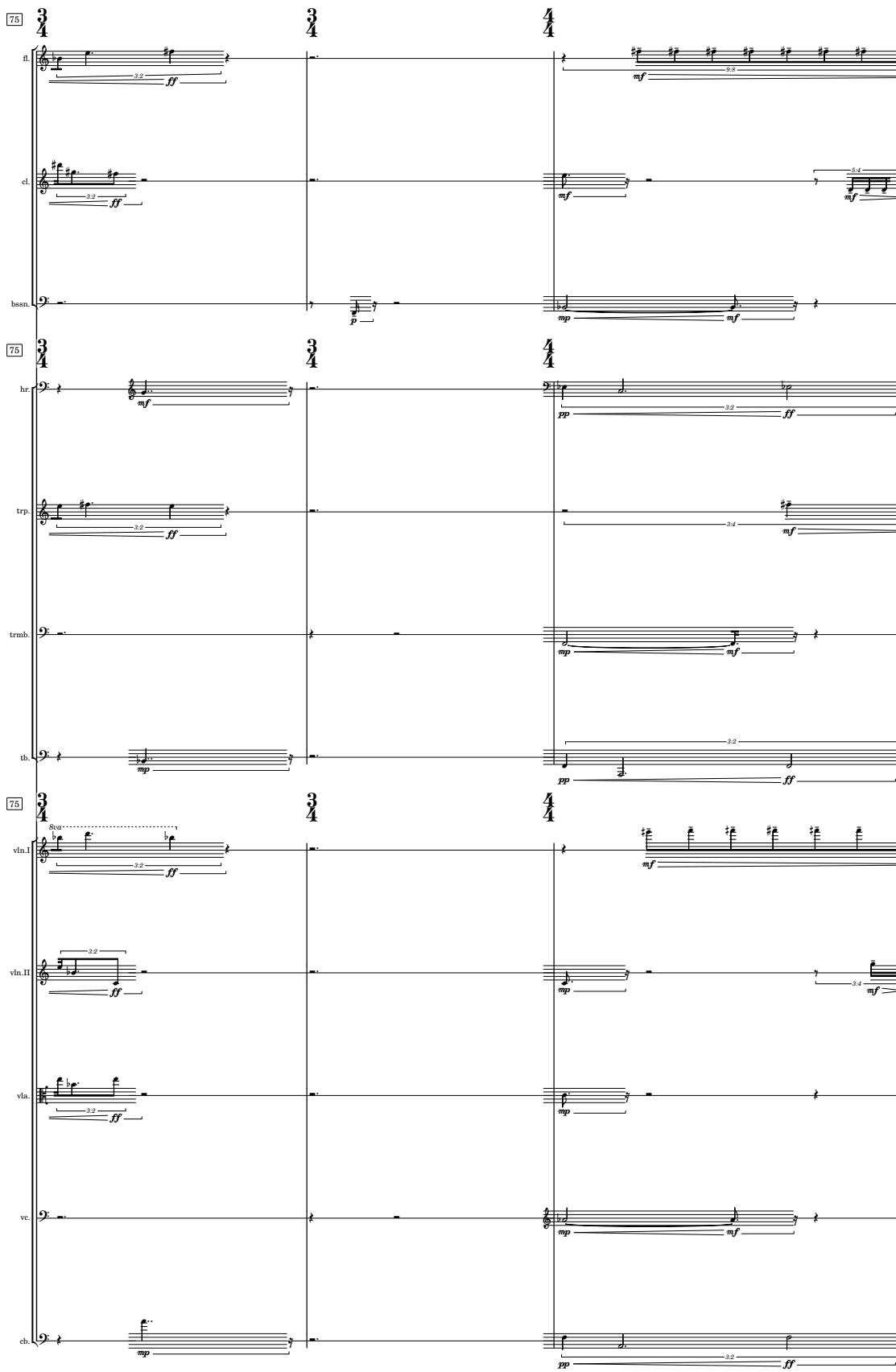
110 

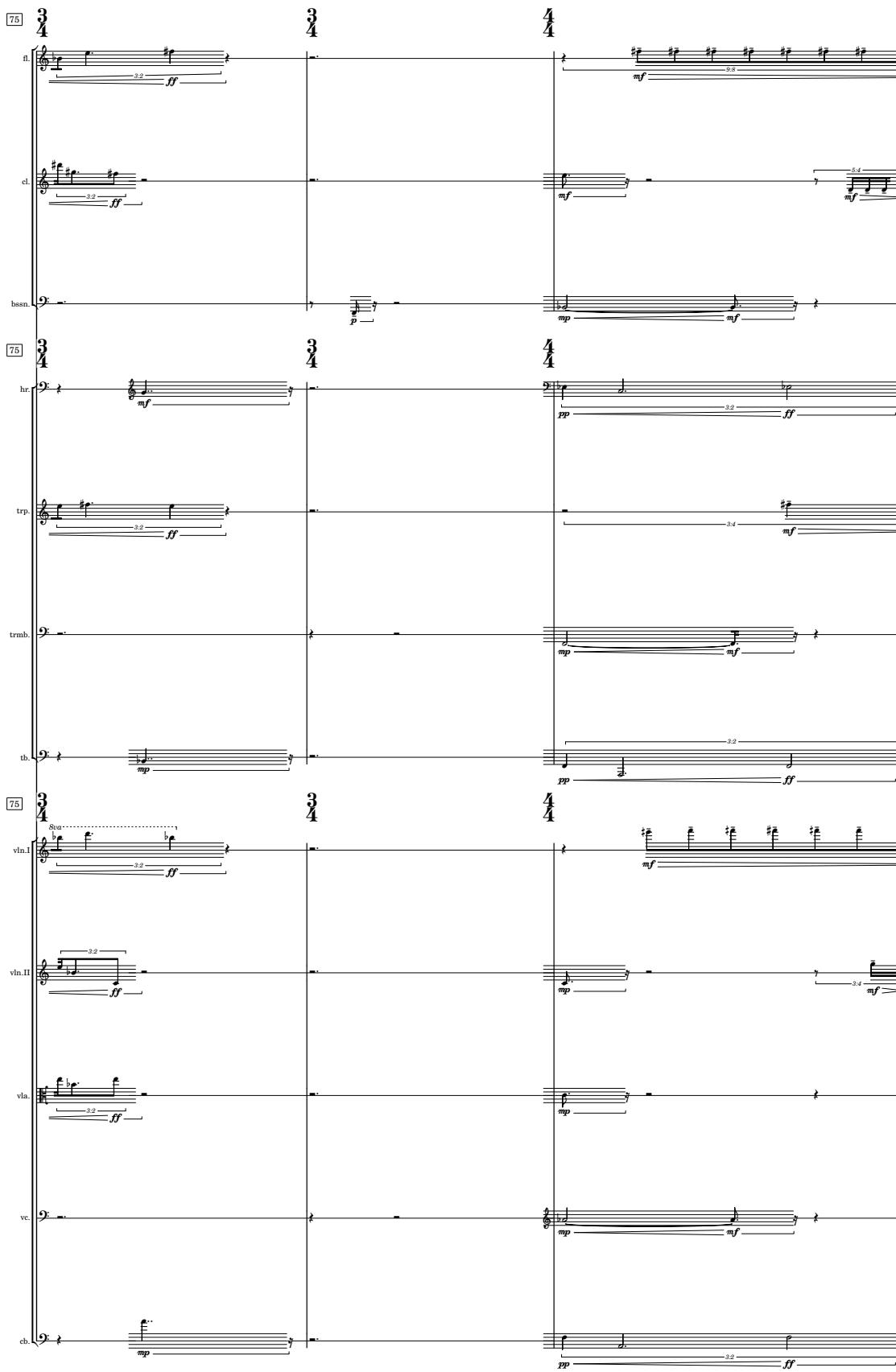
111 

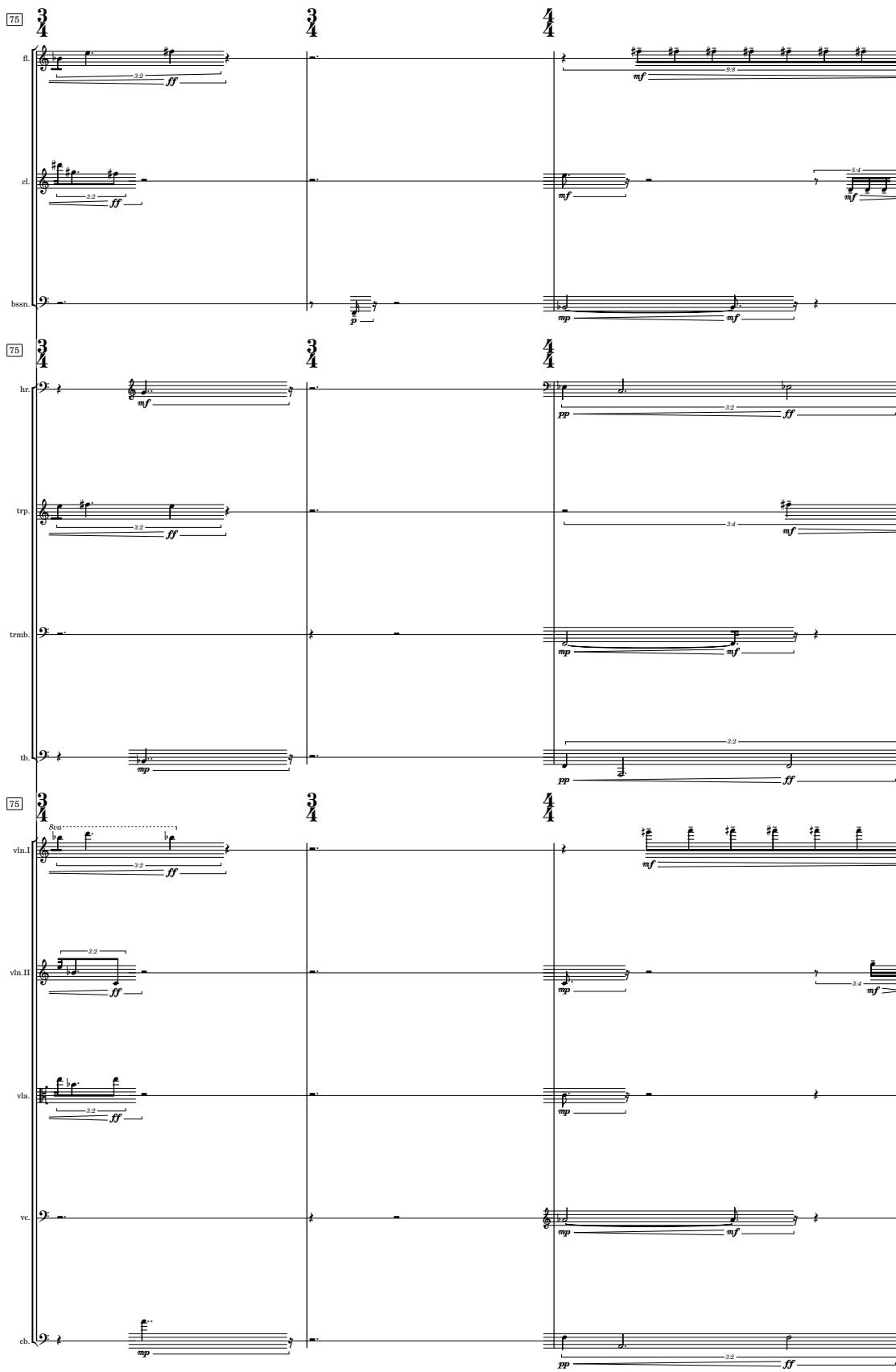
112 

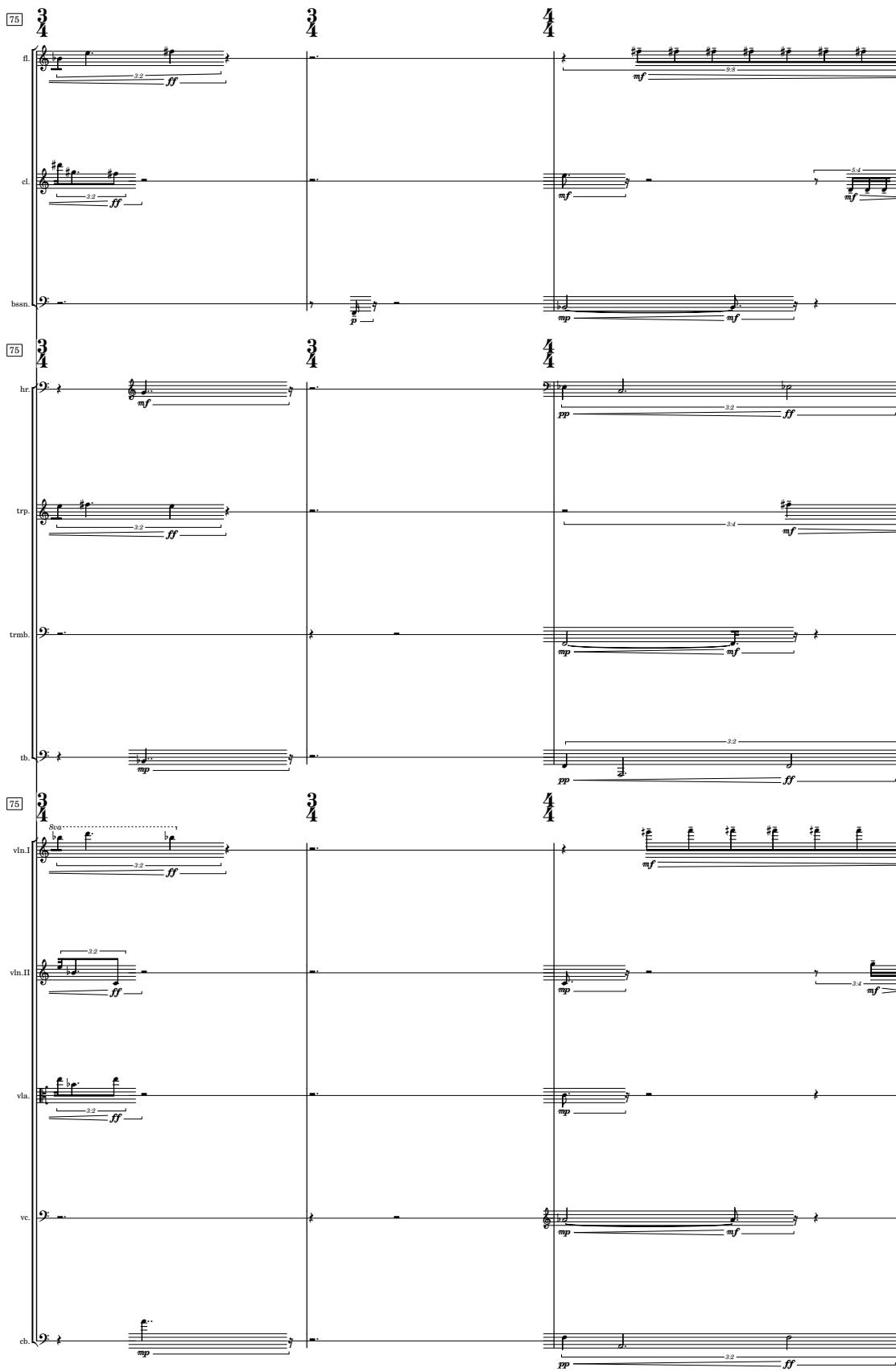
113 

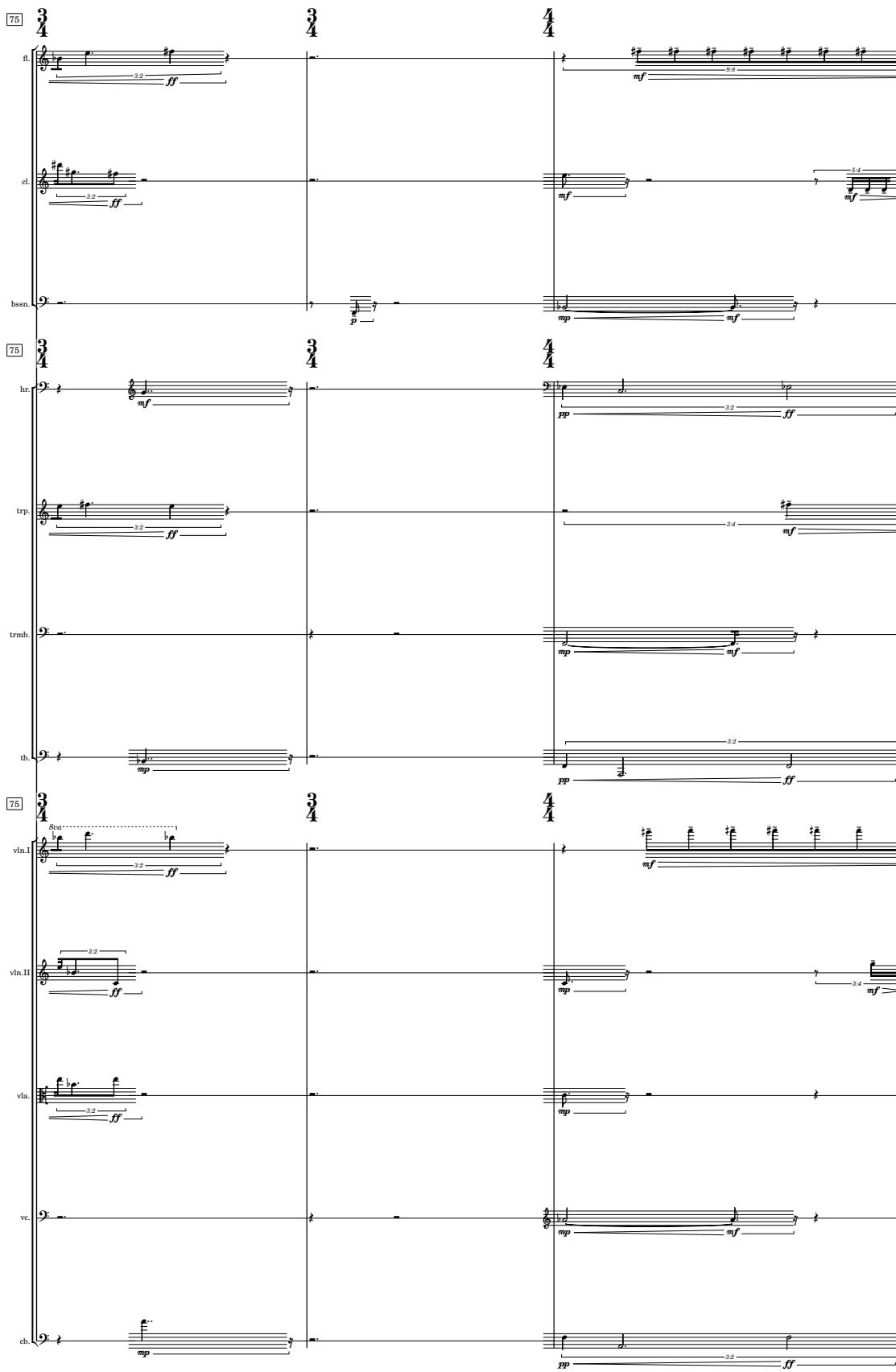
114 

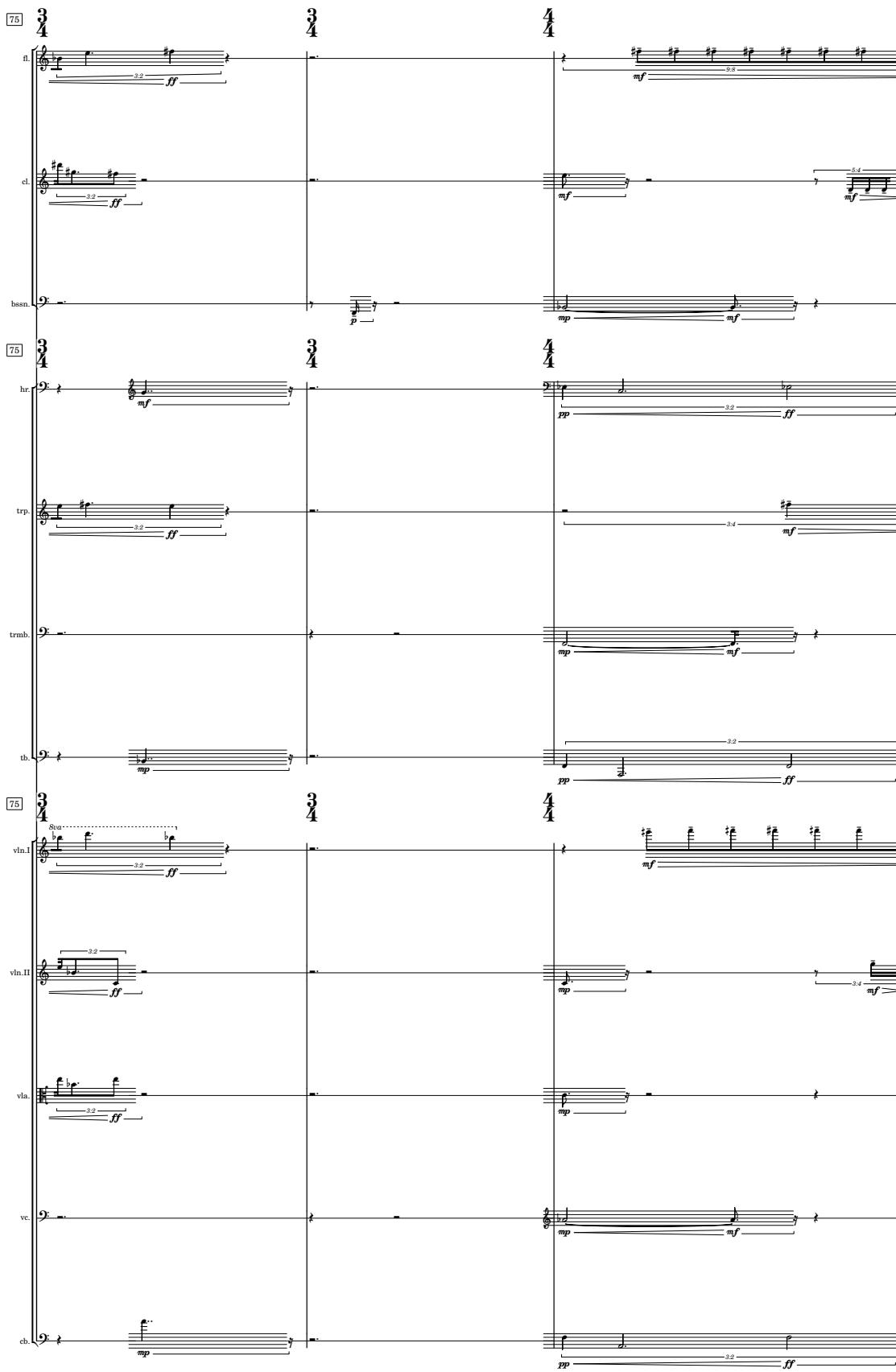
115 

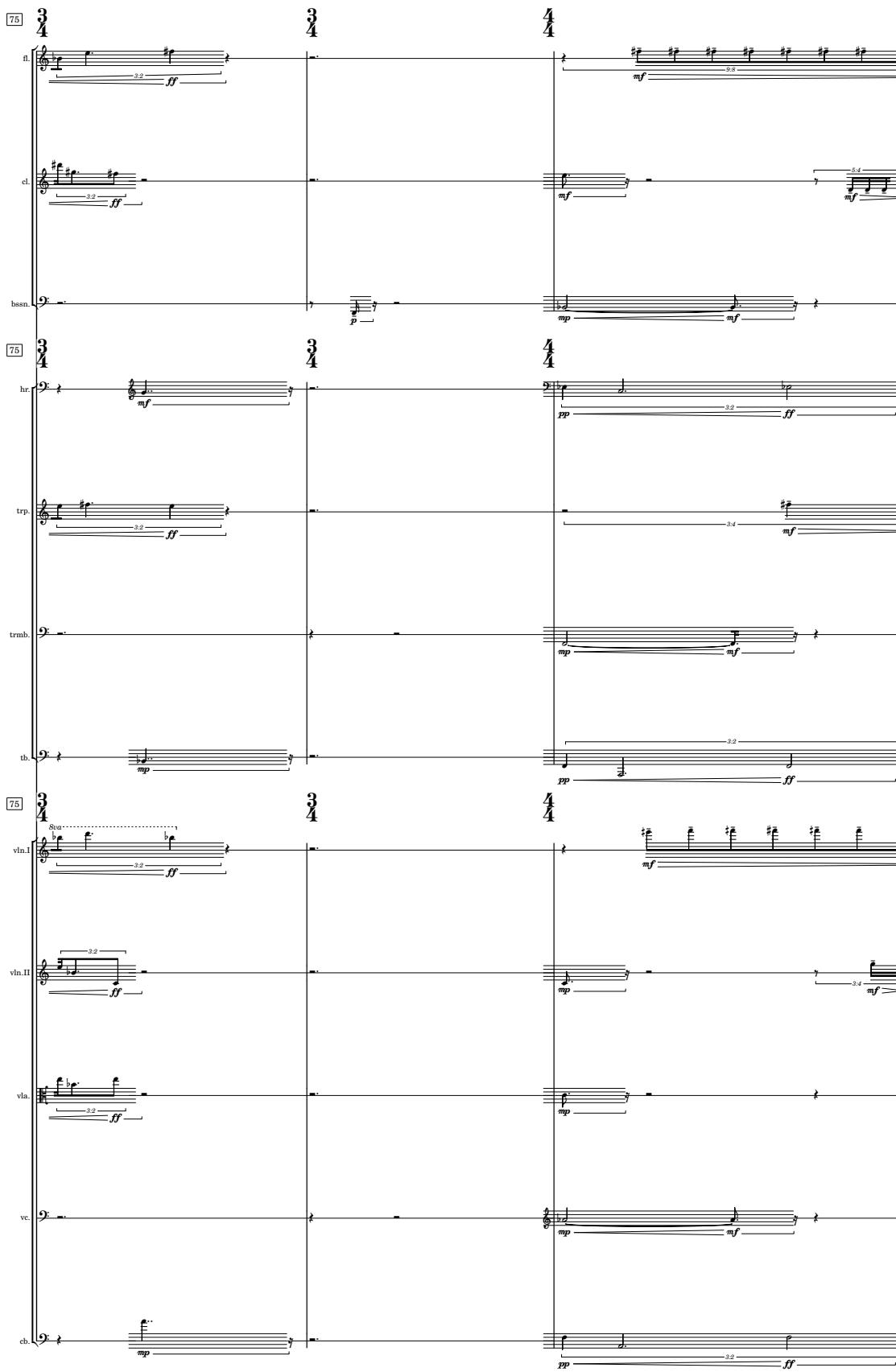
116 

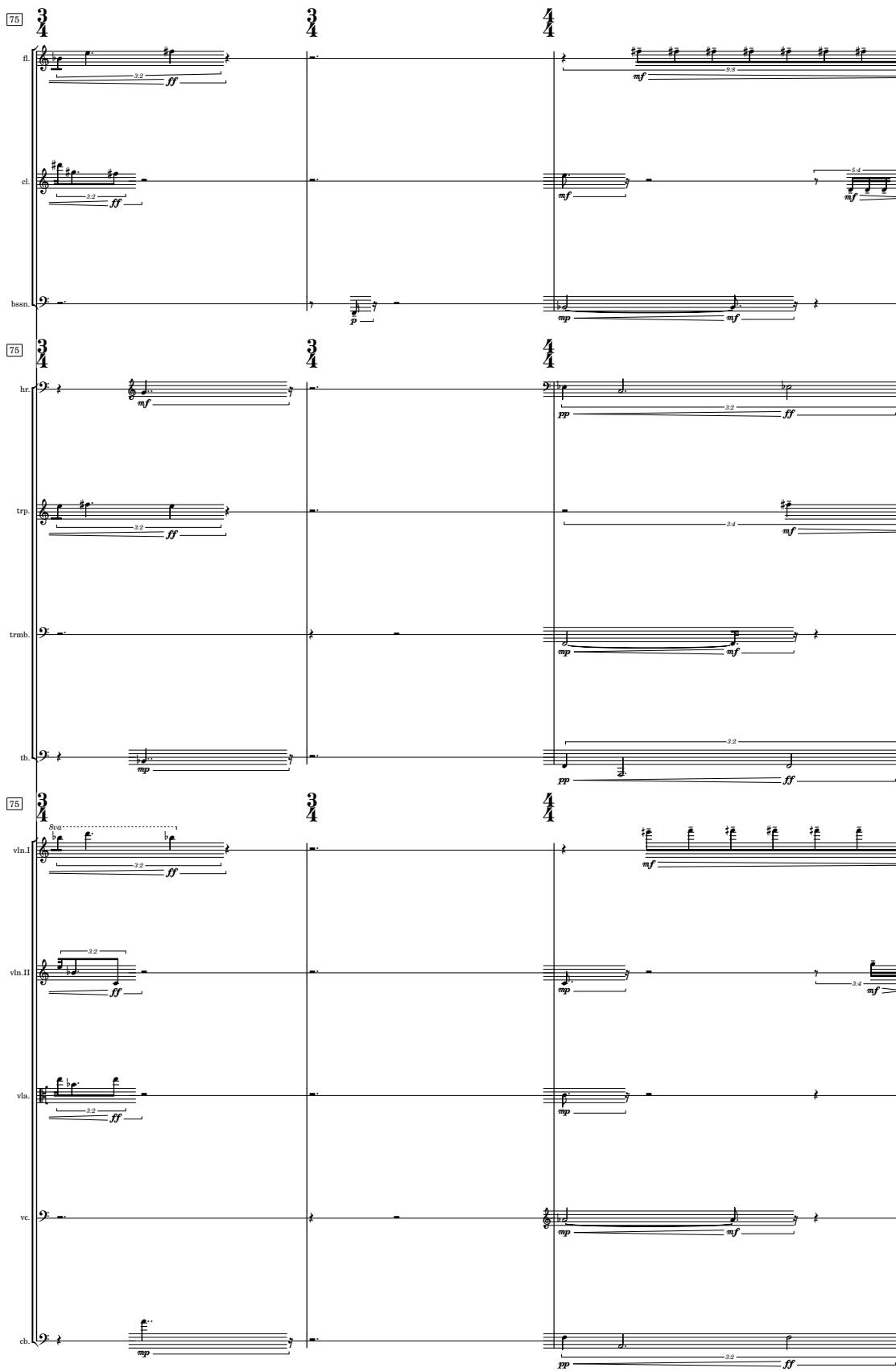
117 

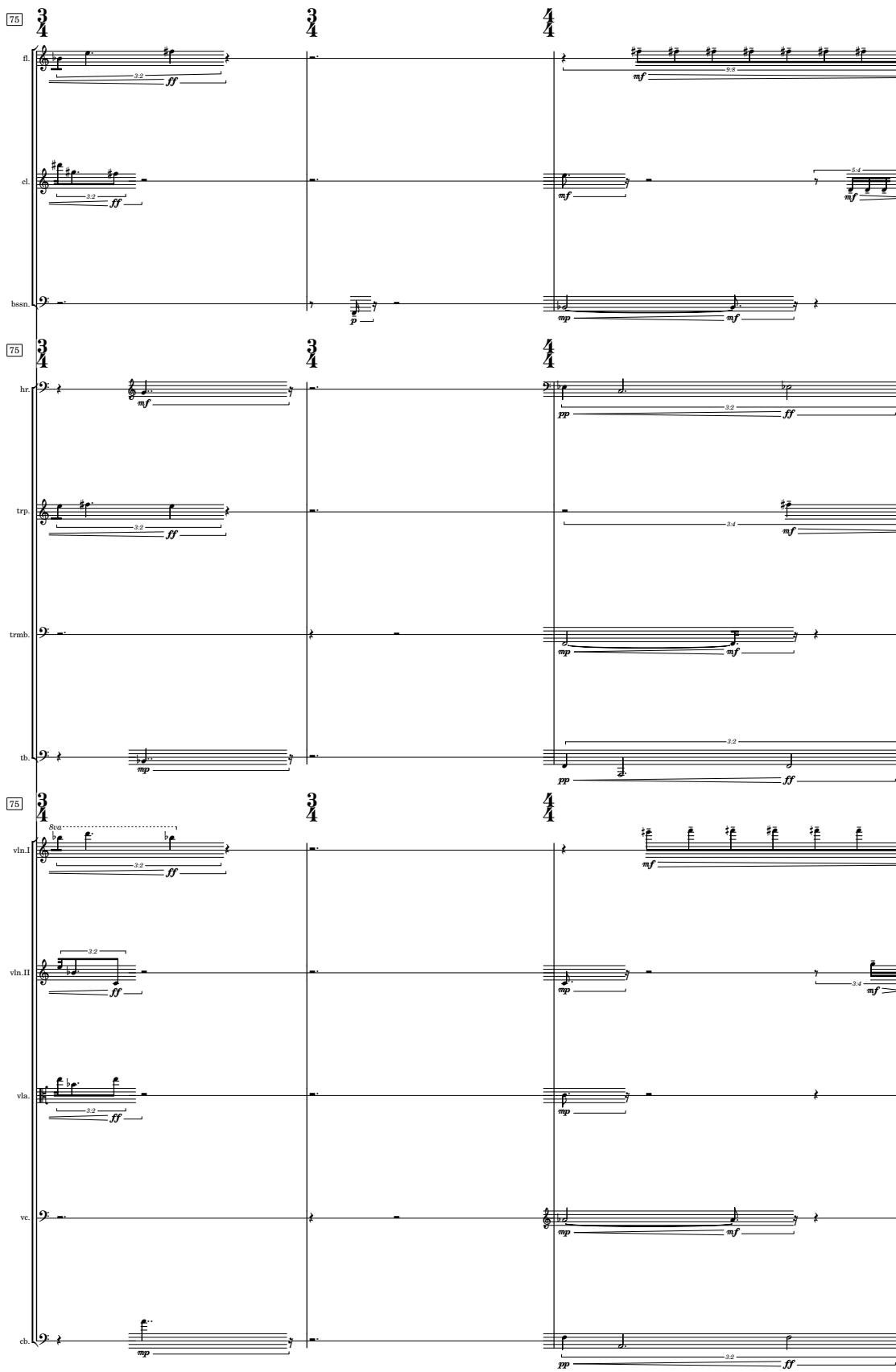
118 

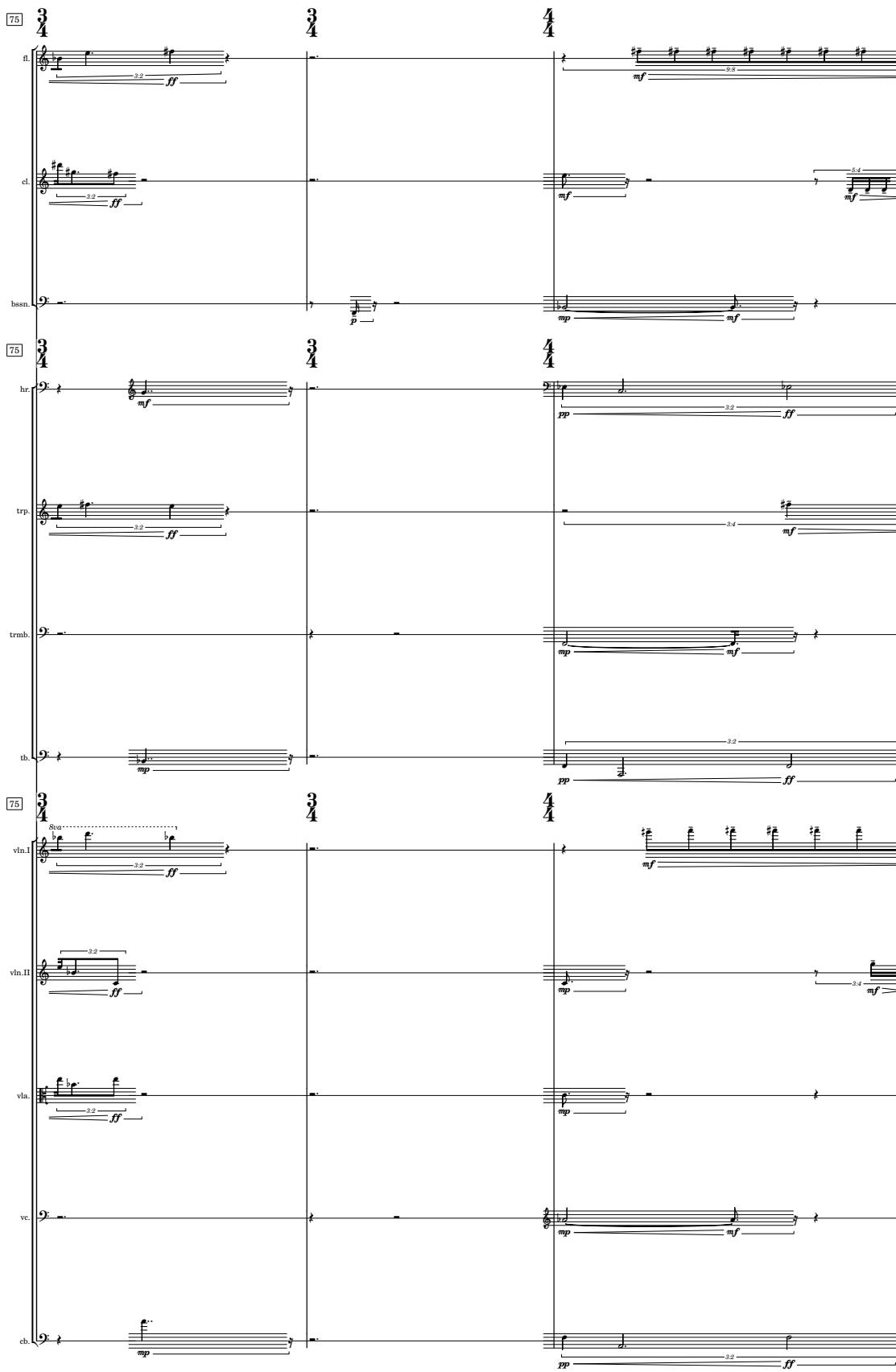
119 

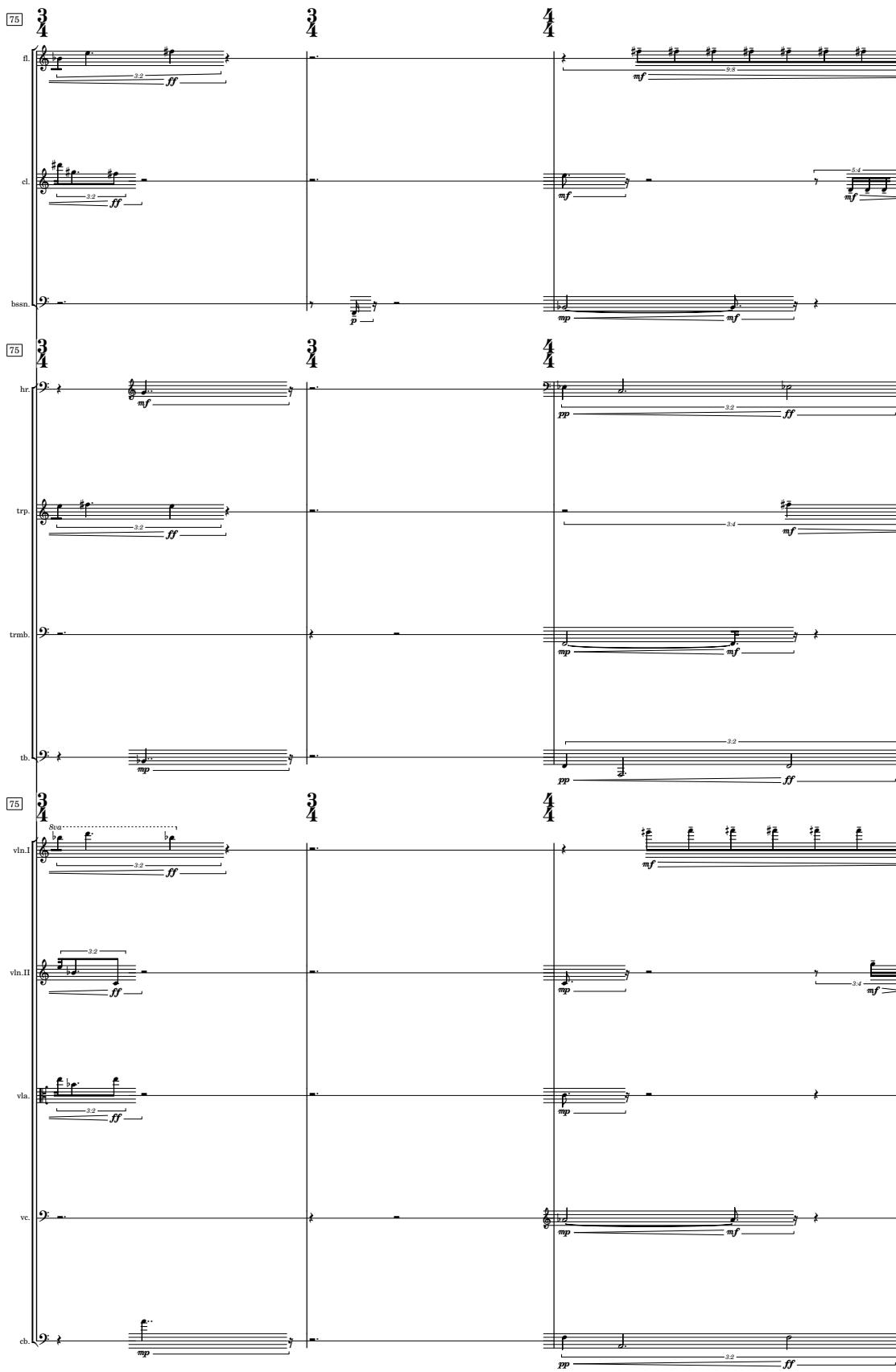
120 

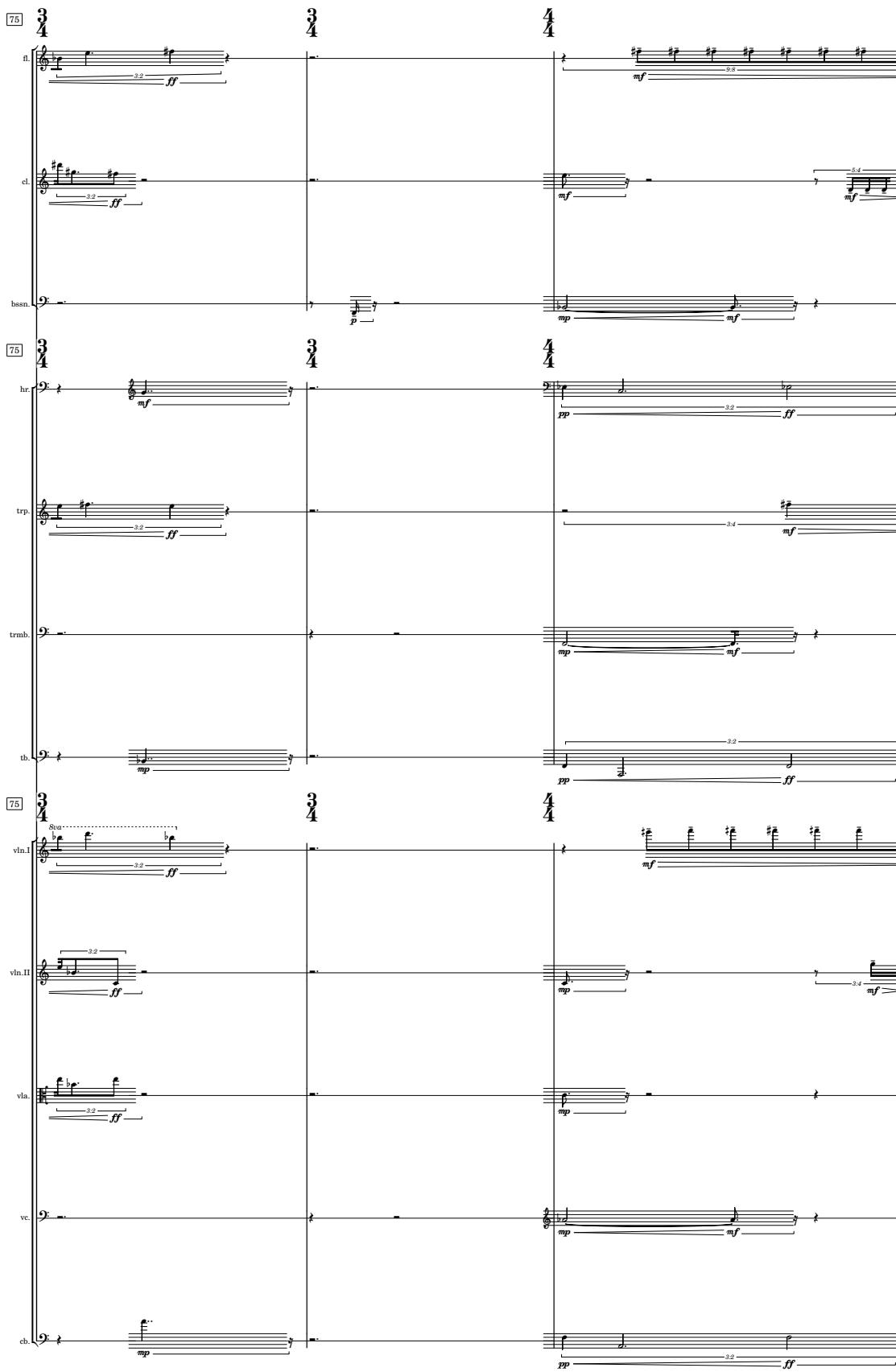
121 

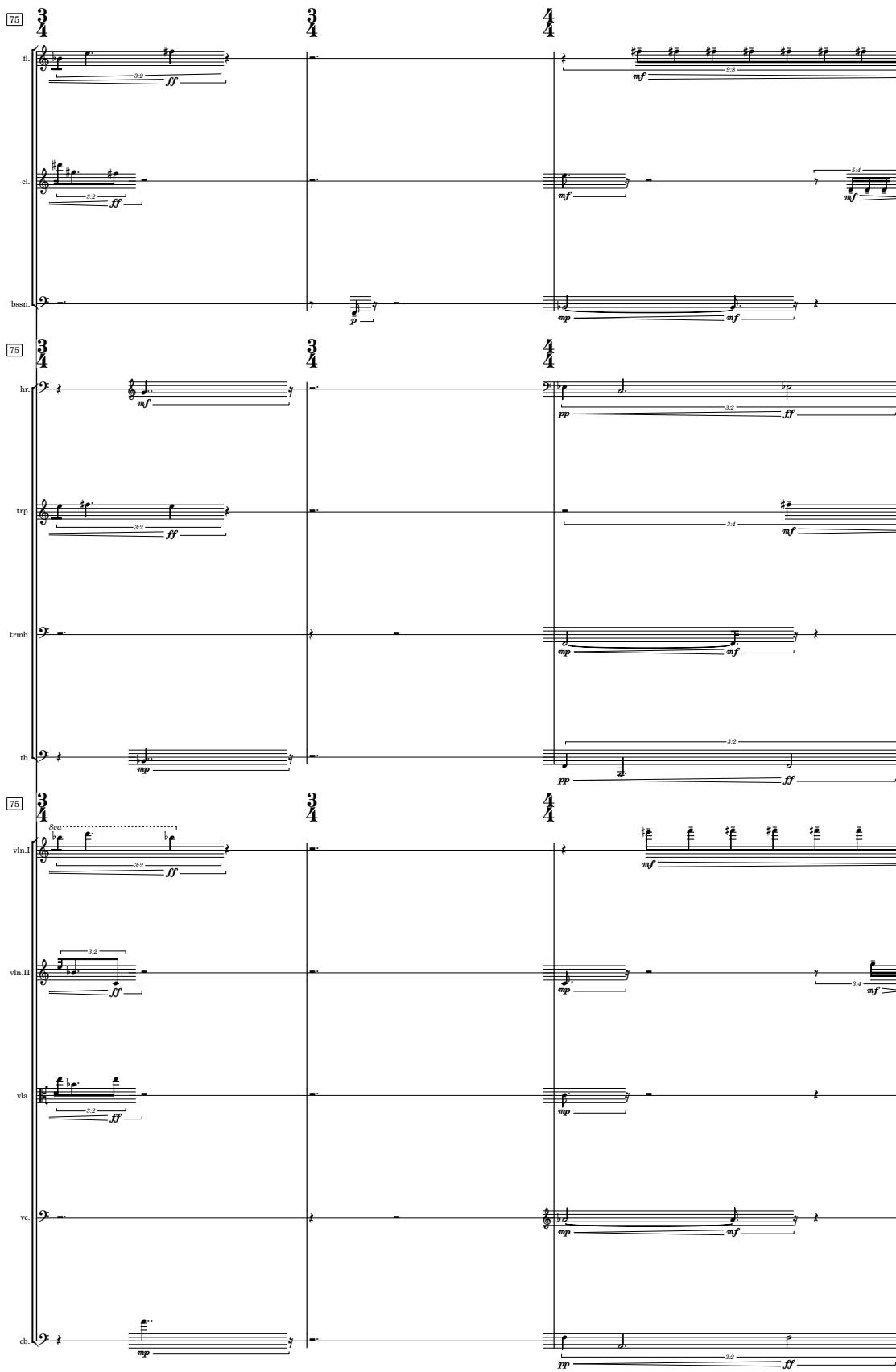
122 

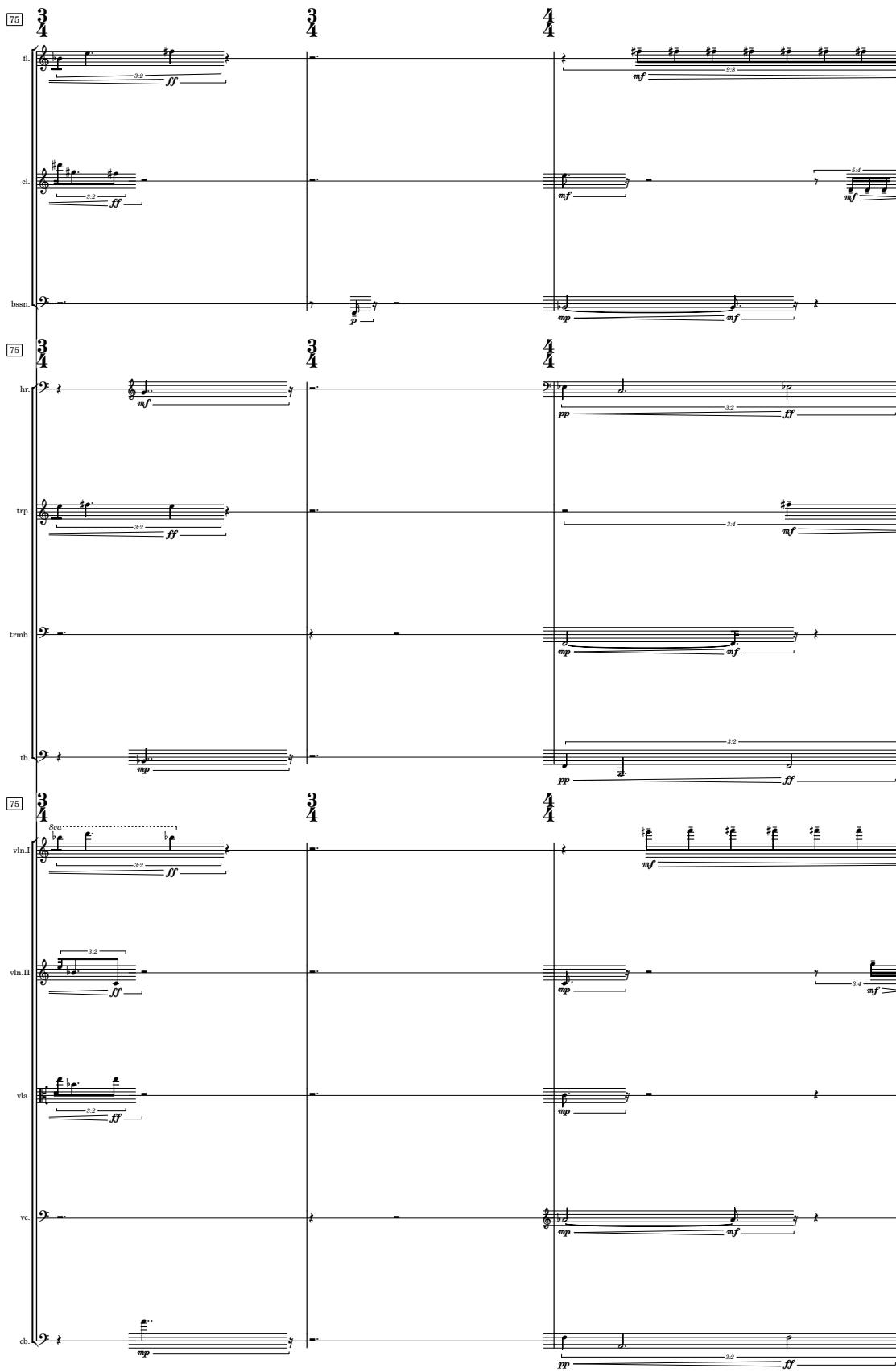
123 

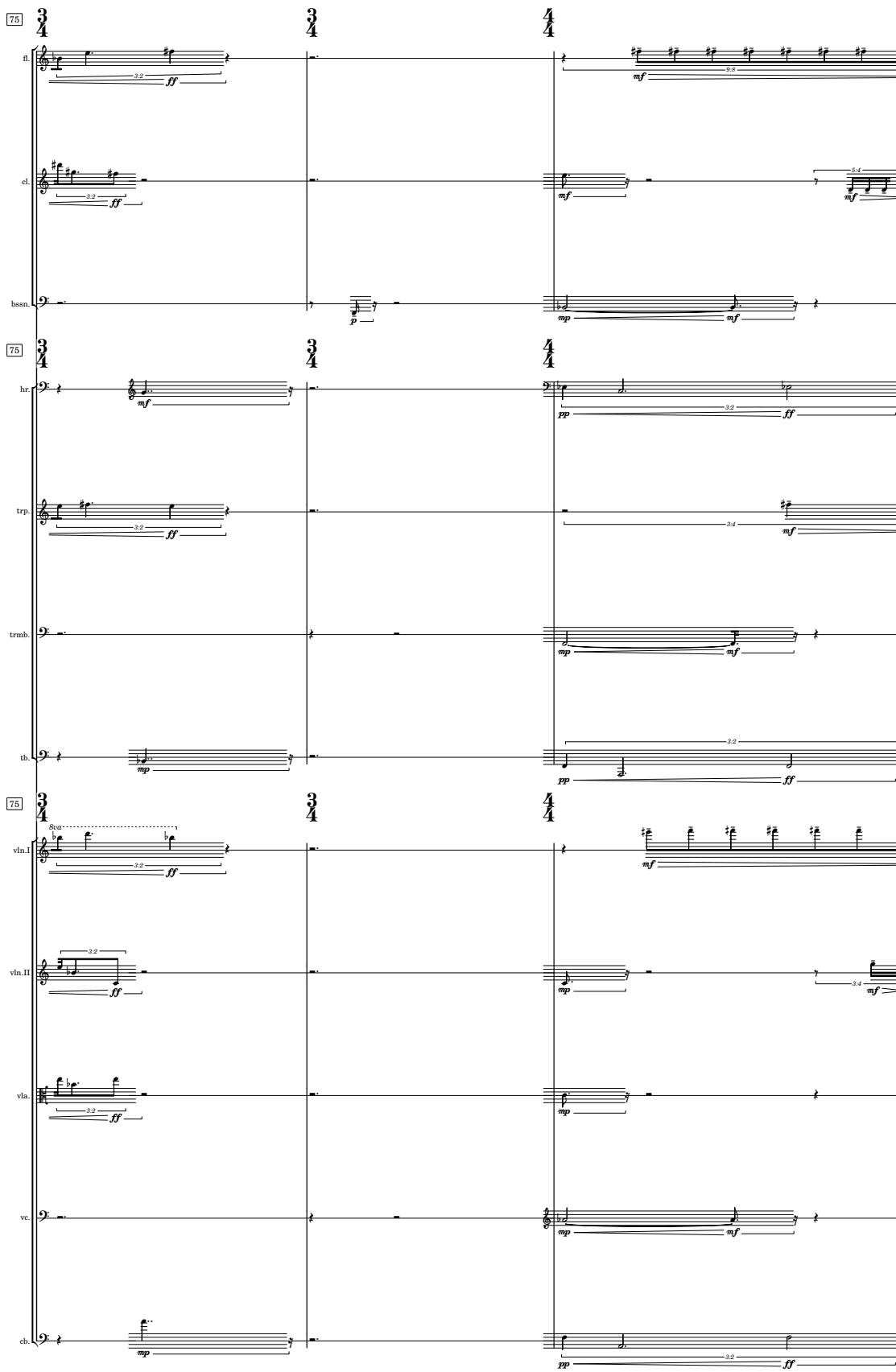
124 

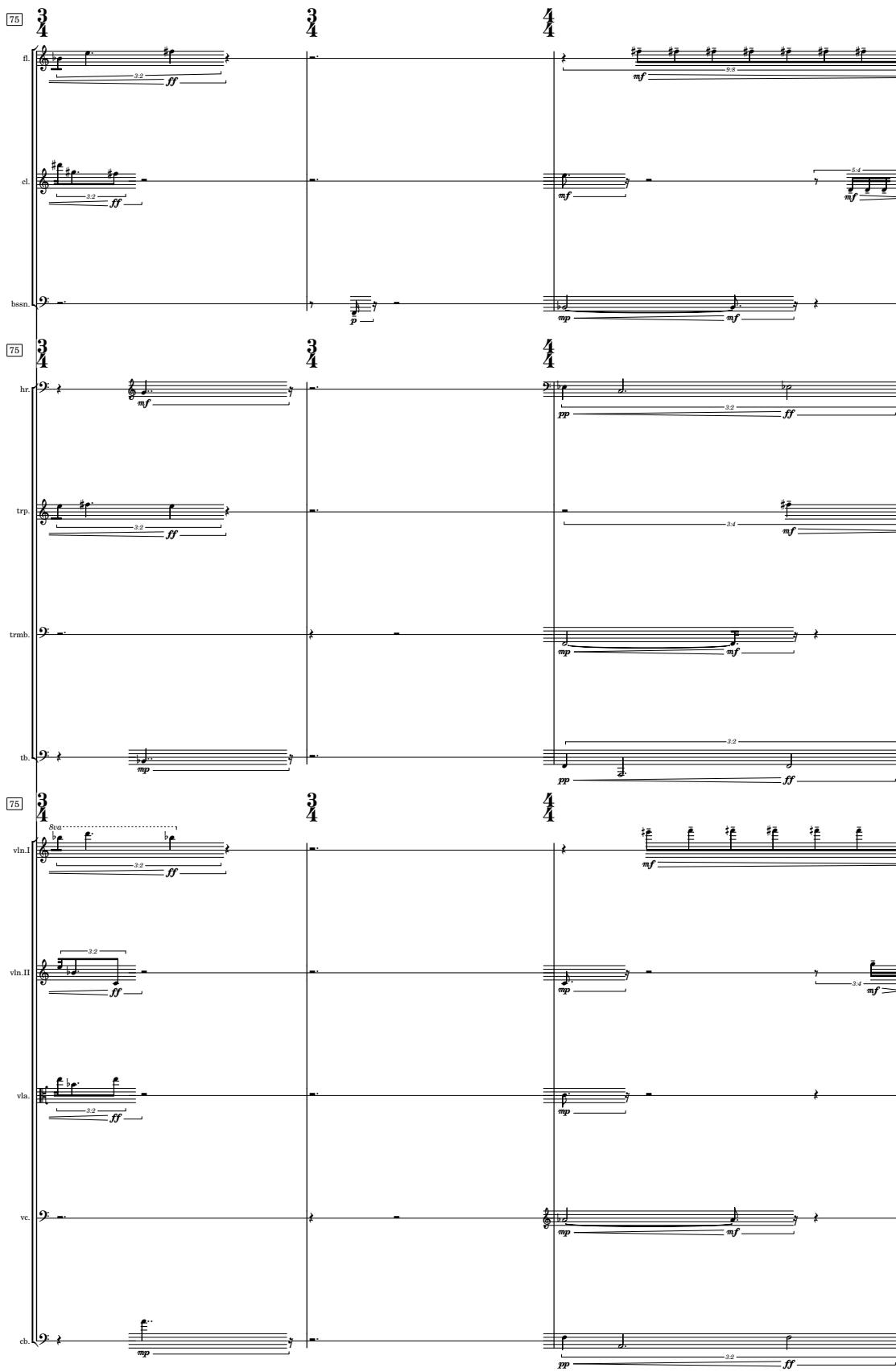
125 

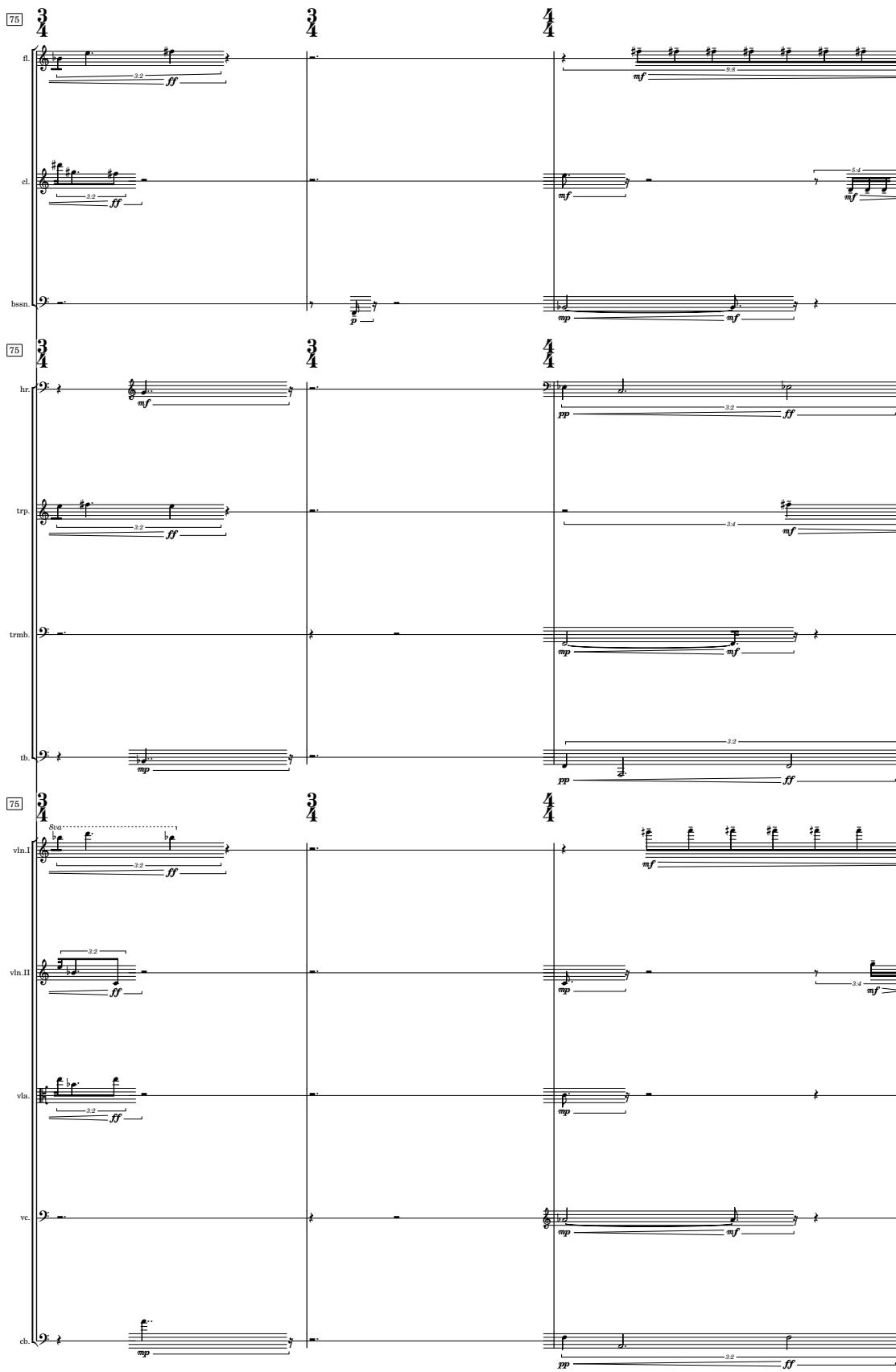
126 

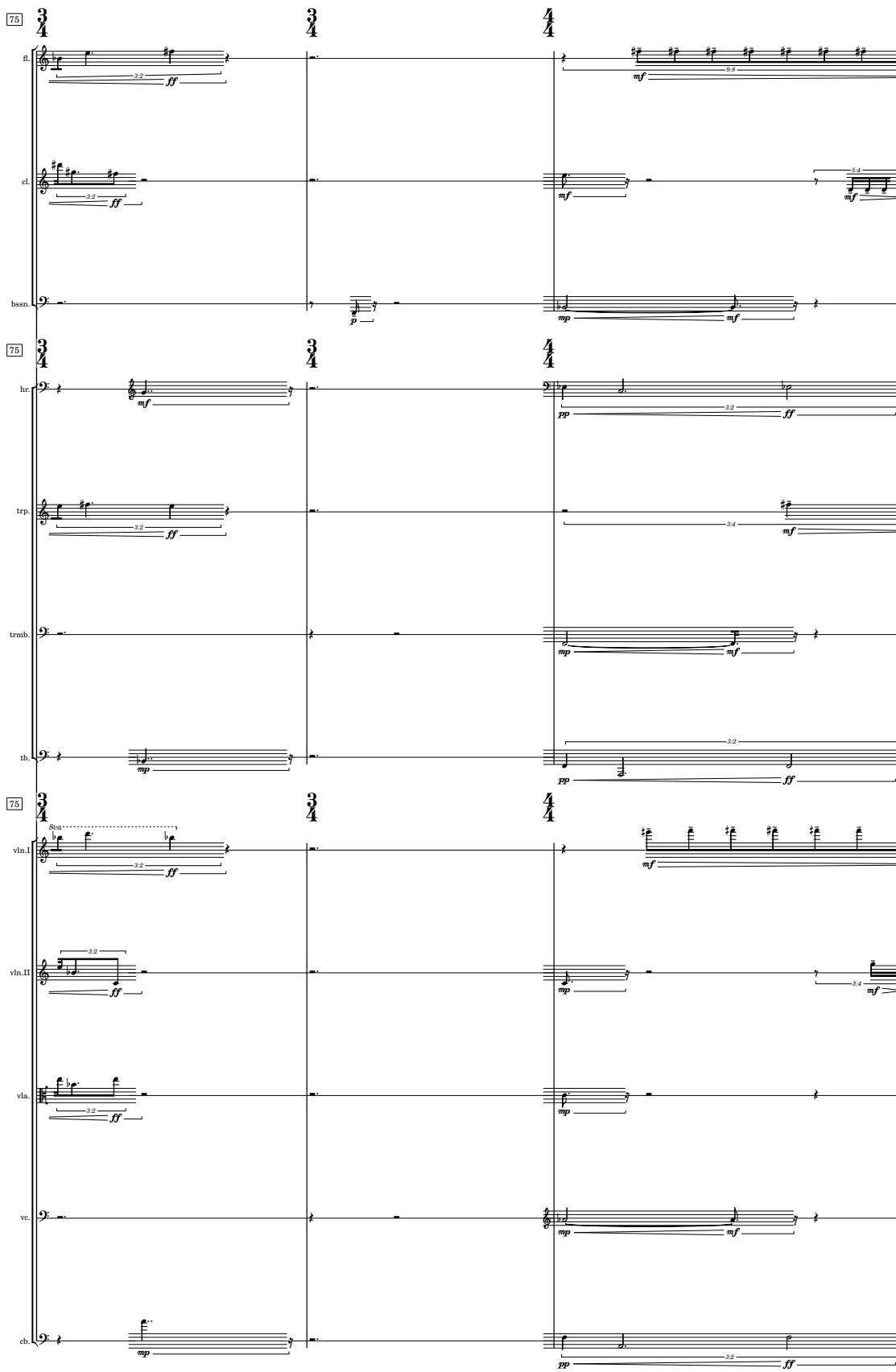
127 

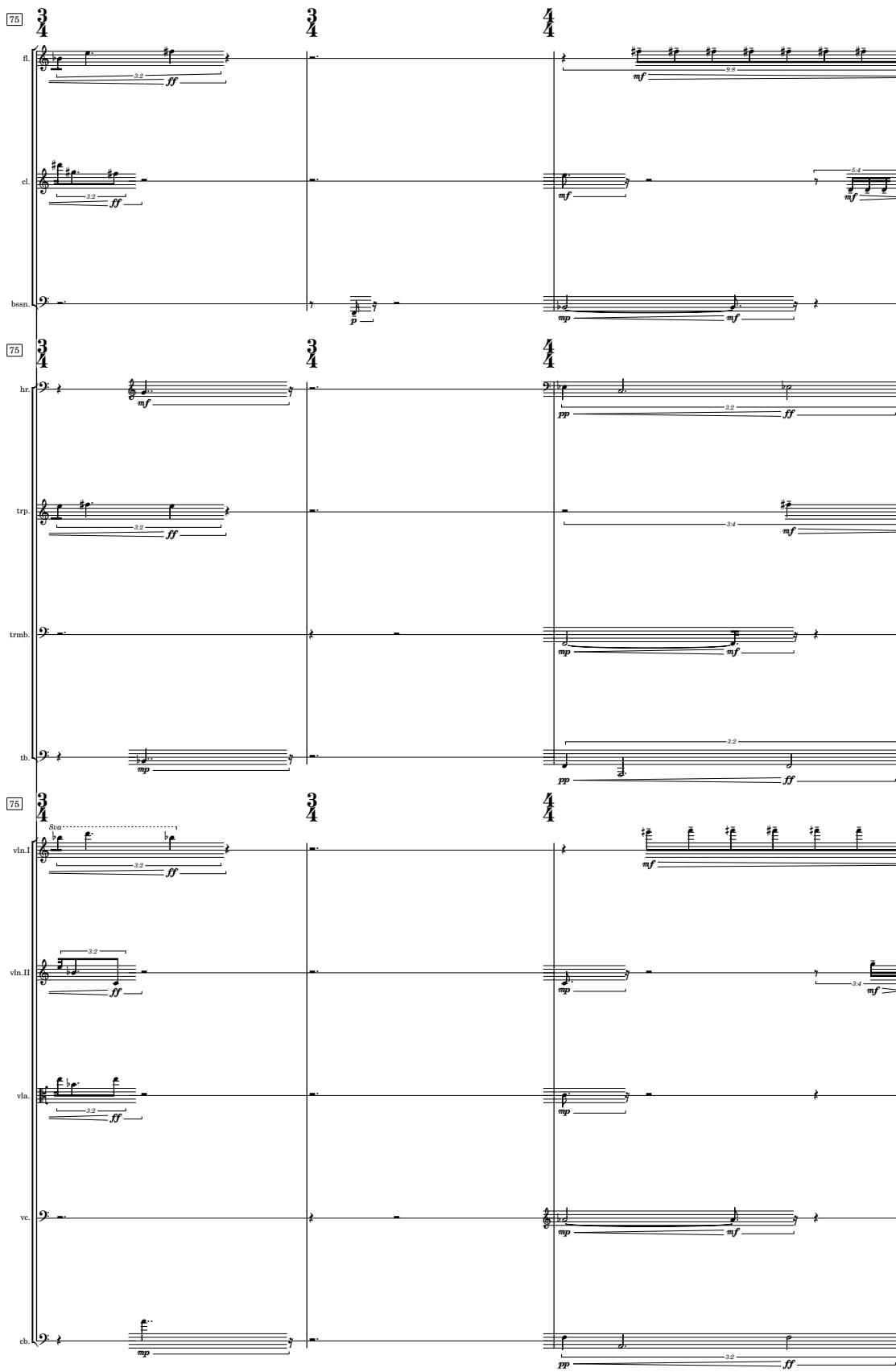
128 

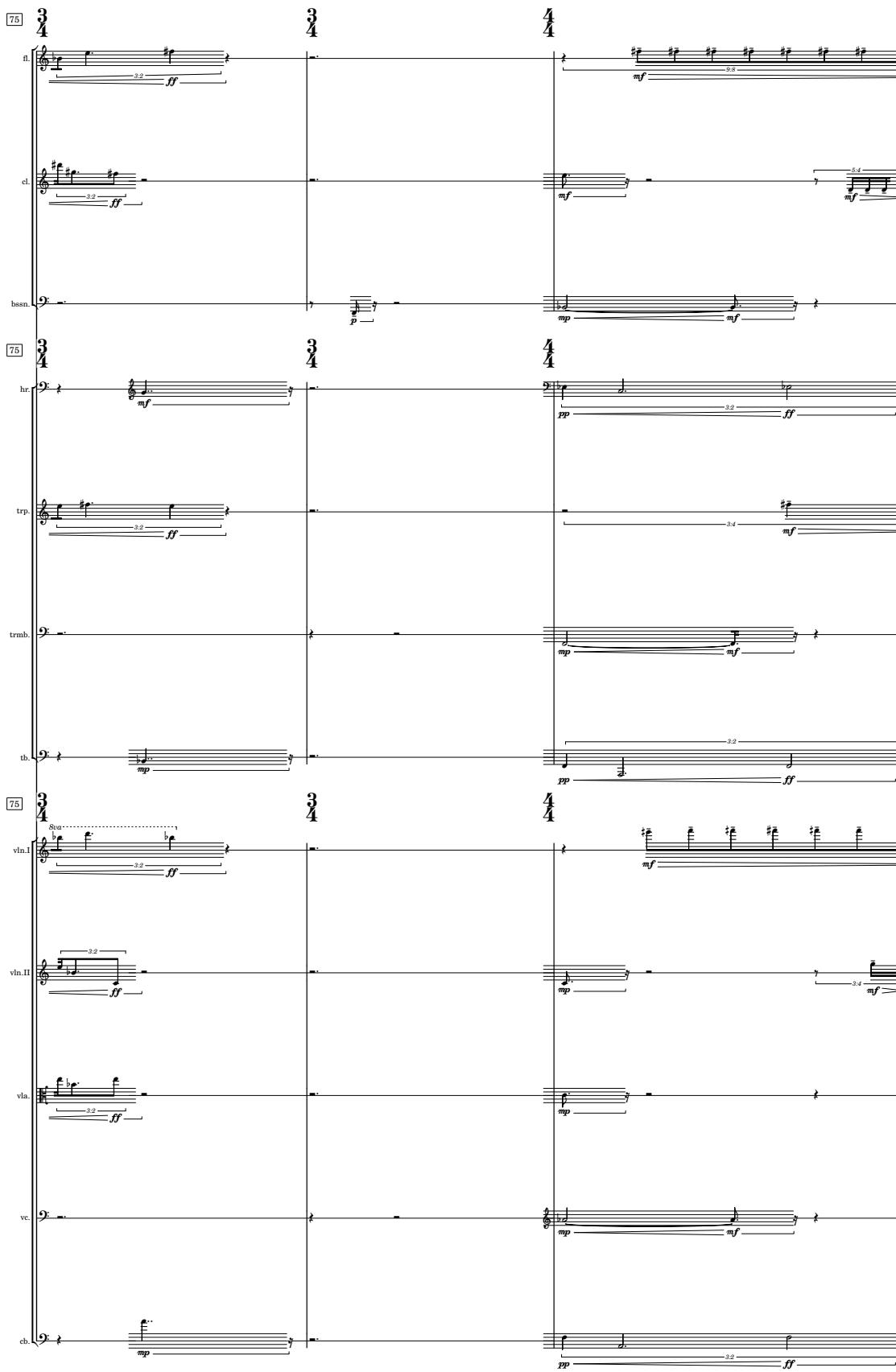
129 

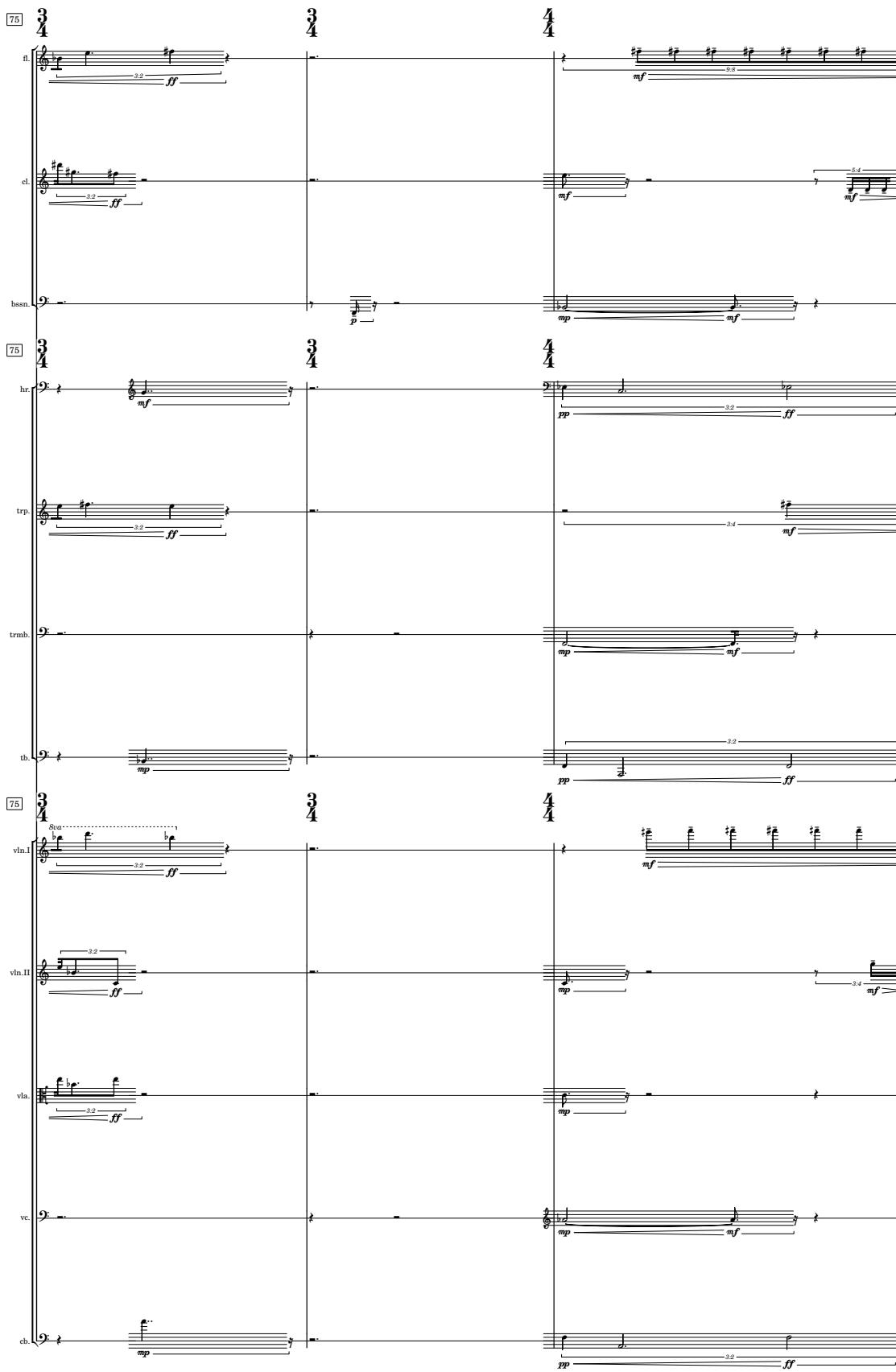
130 

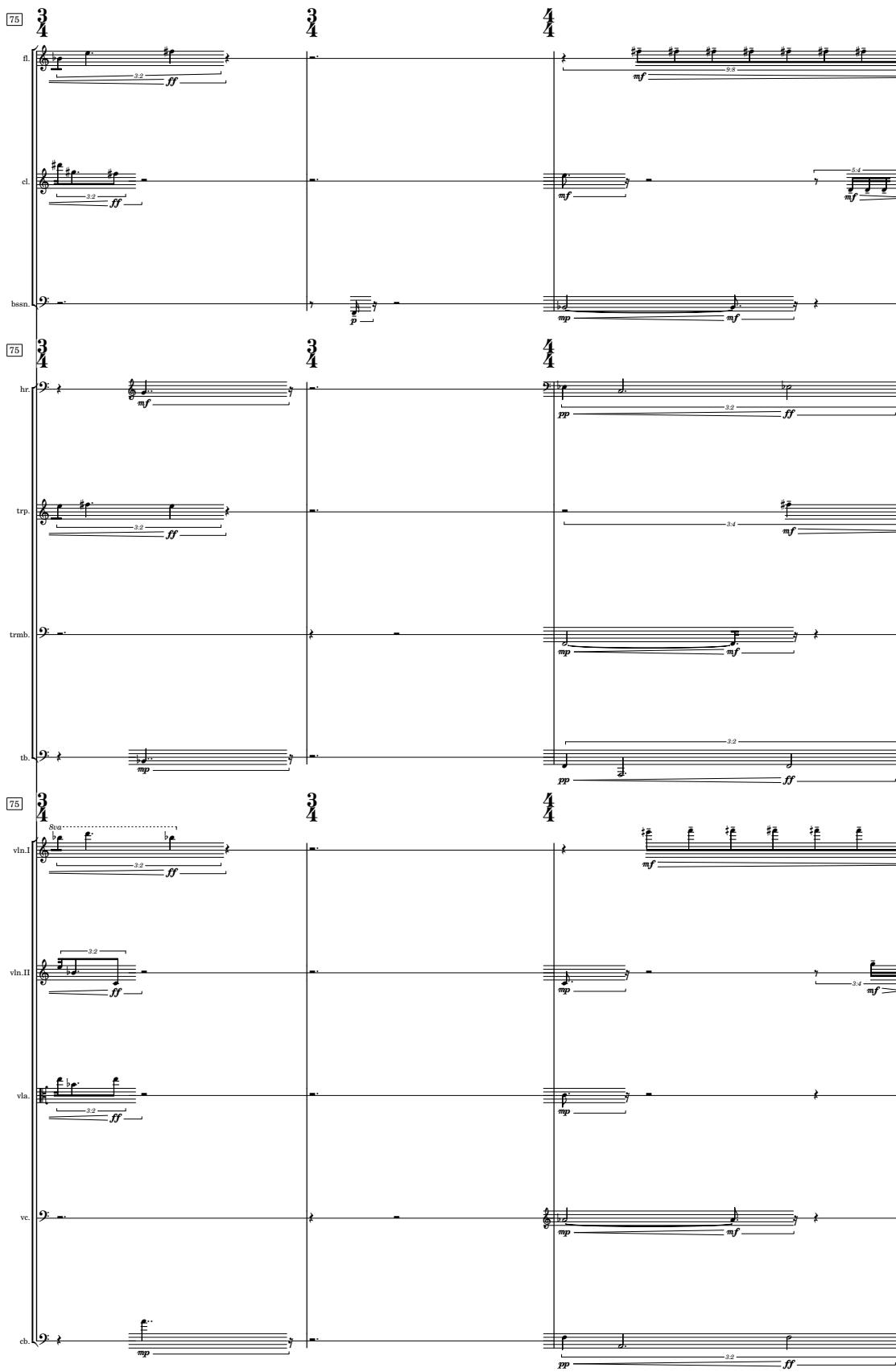
131 

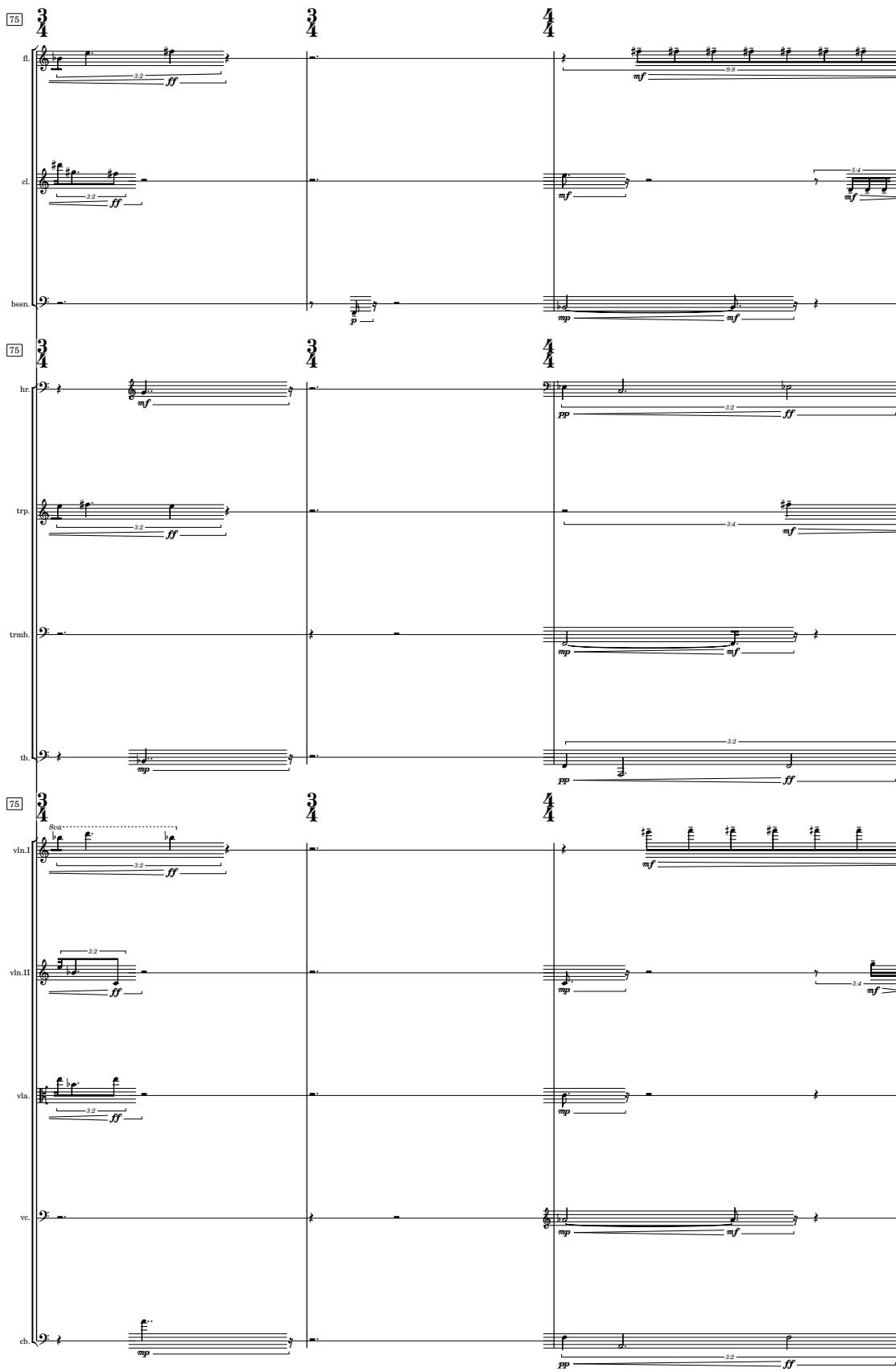
132 

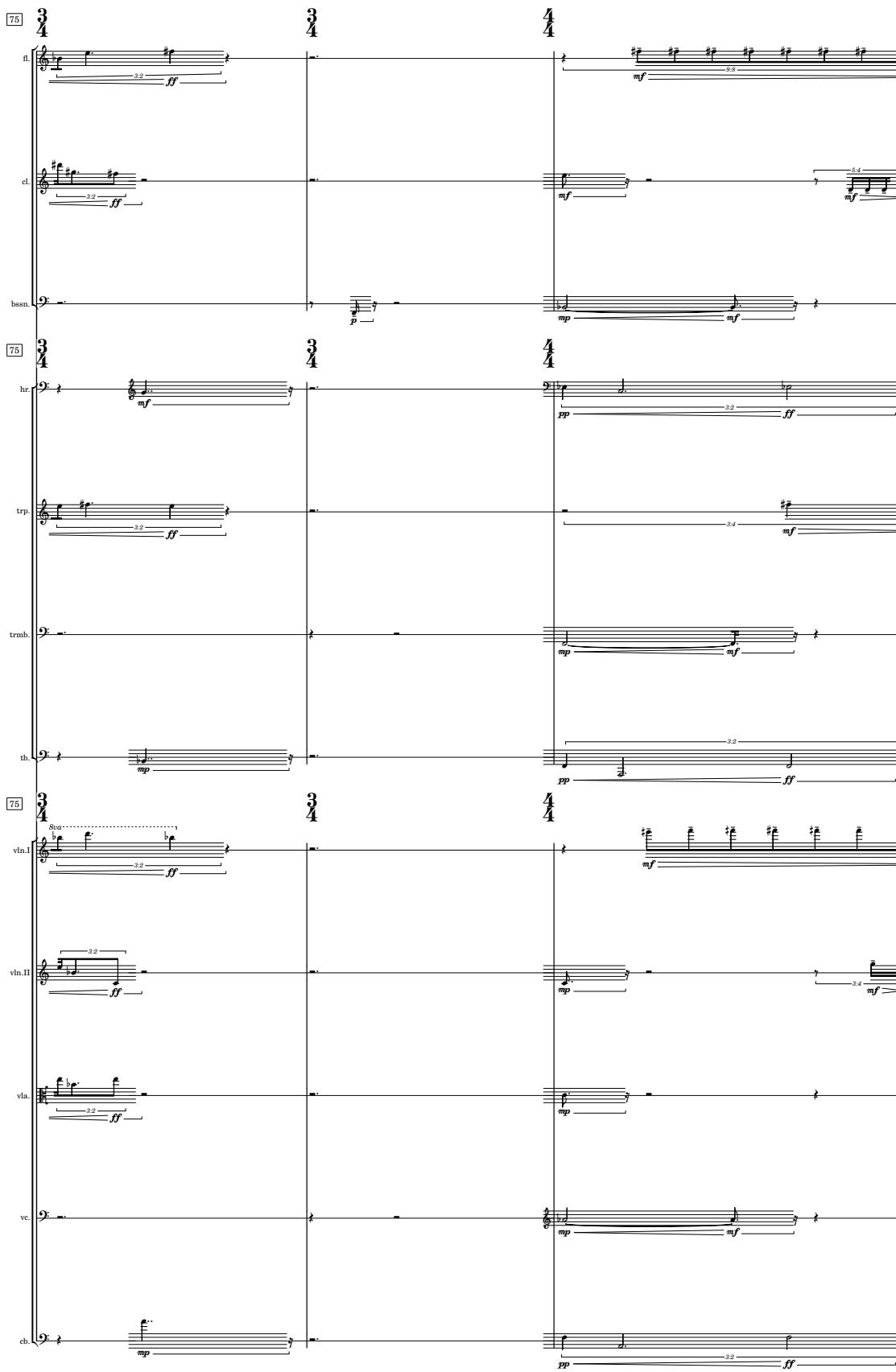
133 

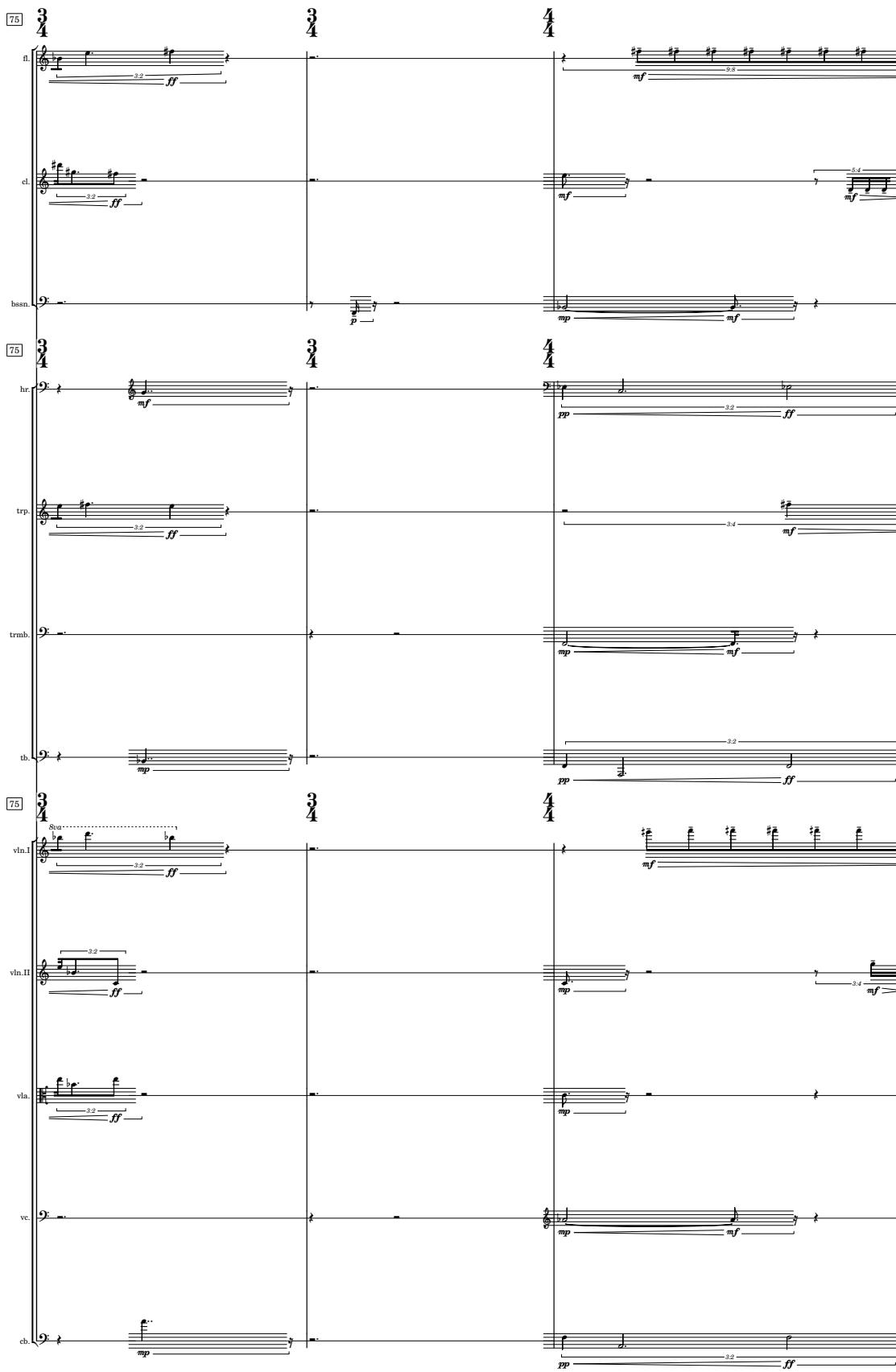
134 

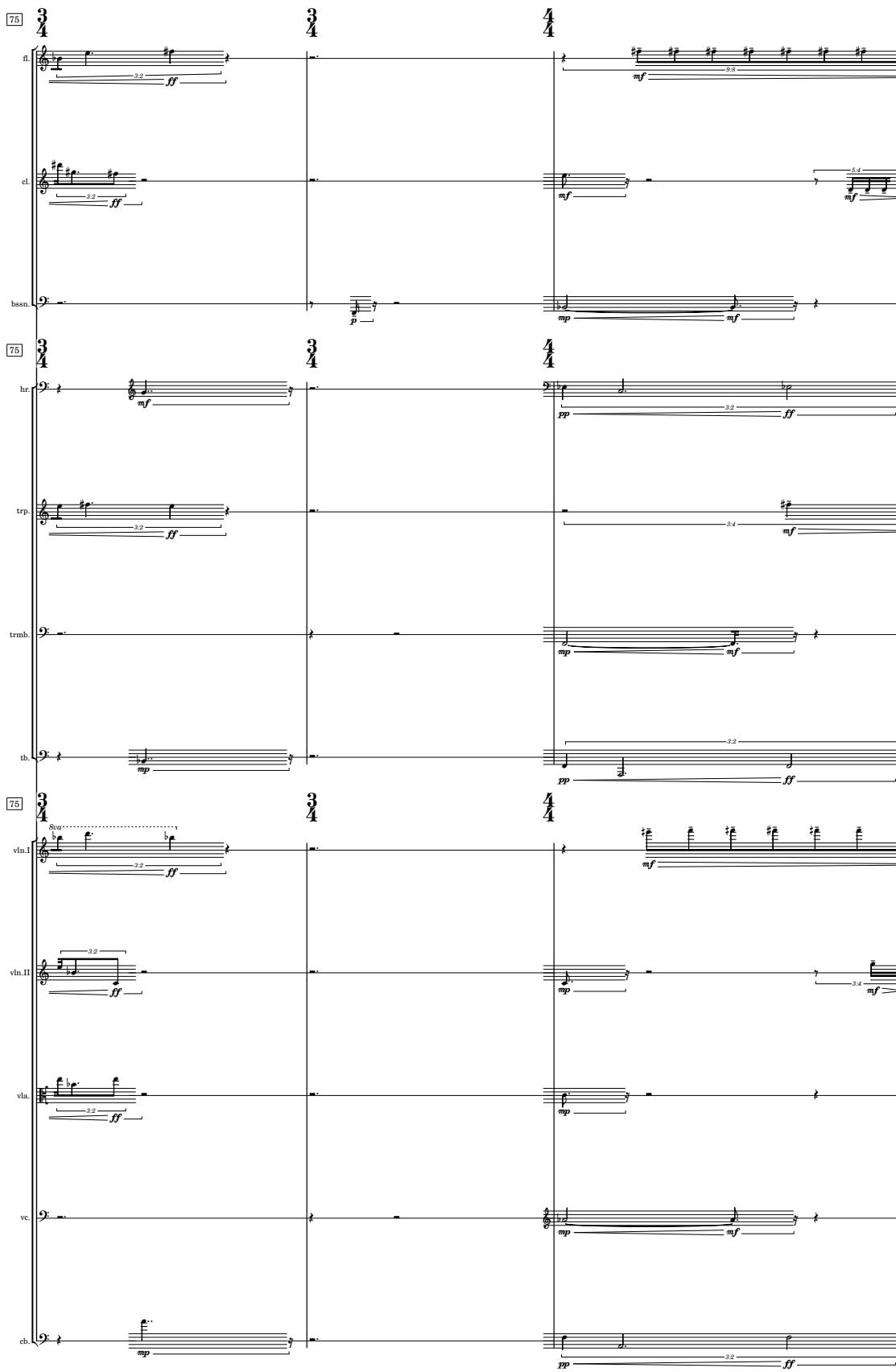
135 

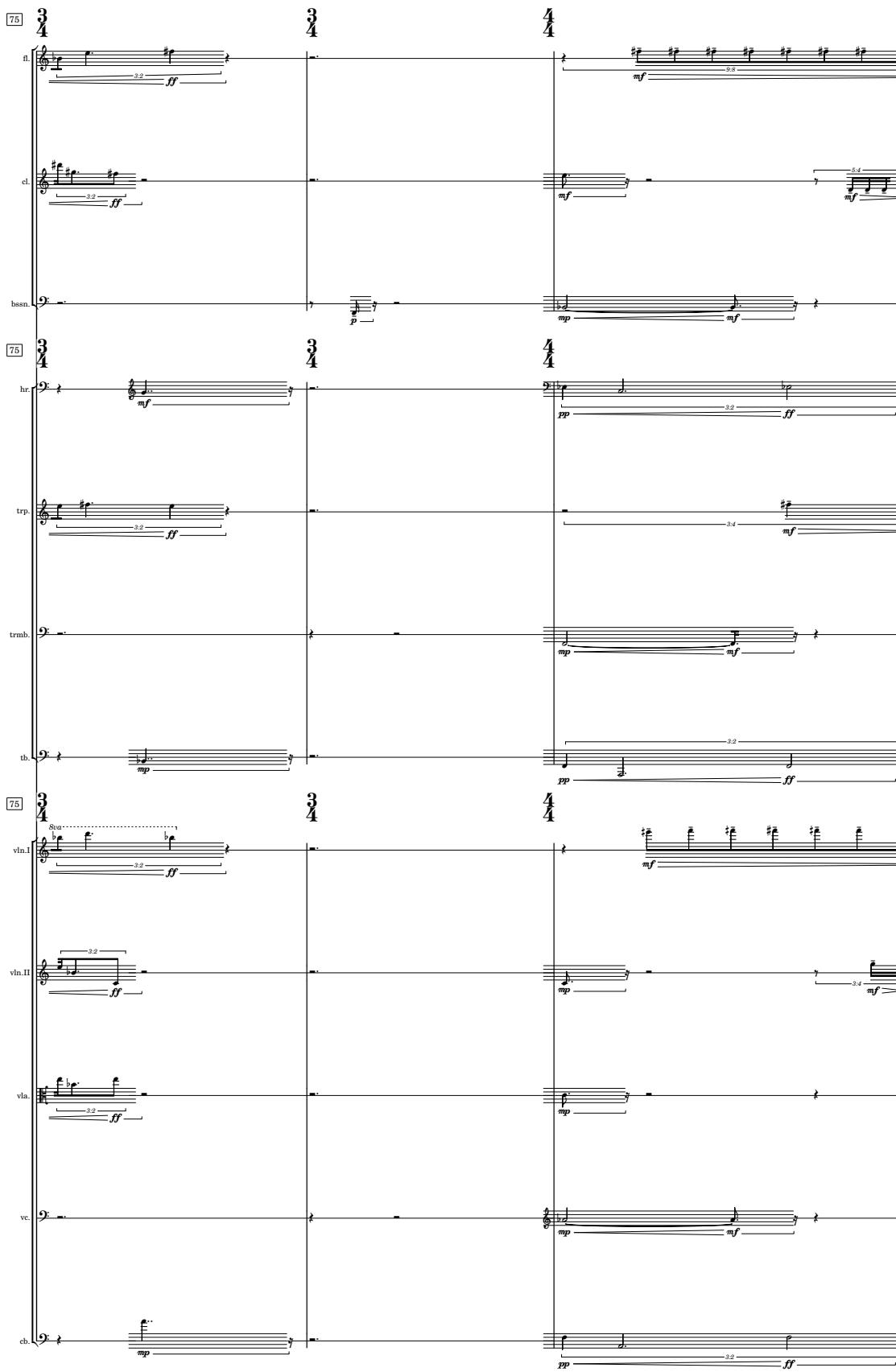
136 

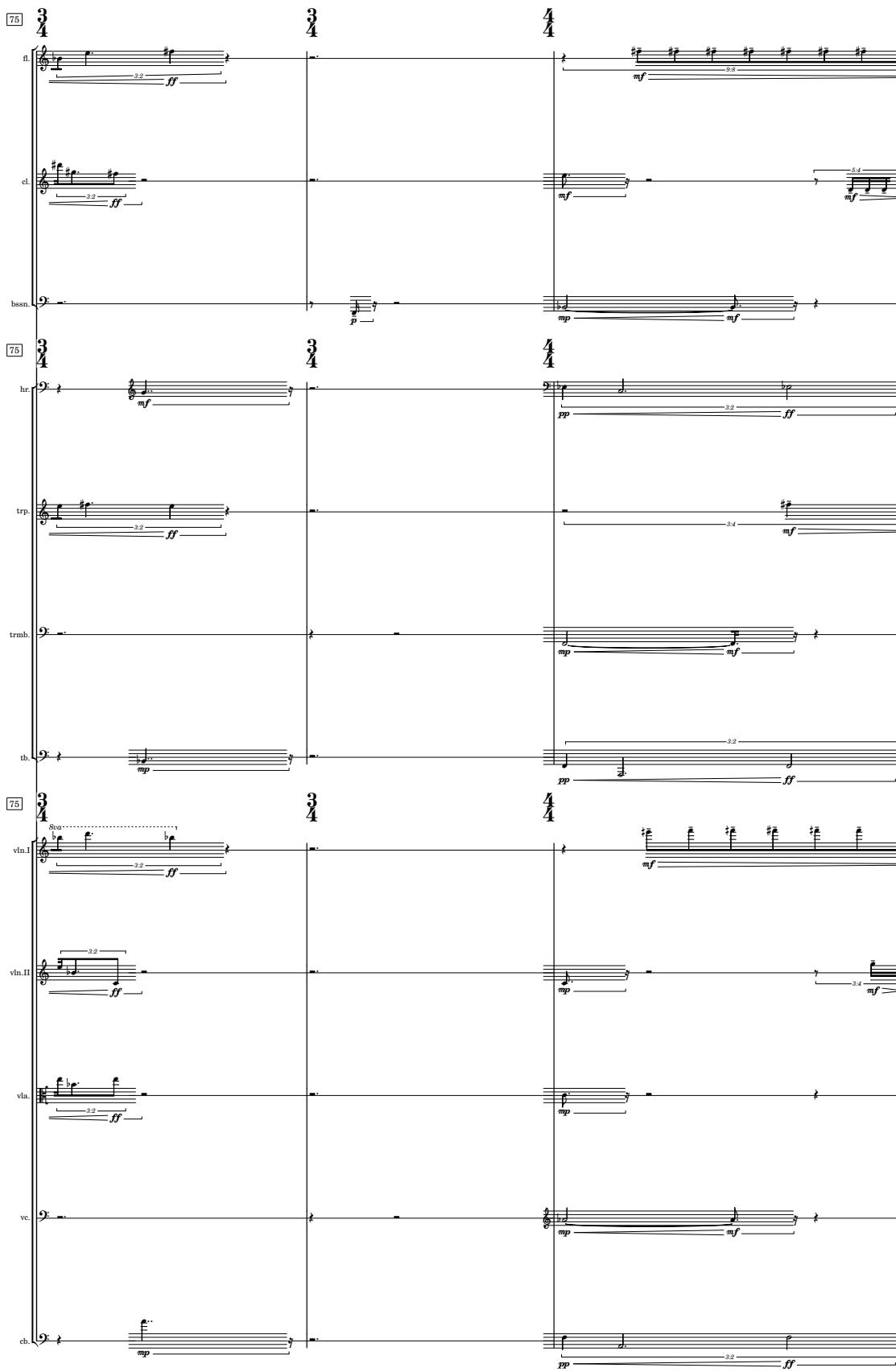
137 

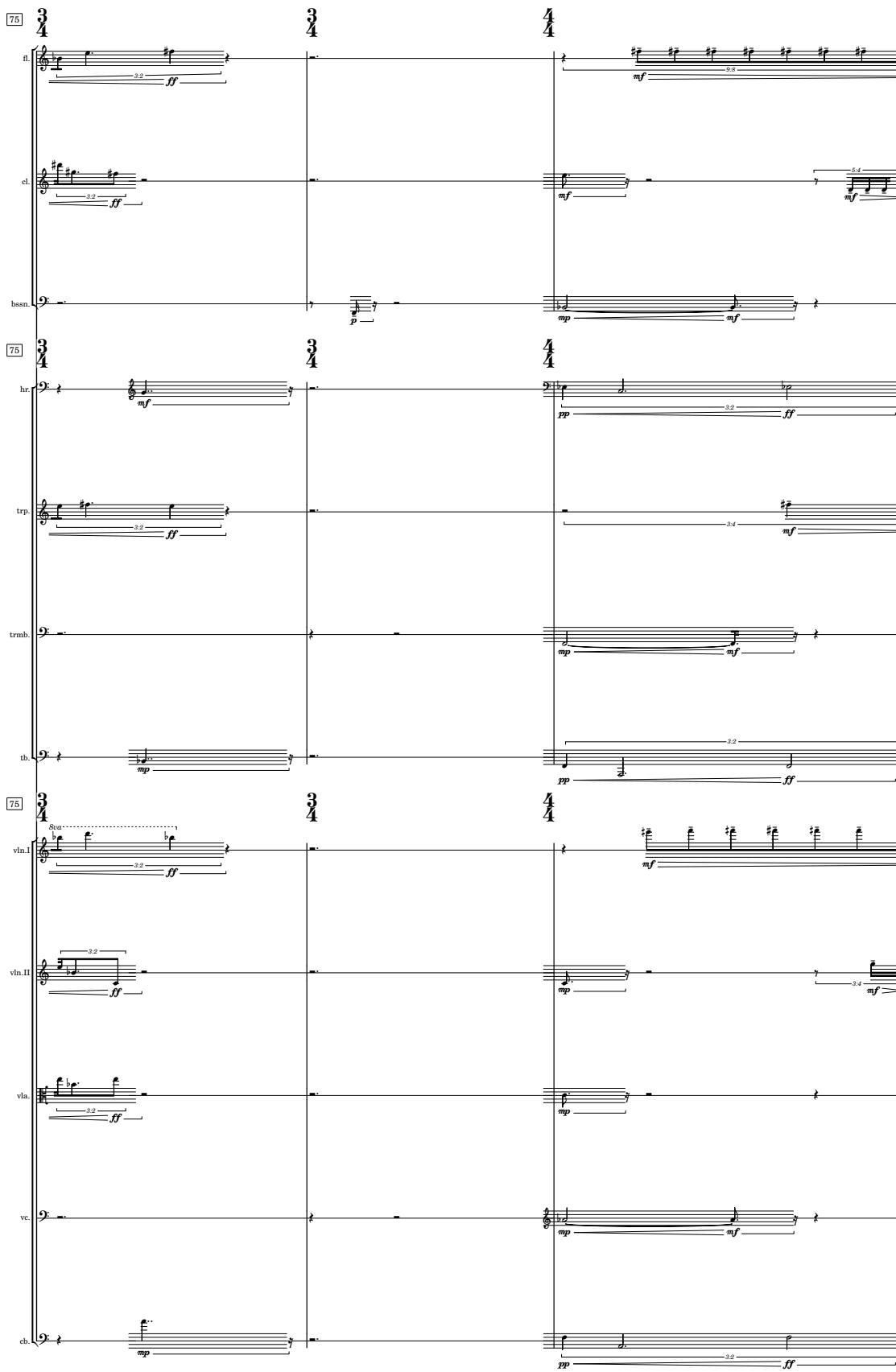
138 

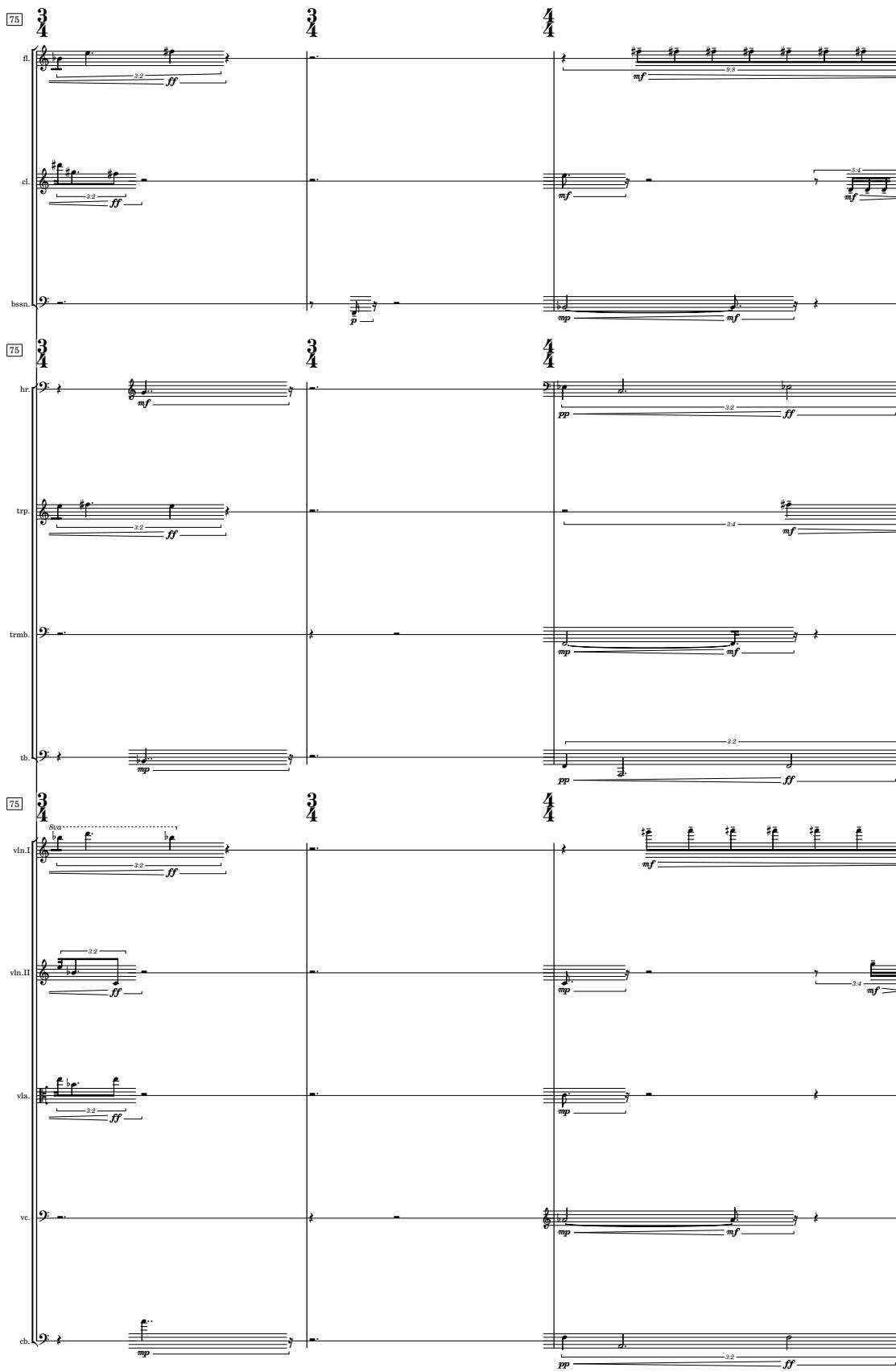
139 

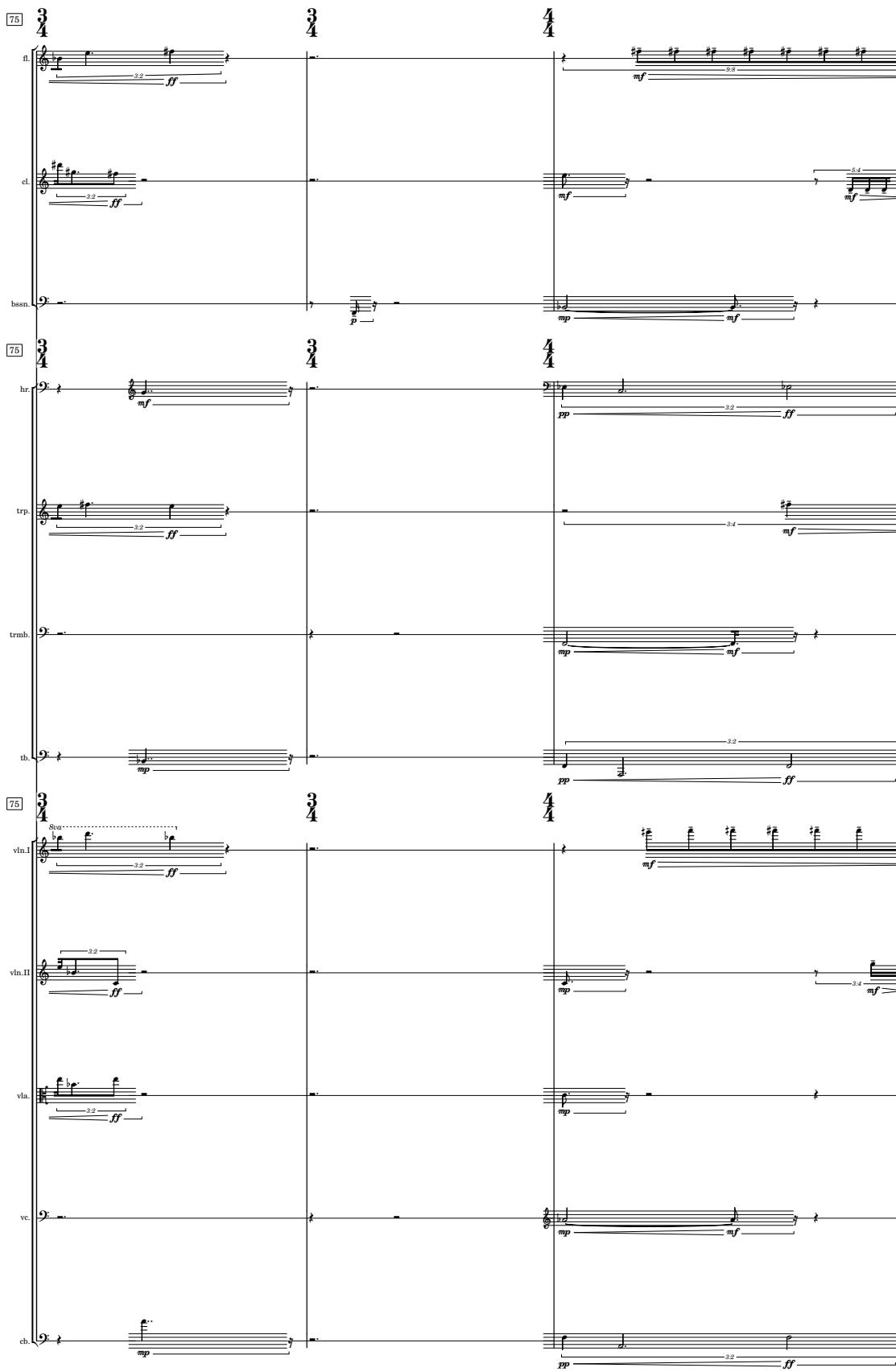
140 

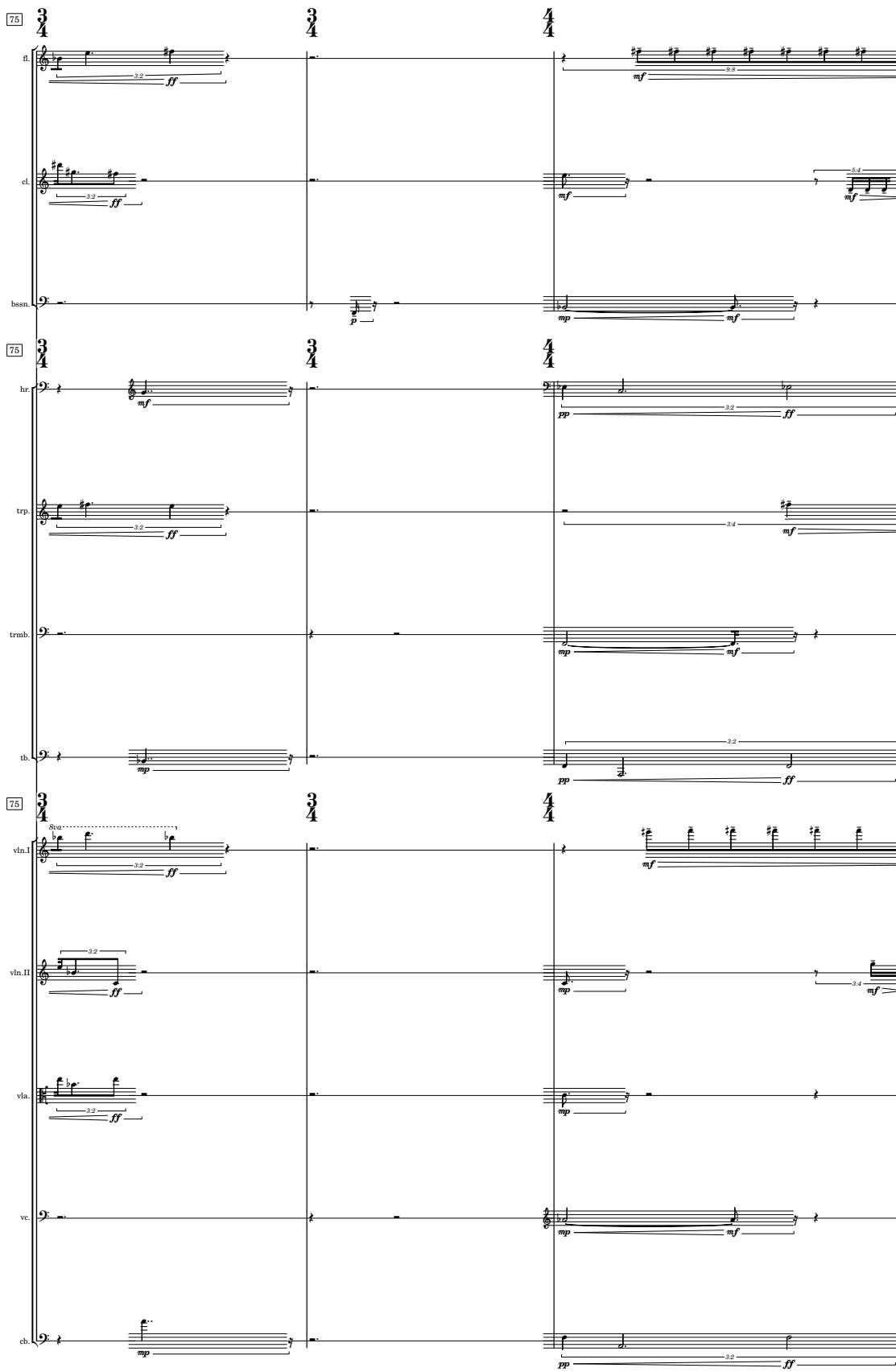
141 

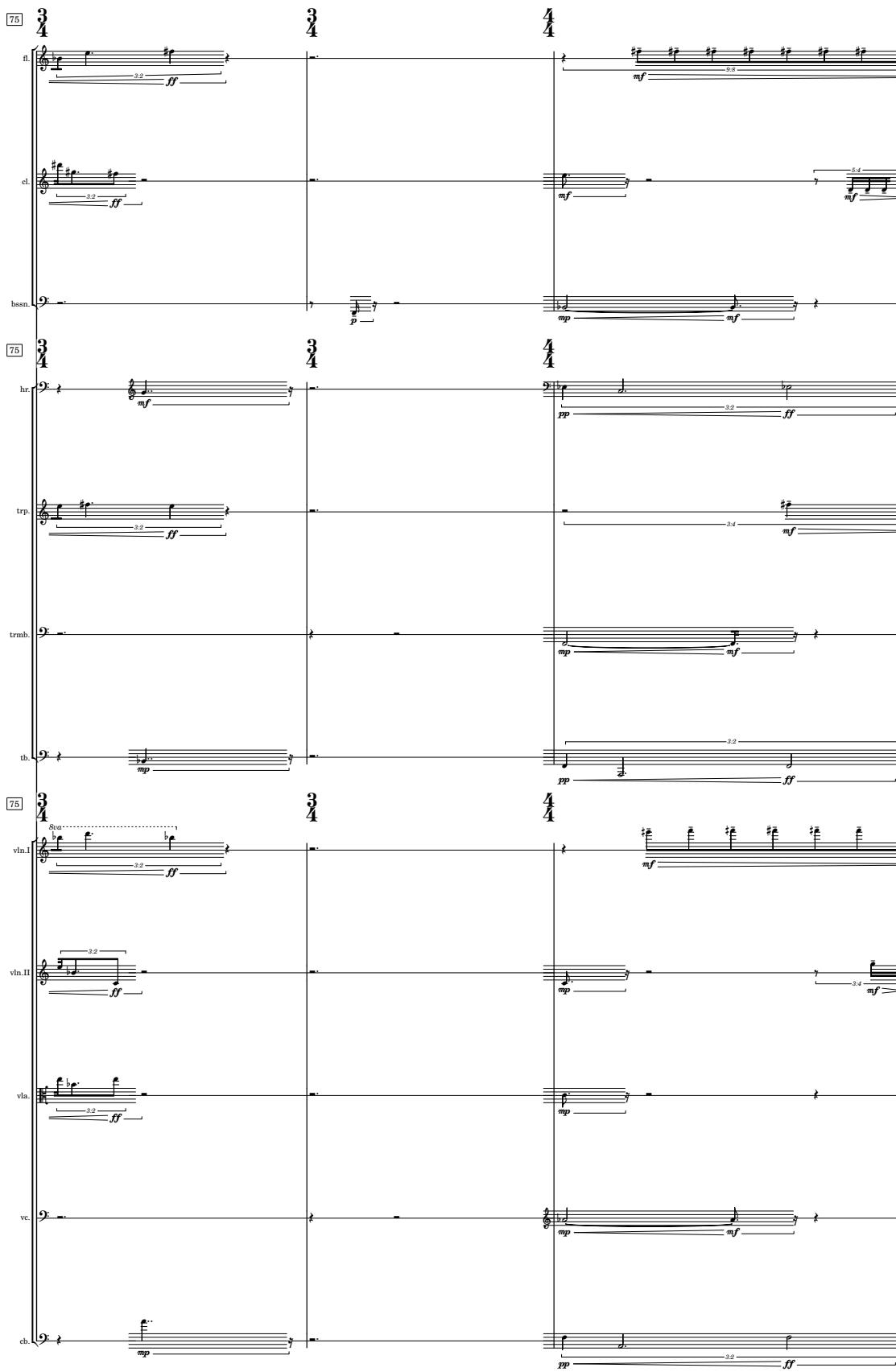
142 

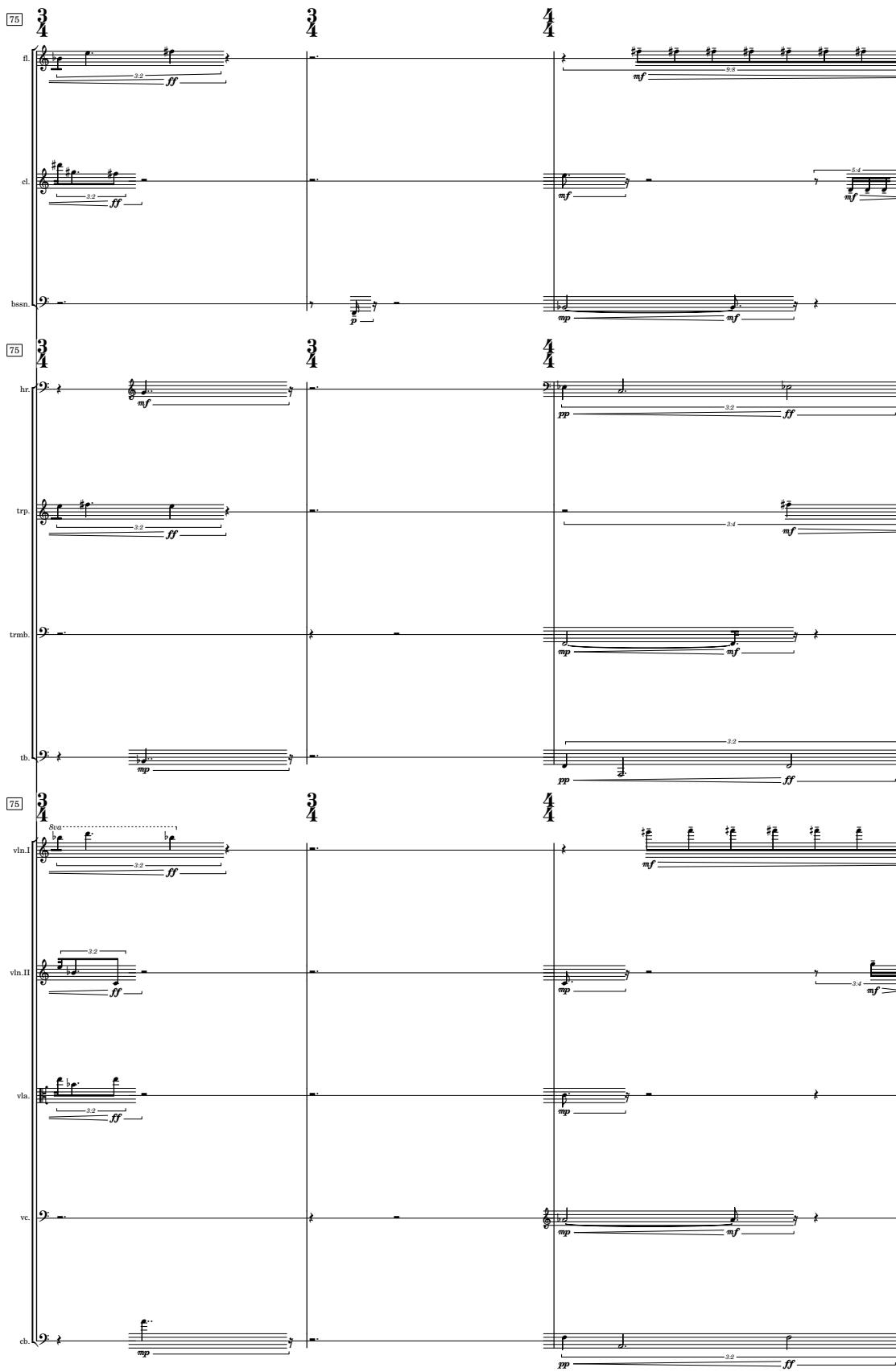
143 

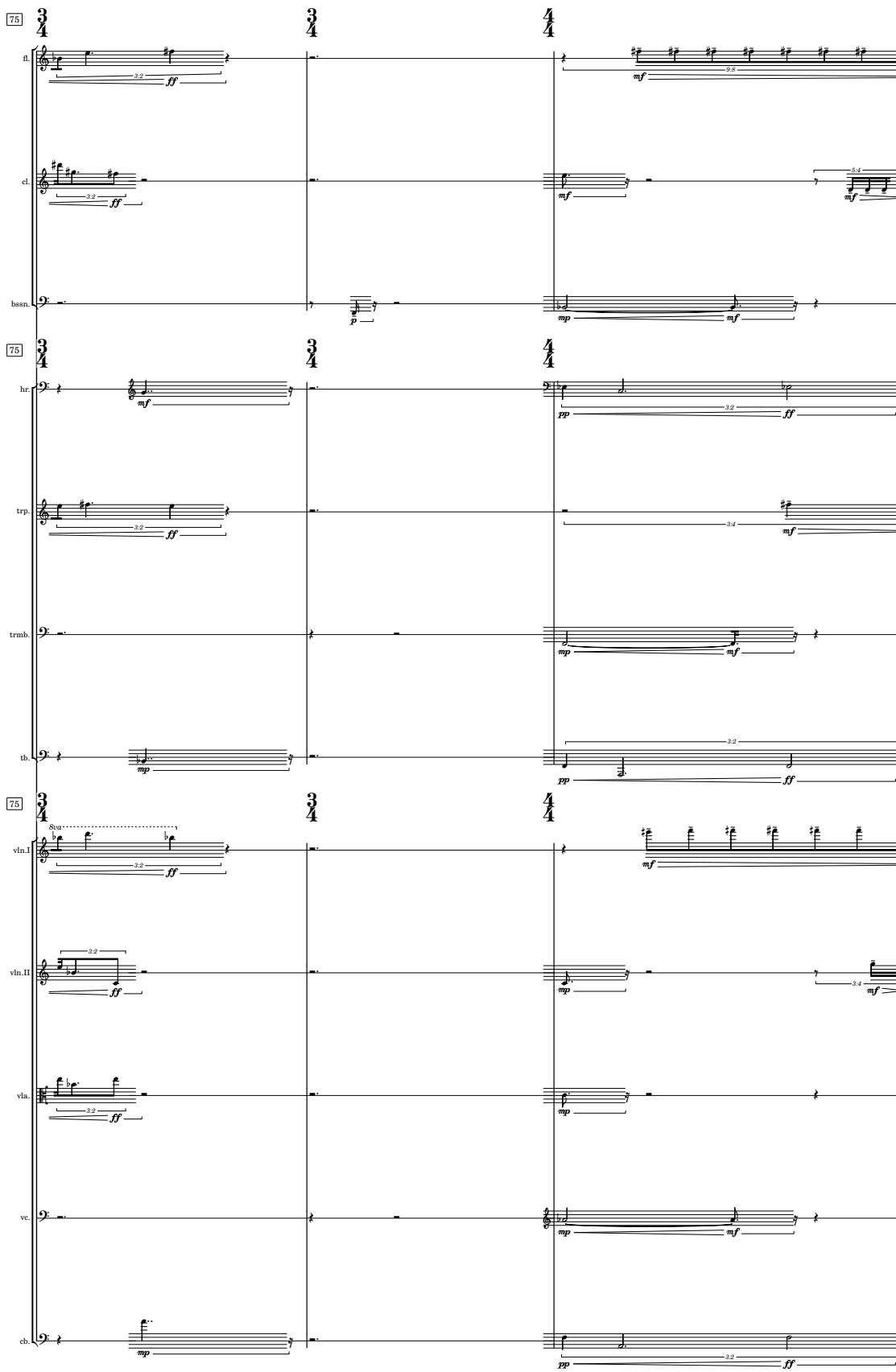
144 

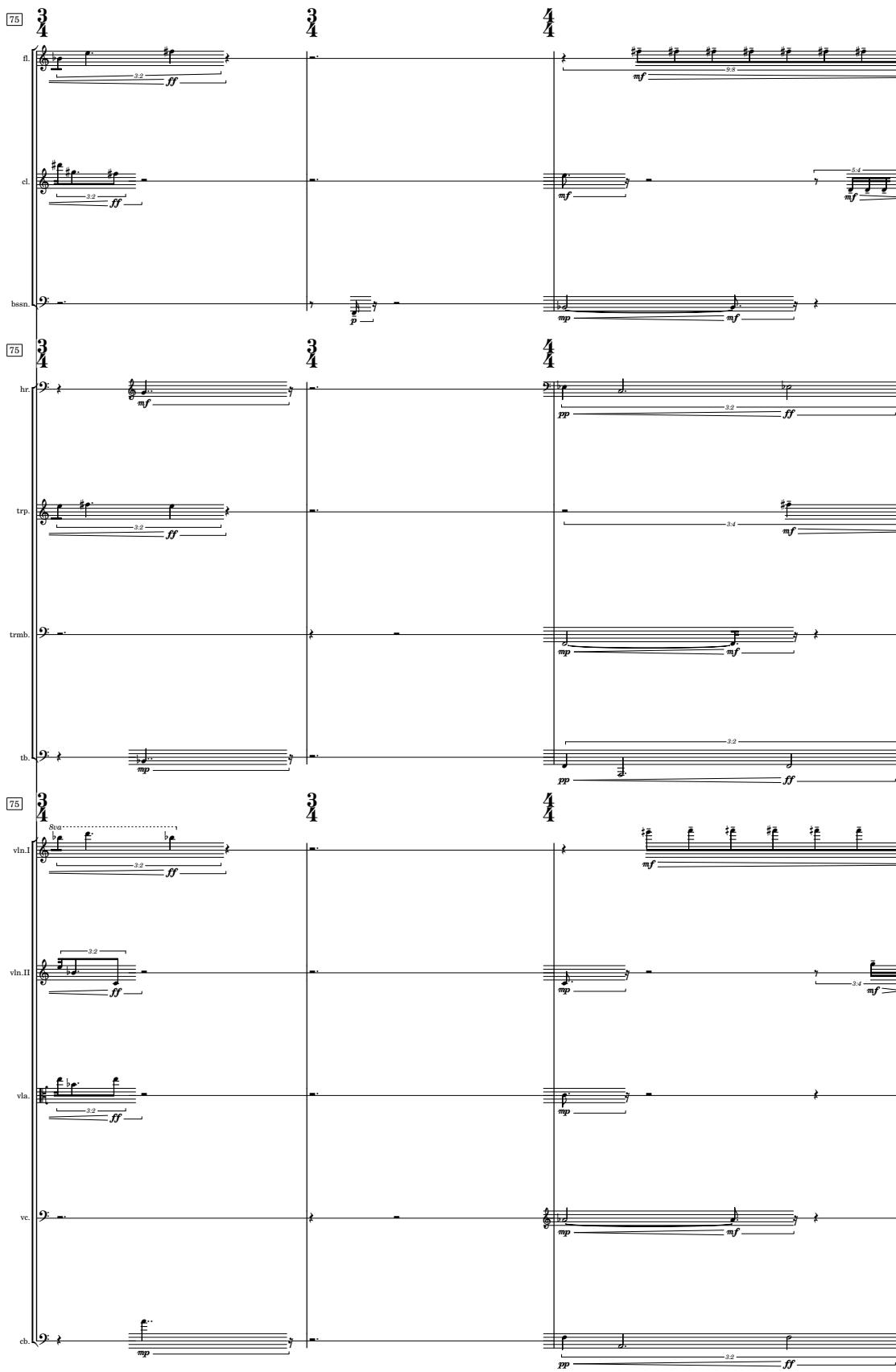
145 

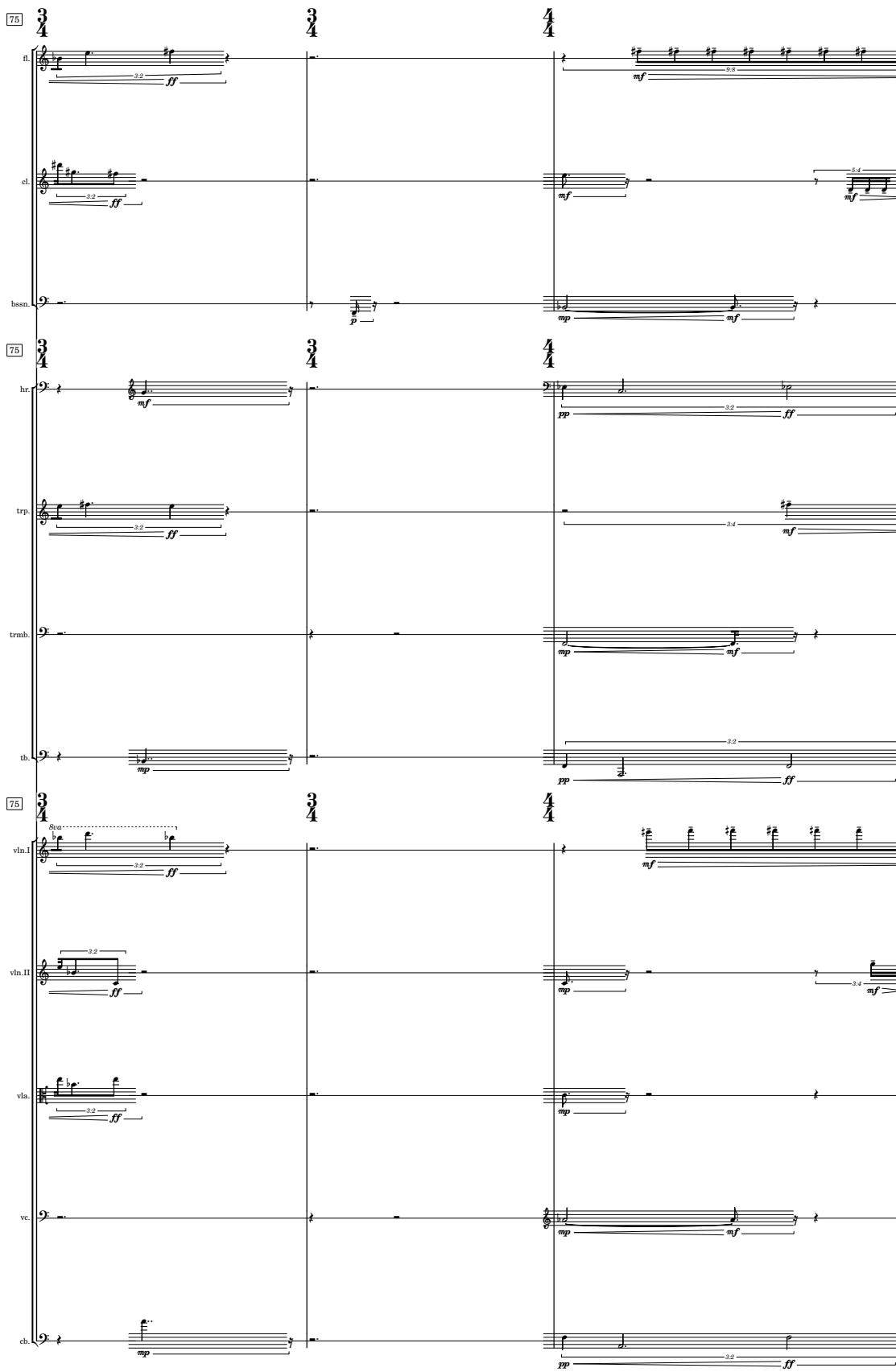
146 

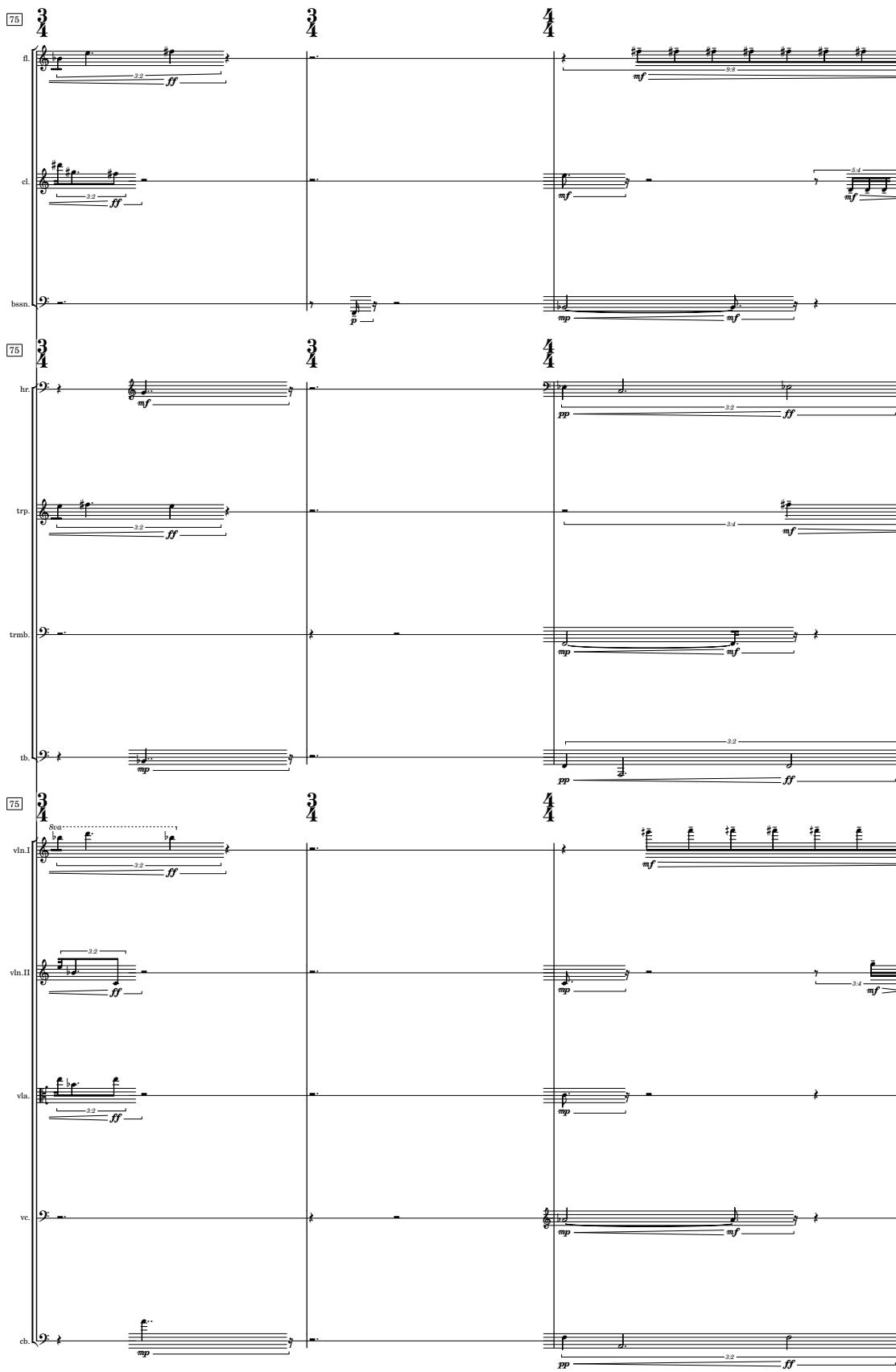
147 

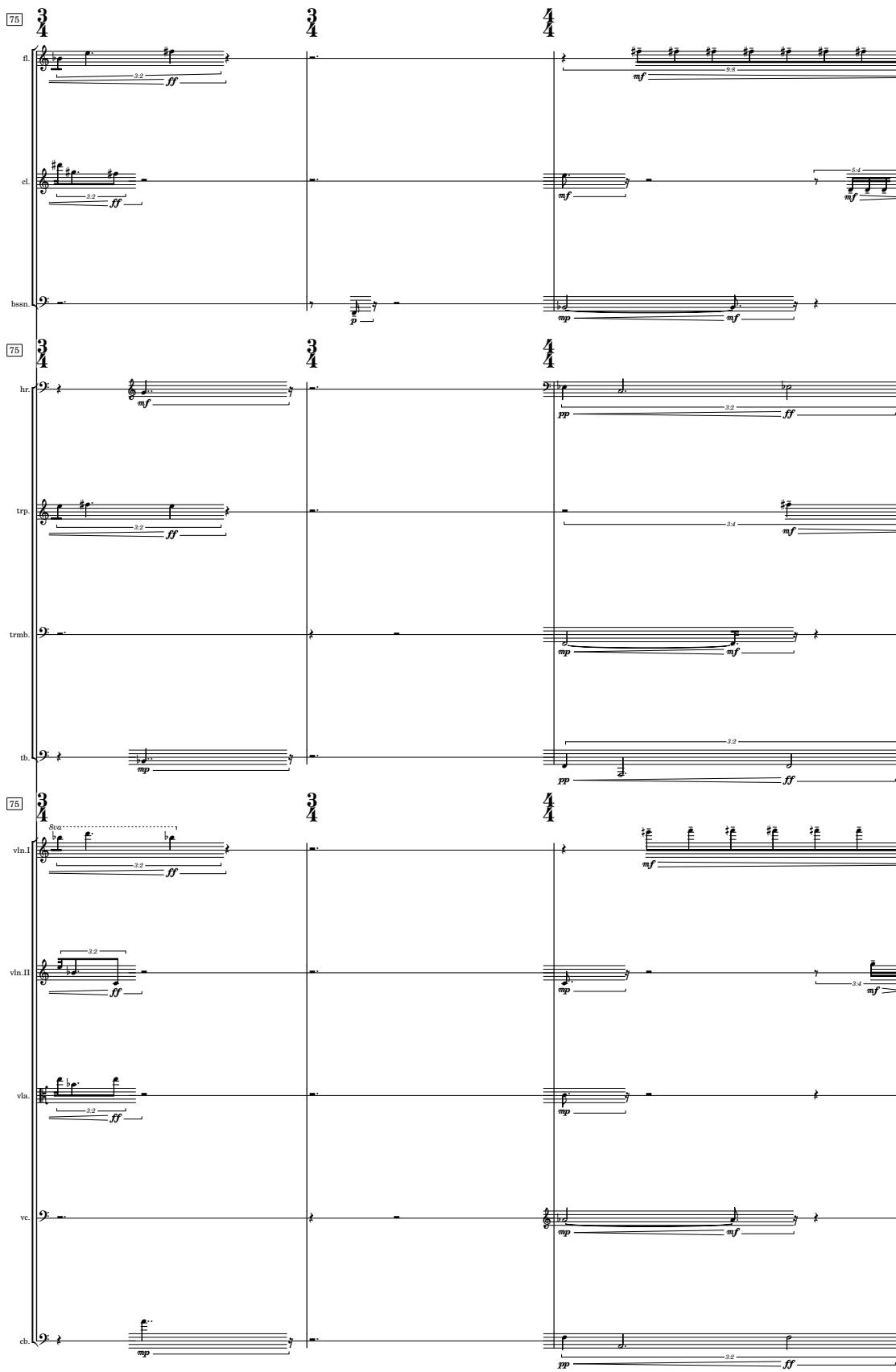
148 

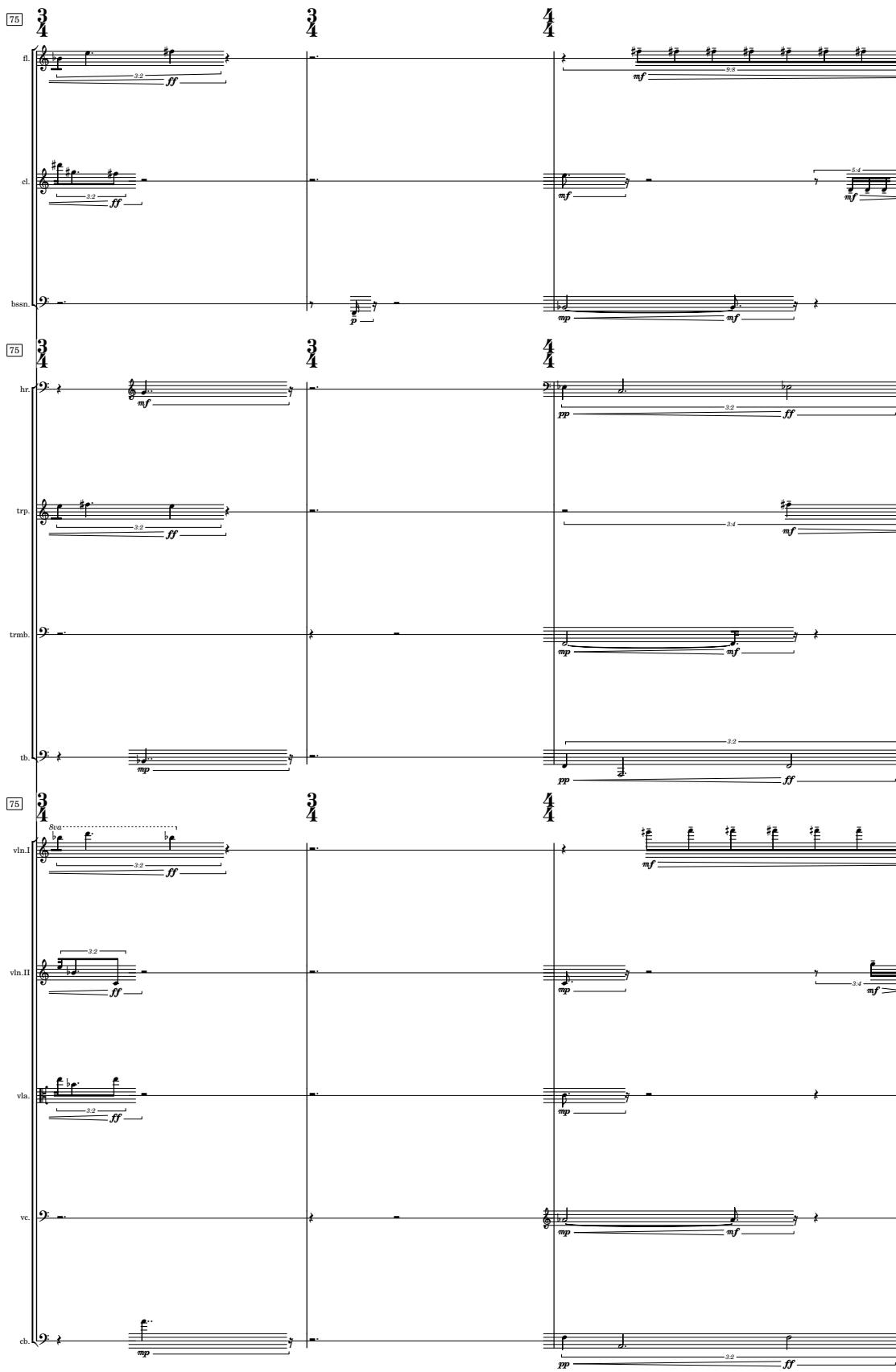
149 

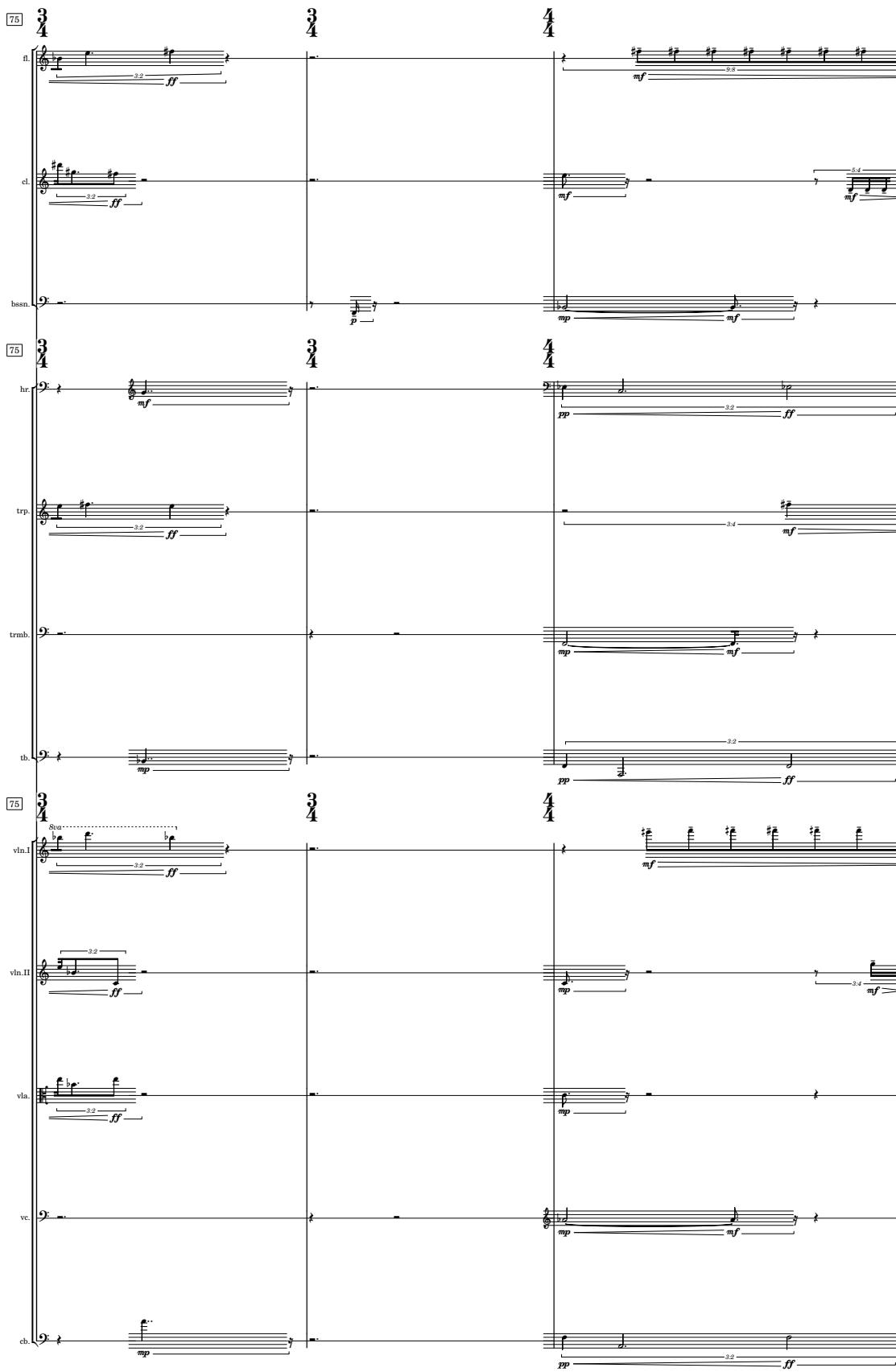
150 

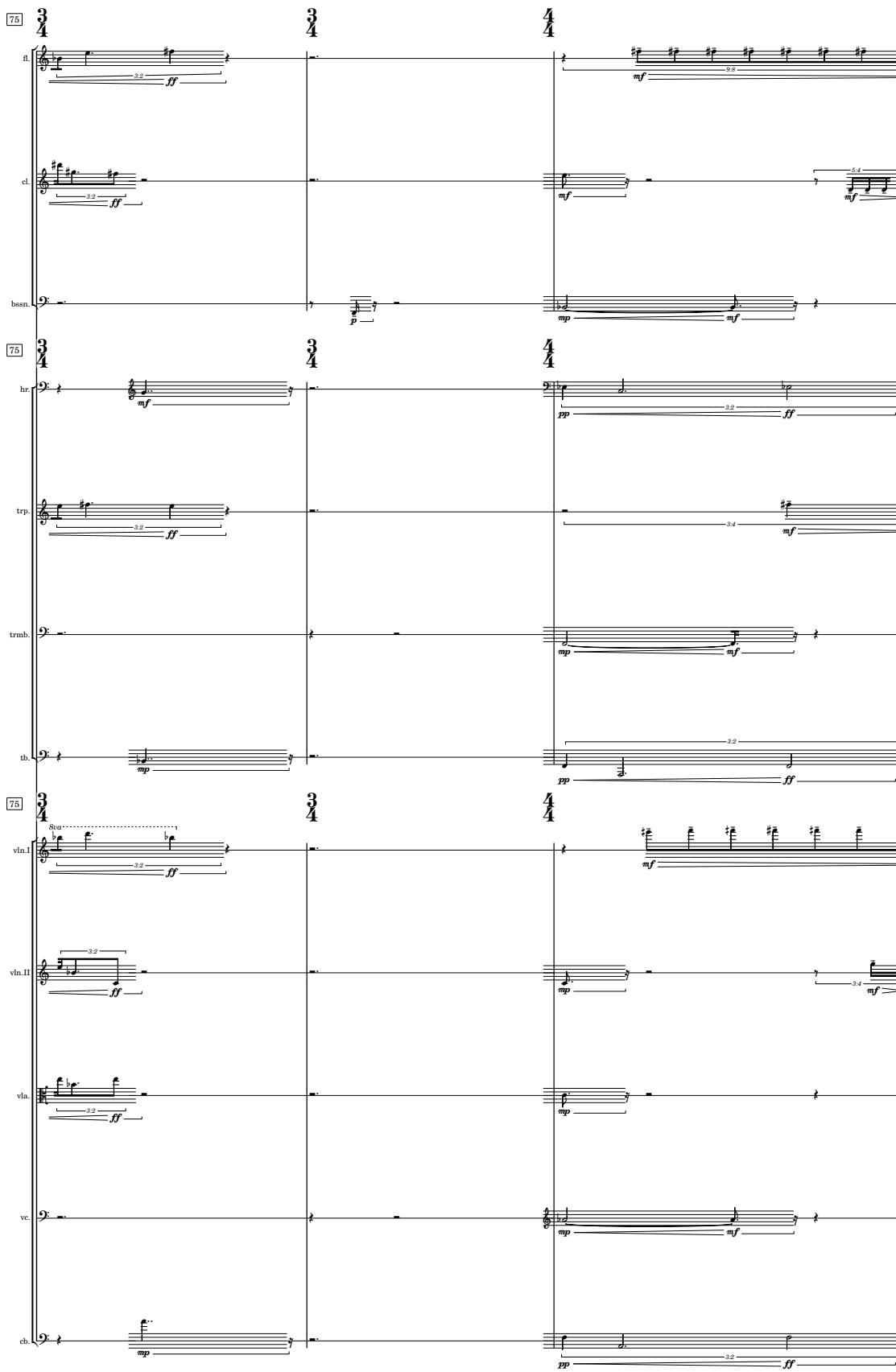
151 

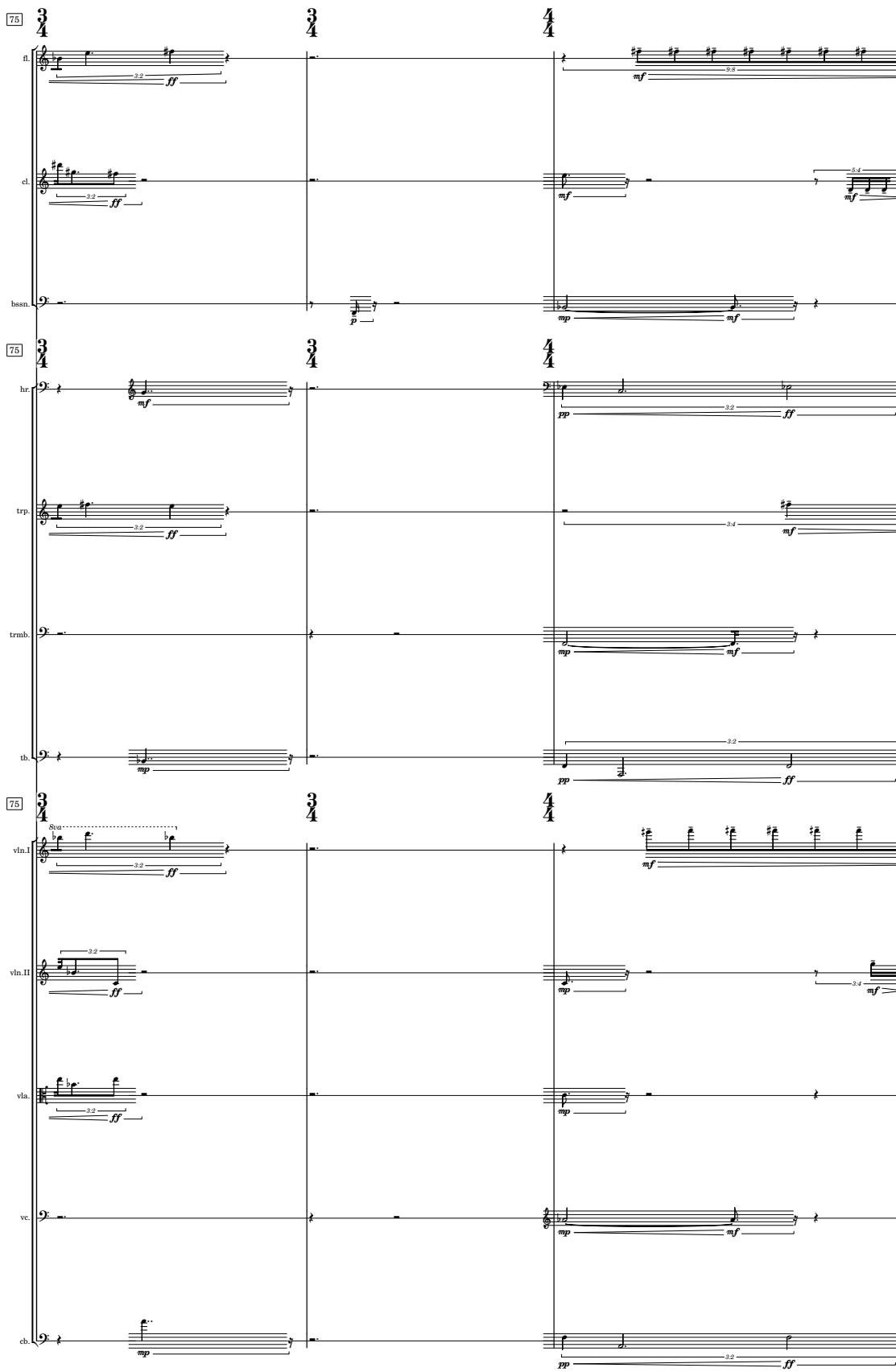
152 

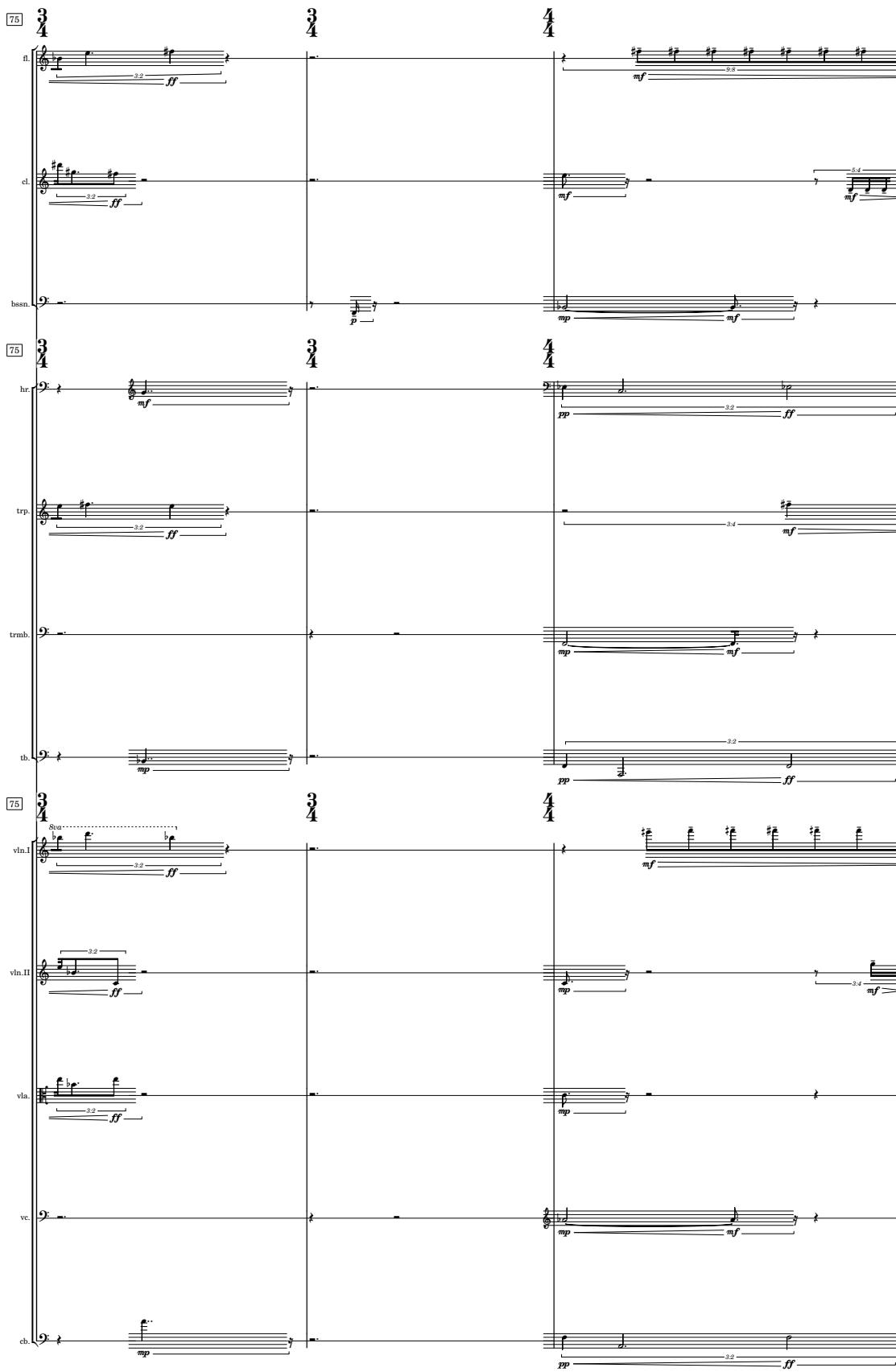
153 

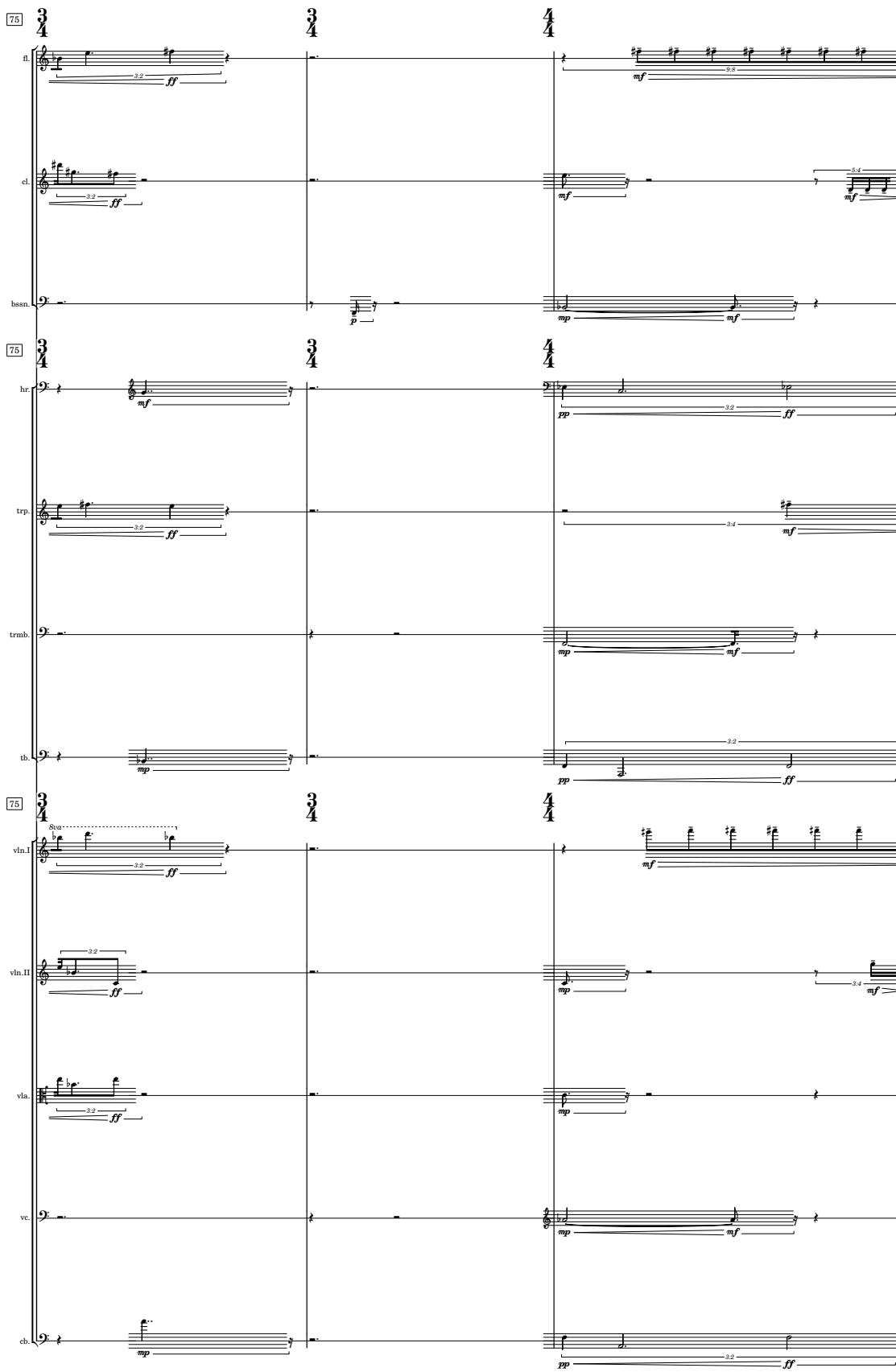
154 

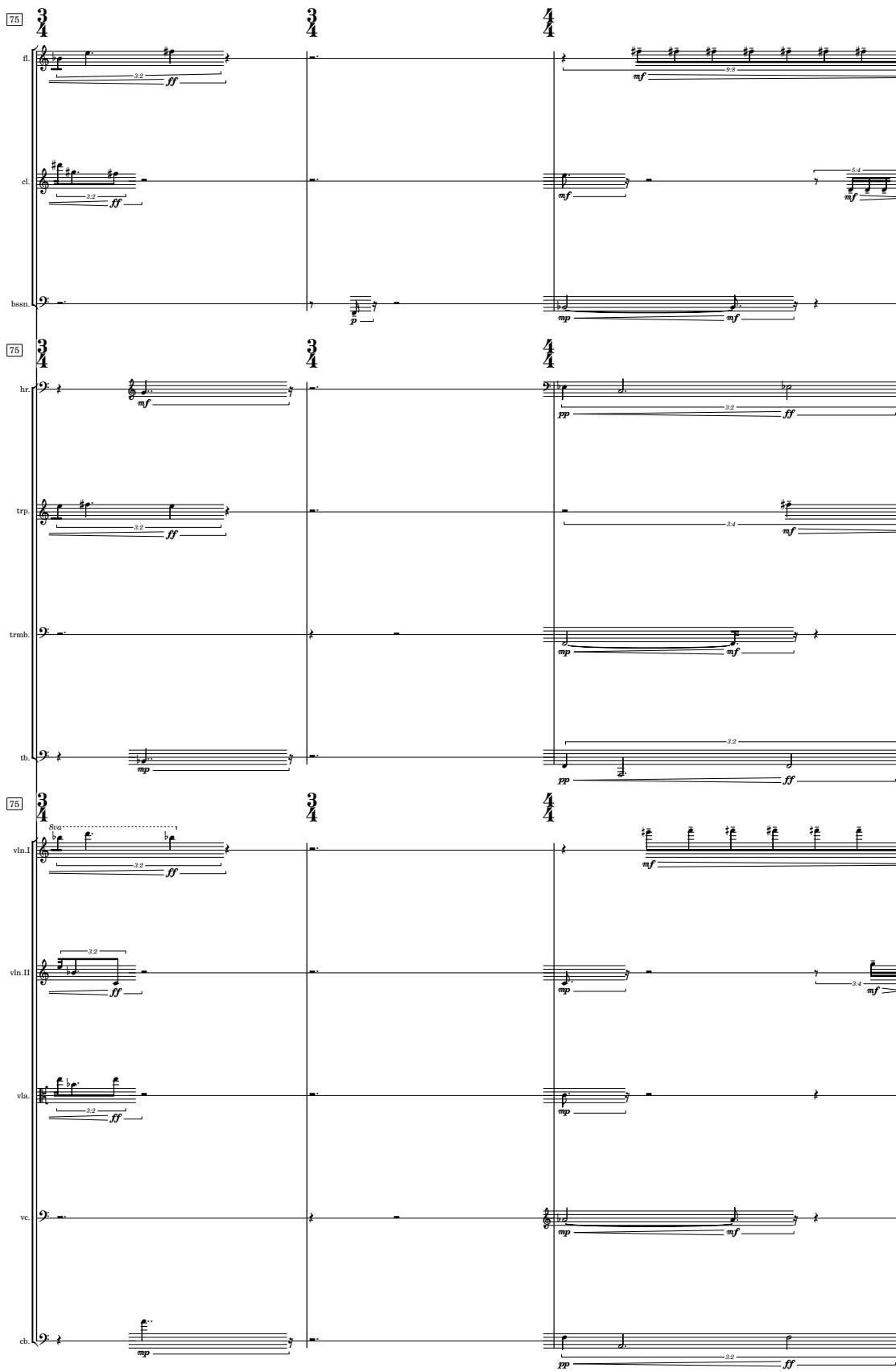
155 

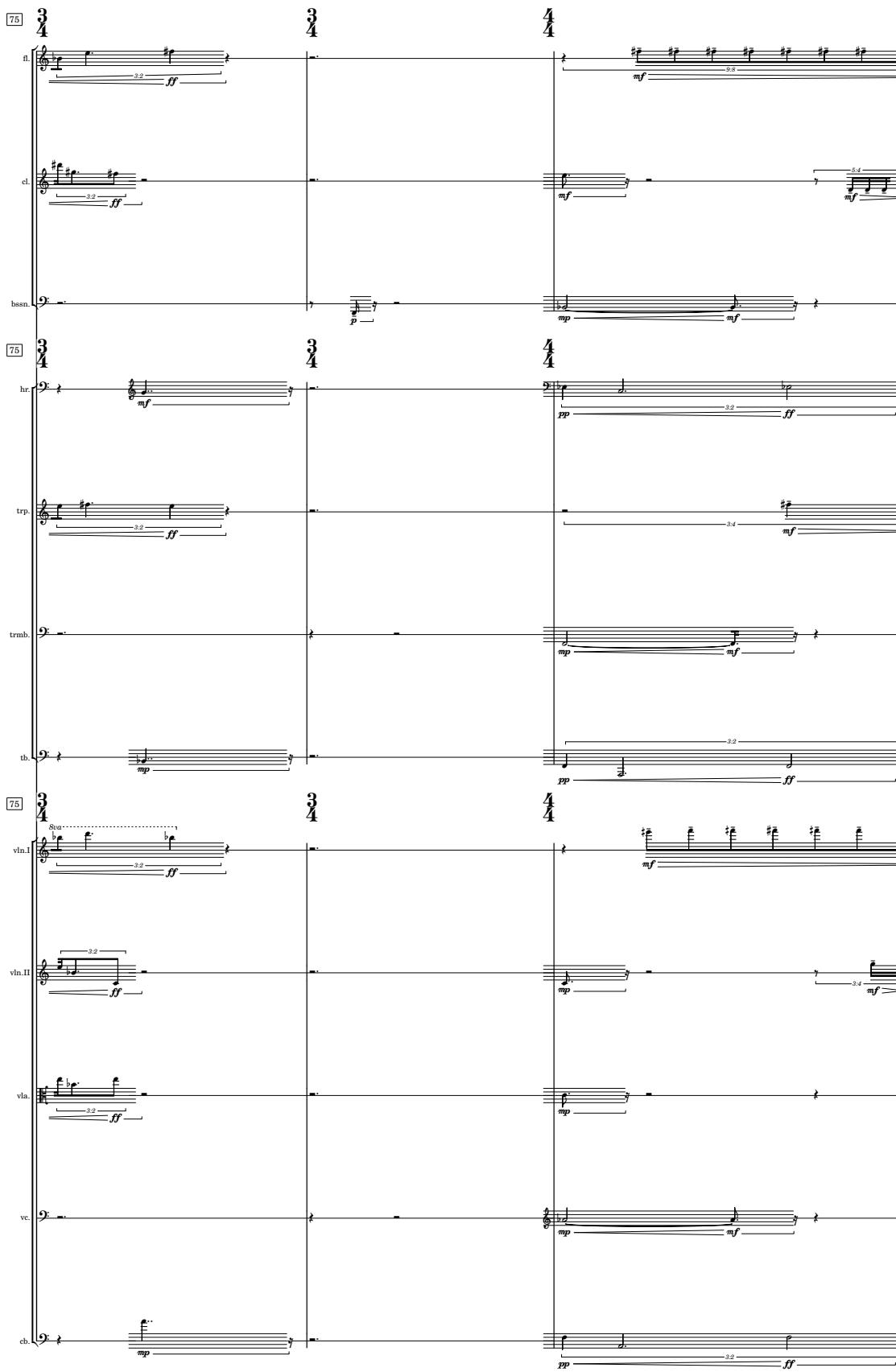
156 

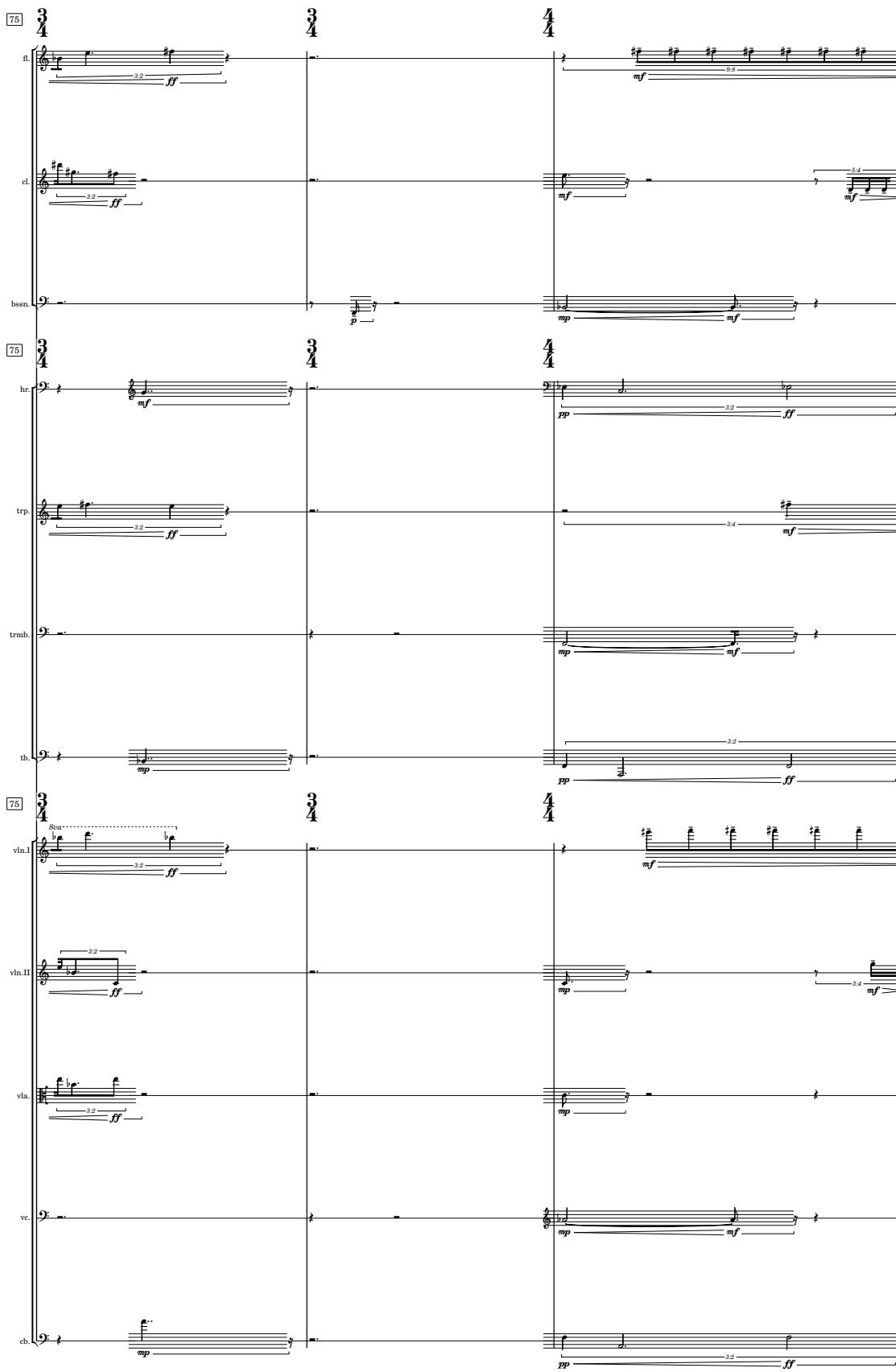
157 

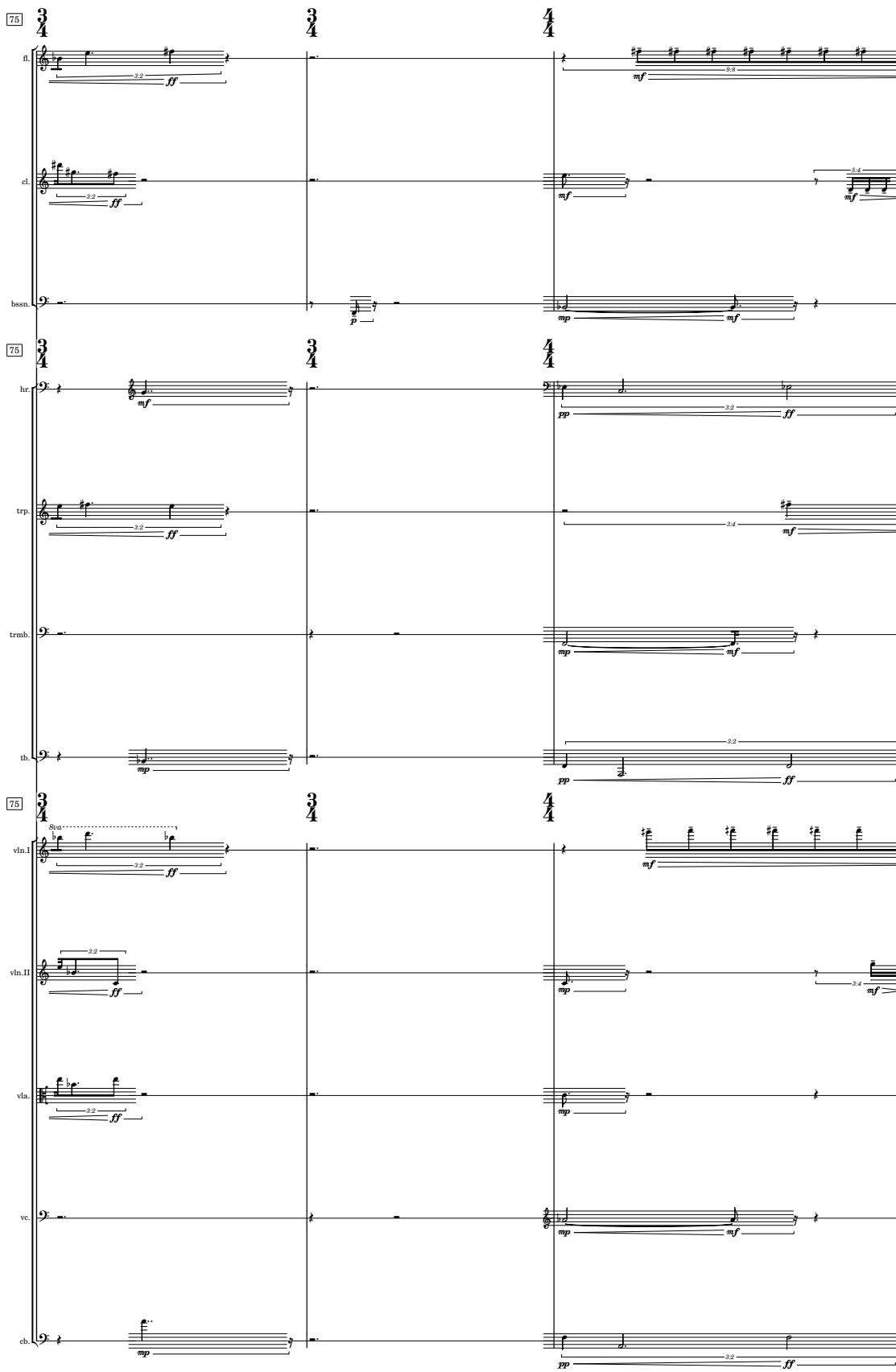
158 

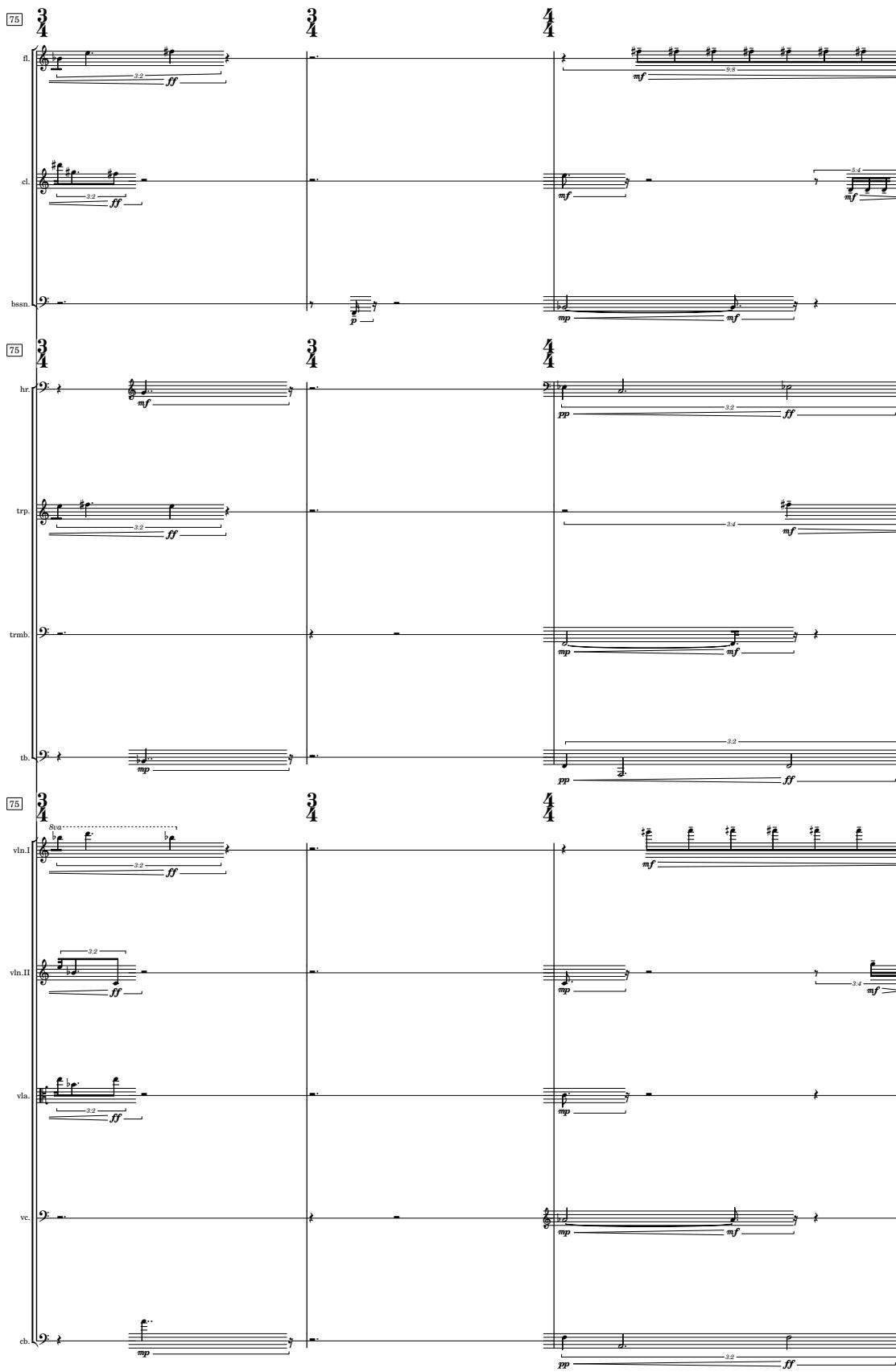
159 

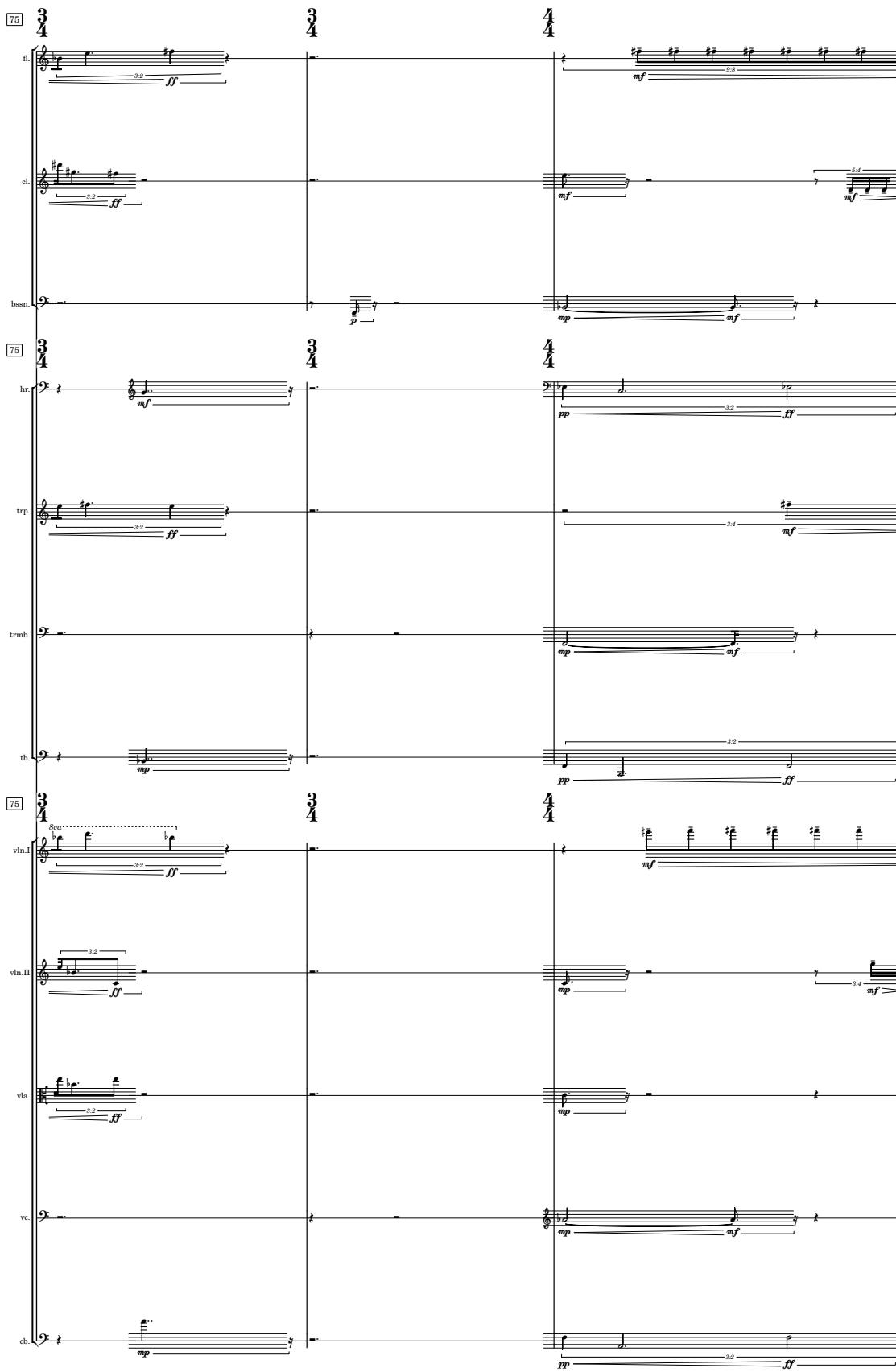
160 

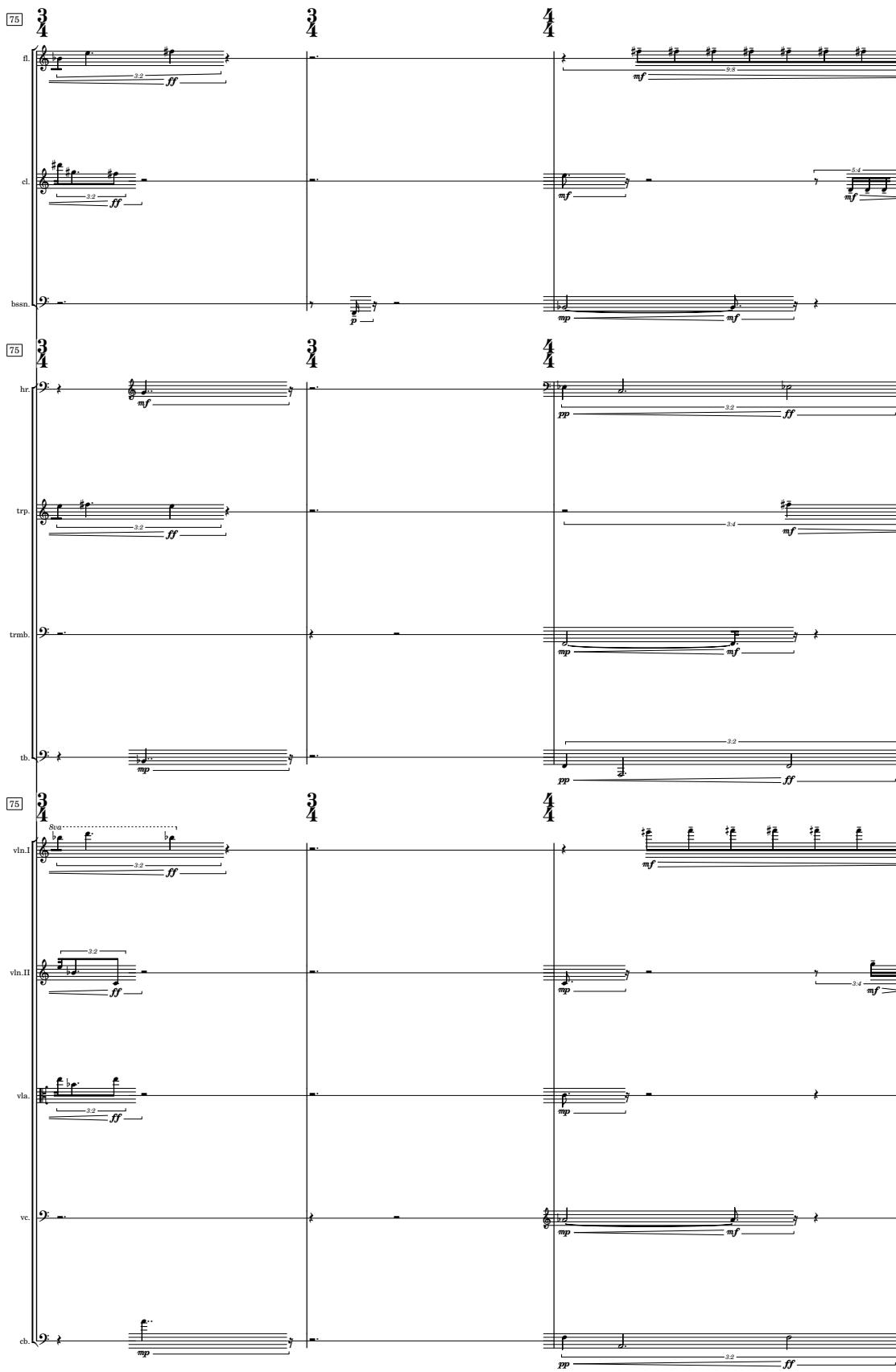
161 

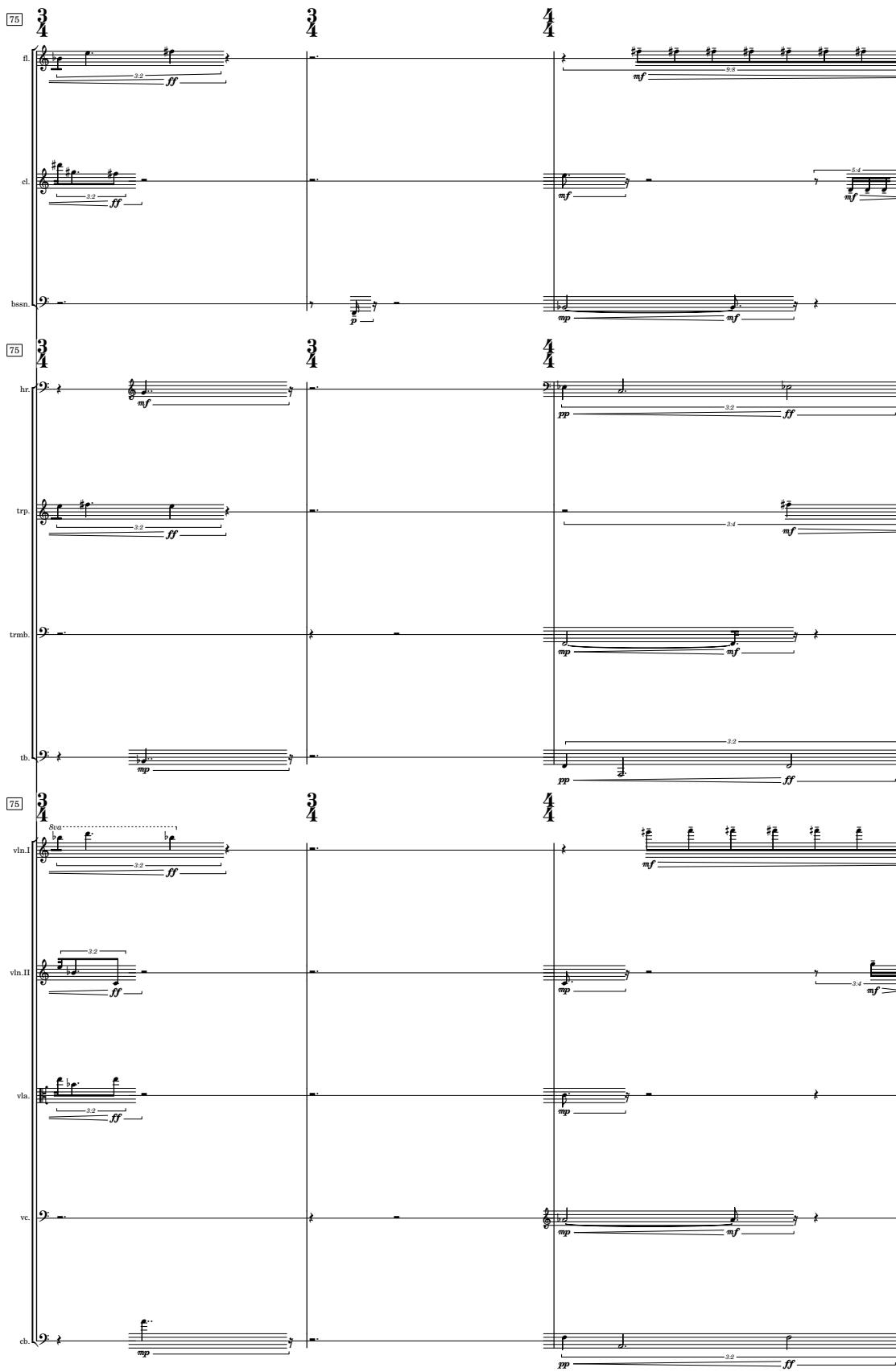
162 

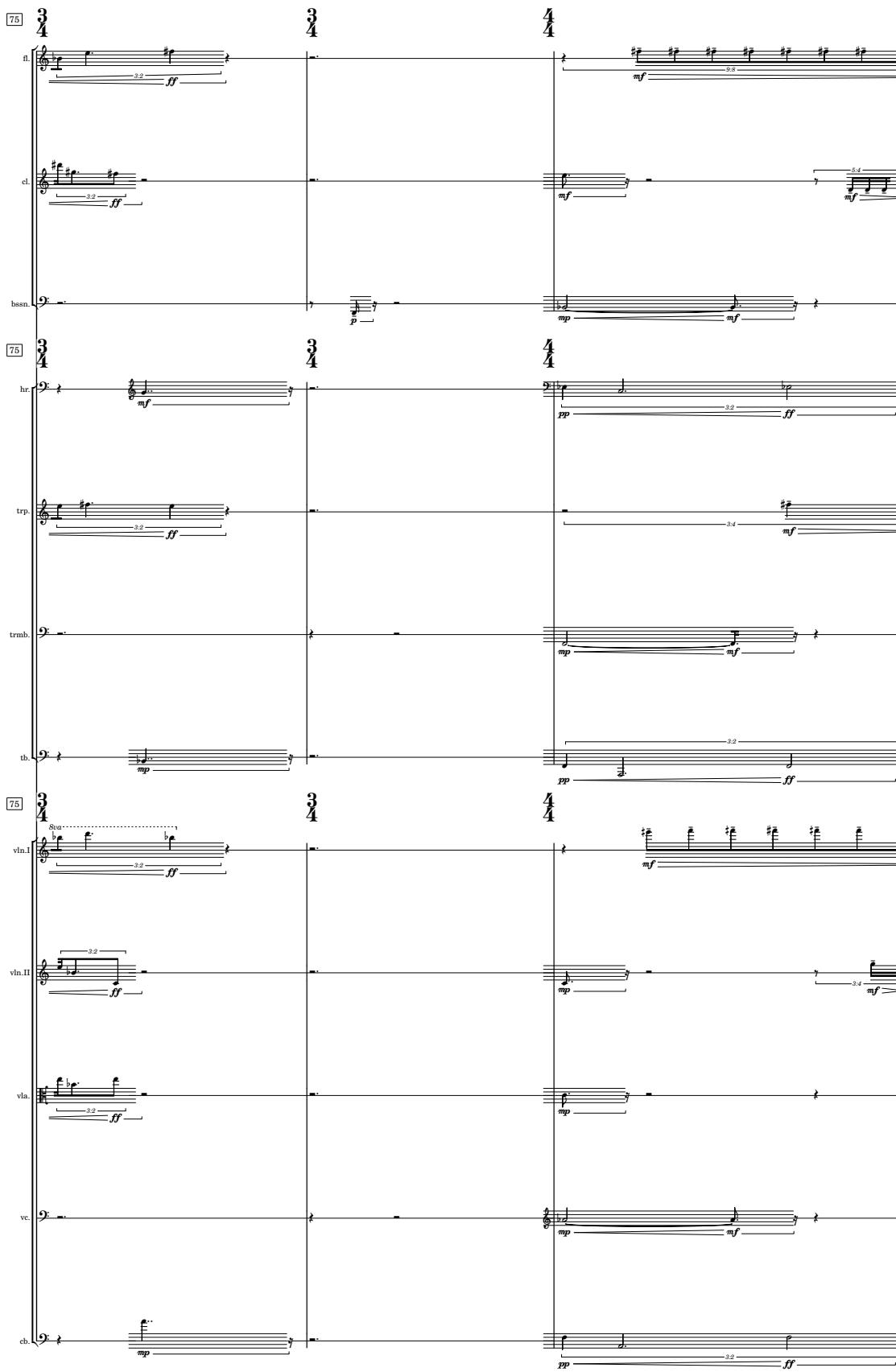
163 

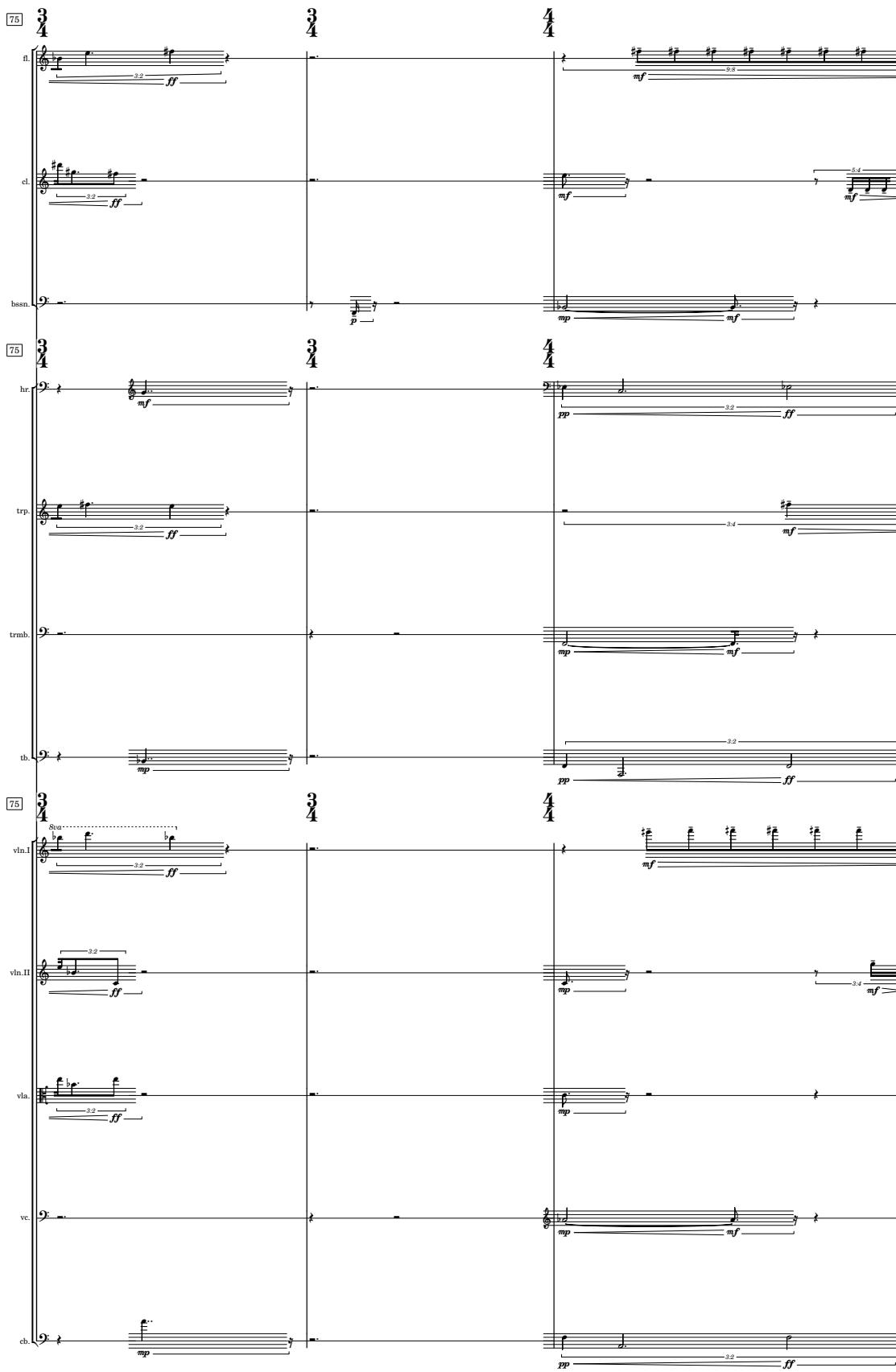
164 

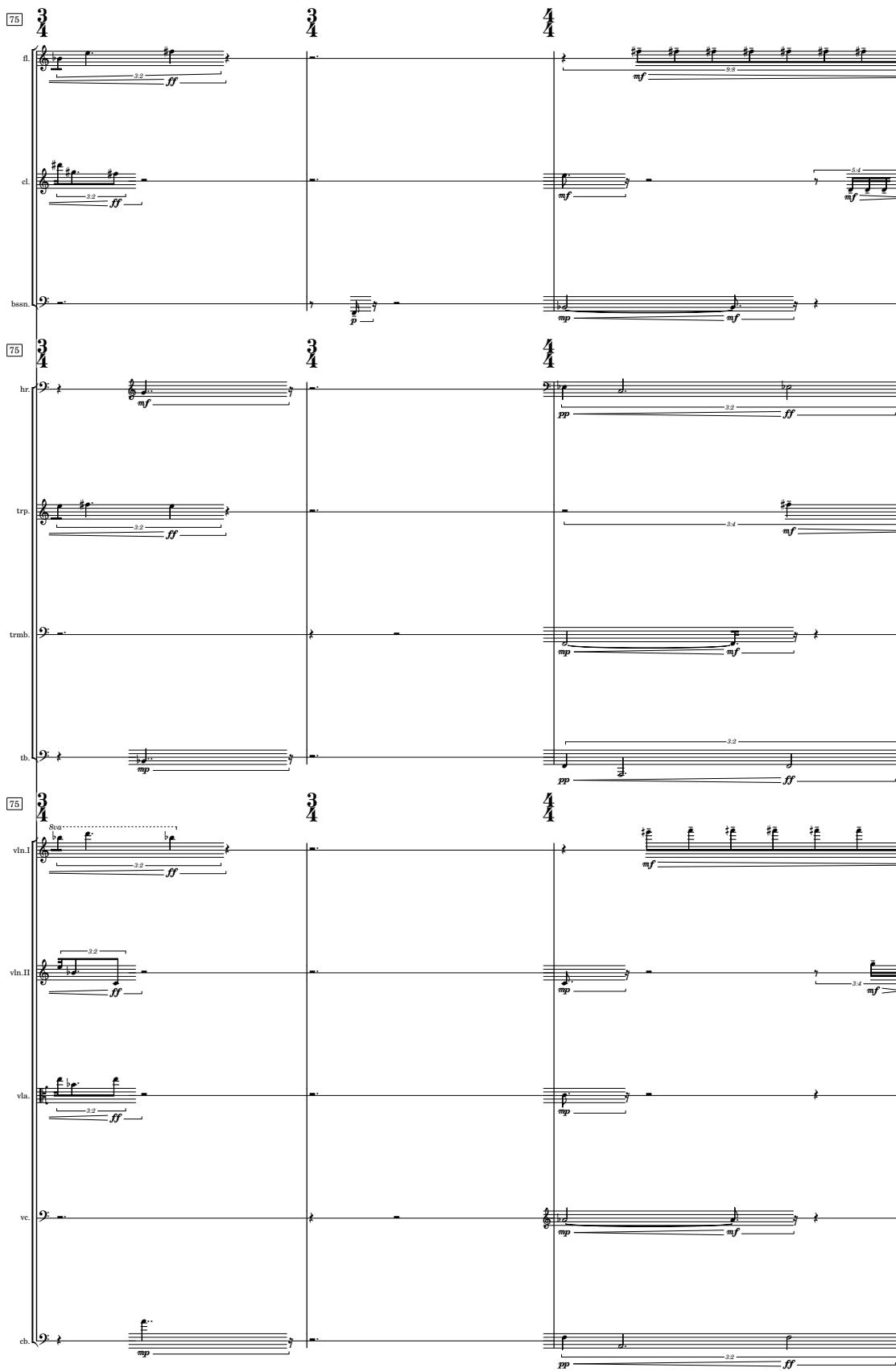
165 

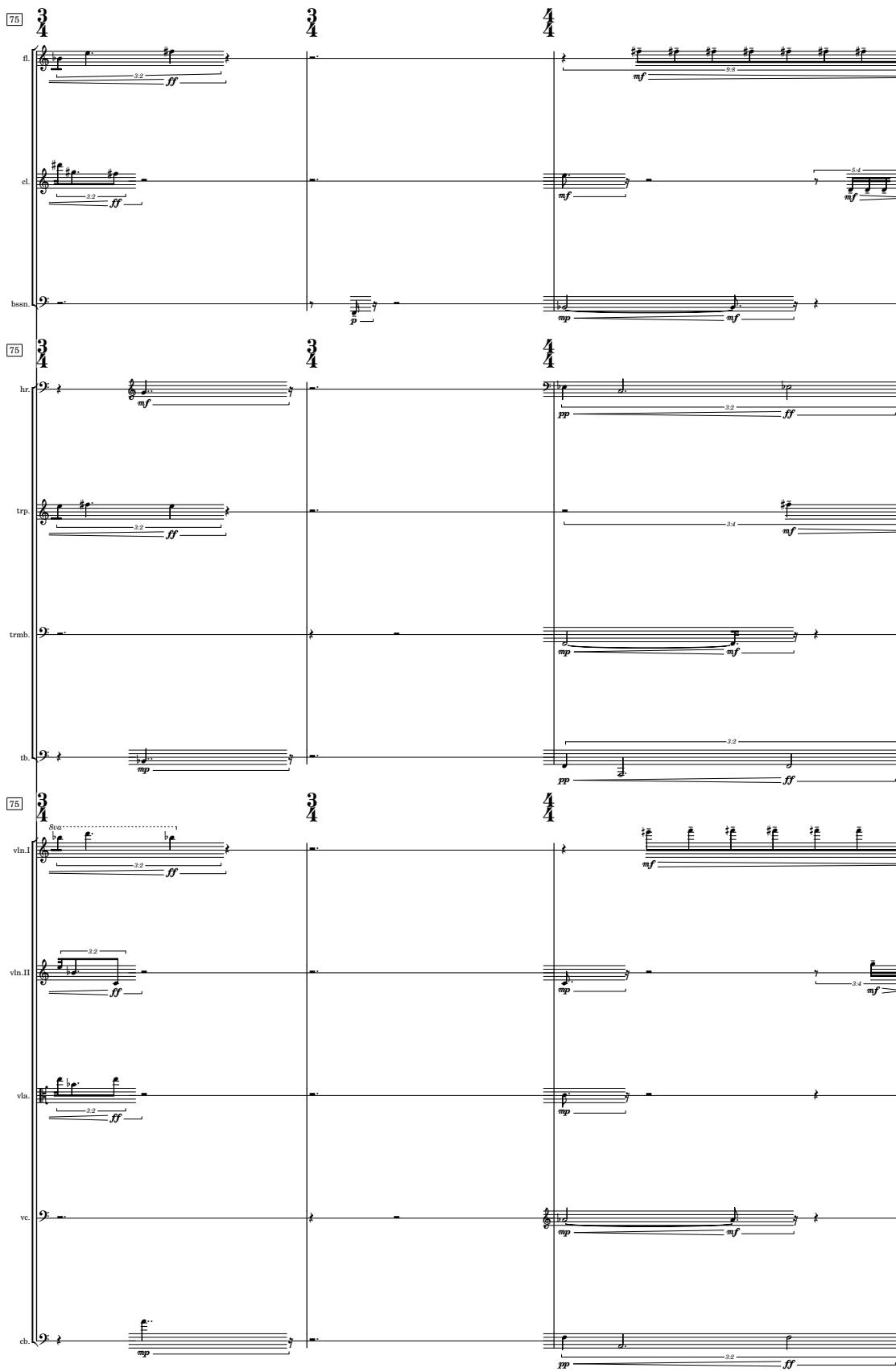
166 

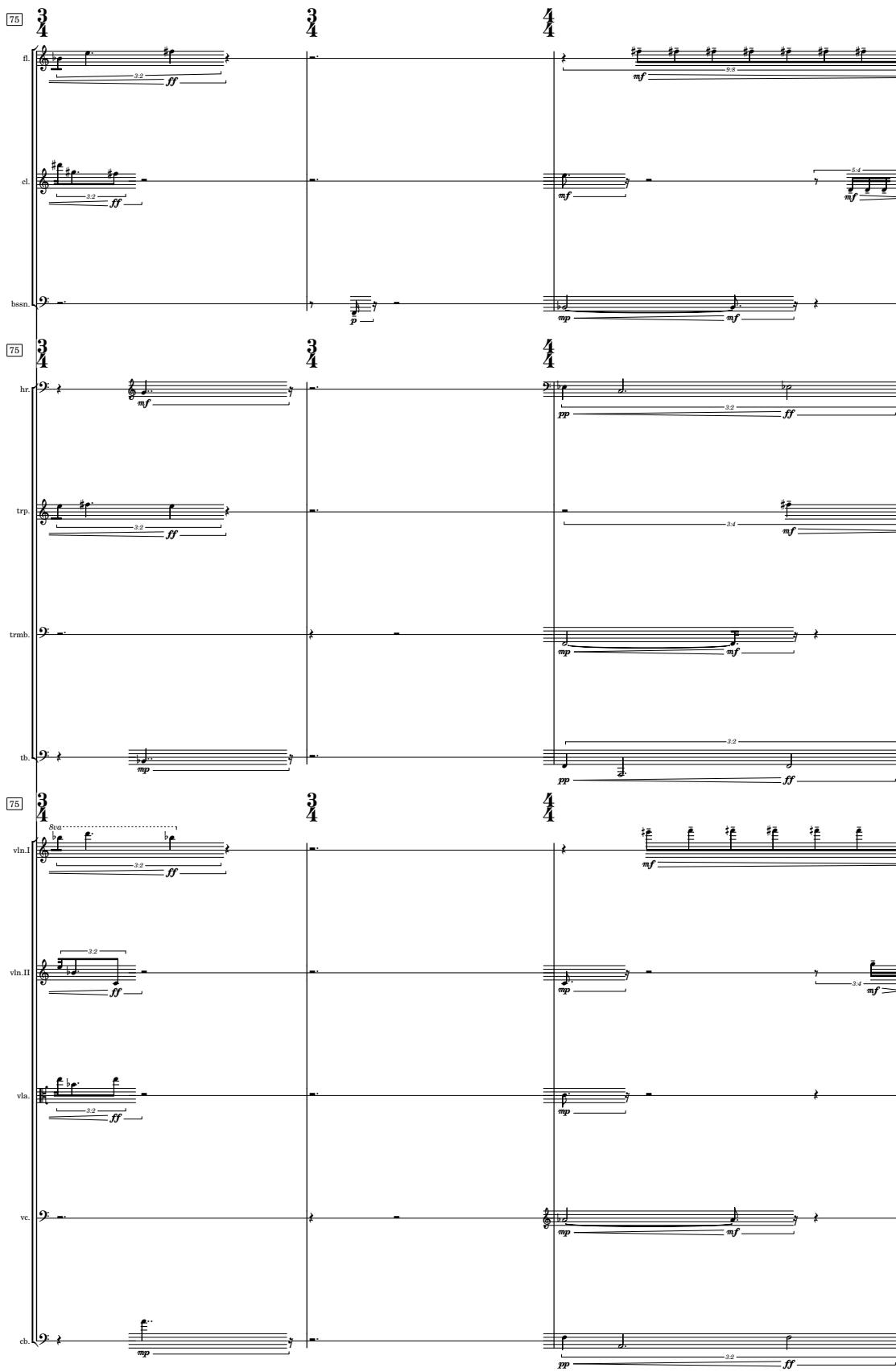
167 

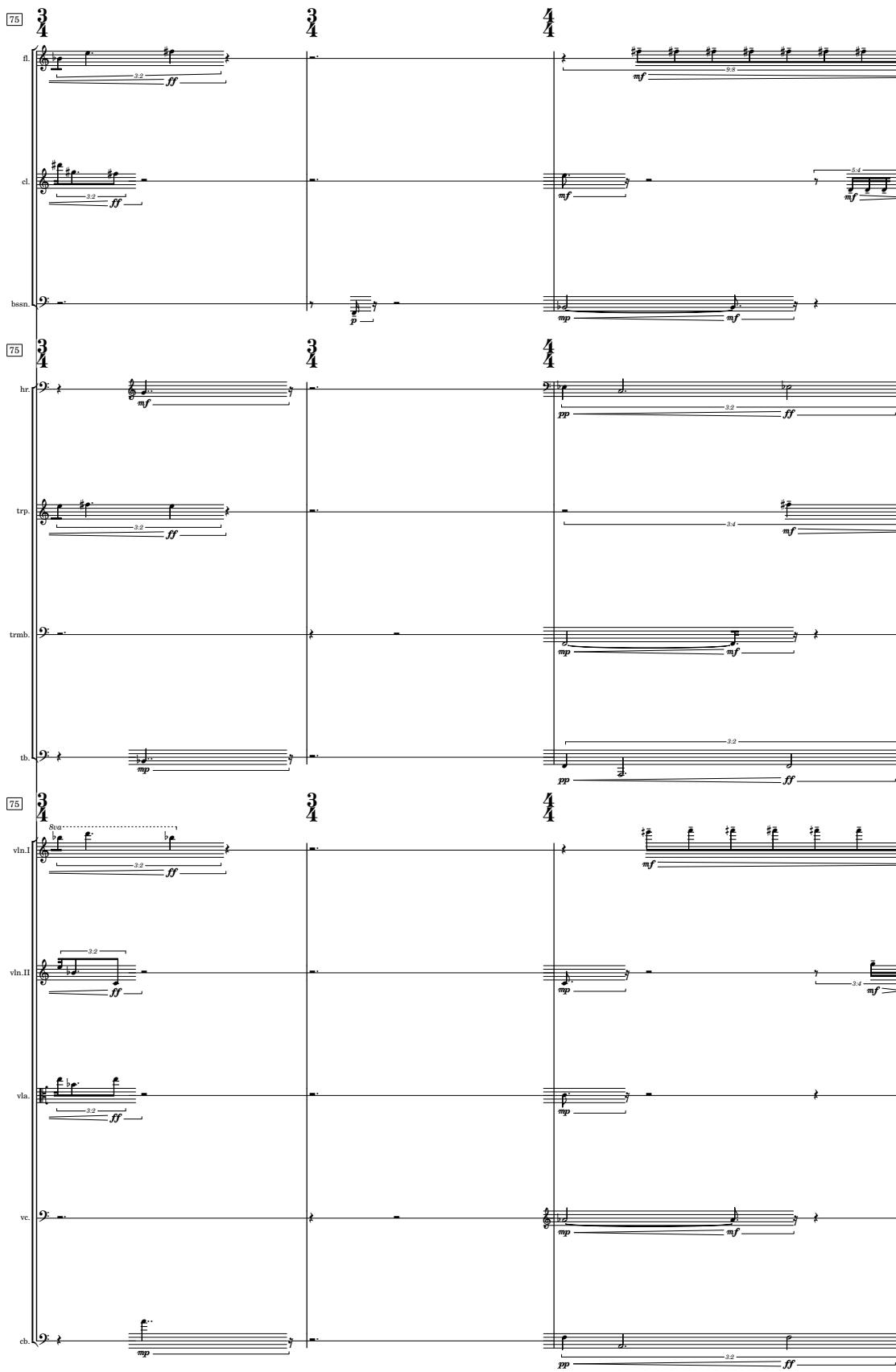
168 

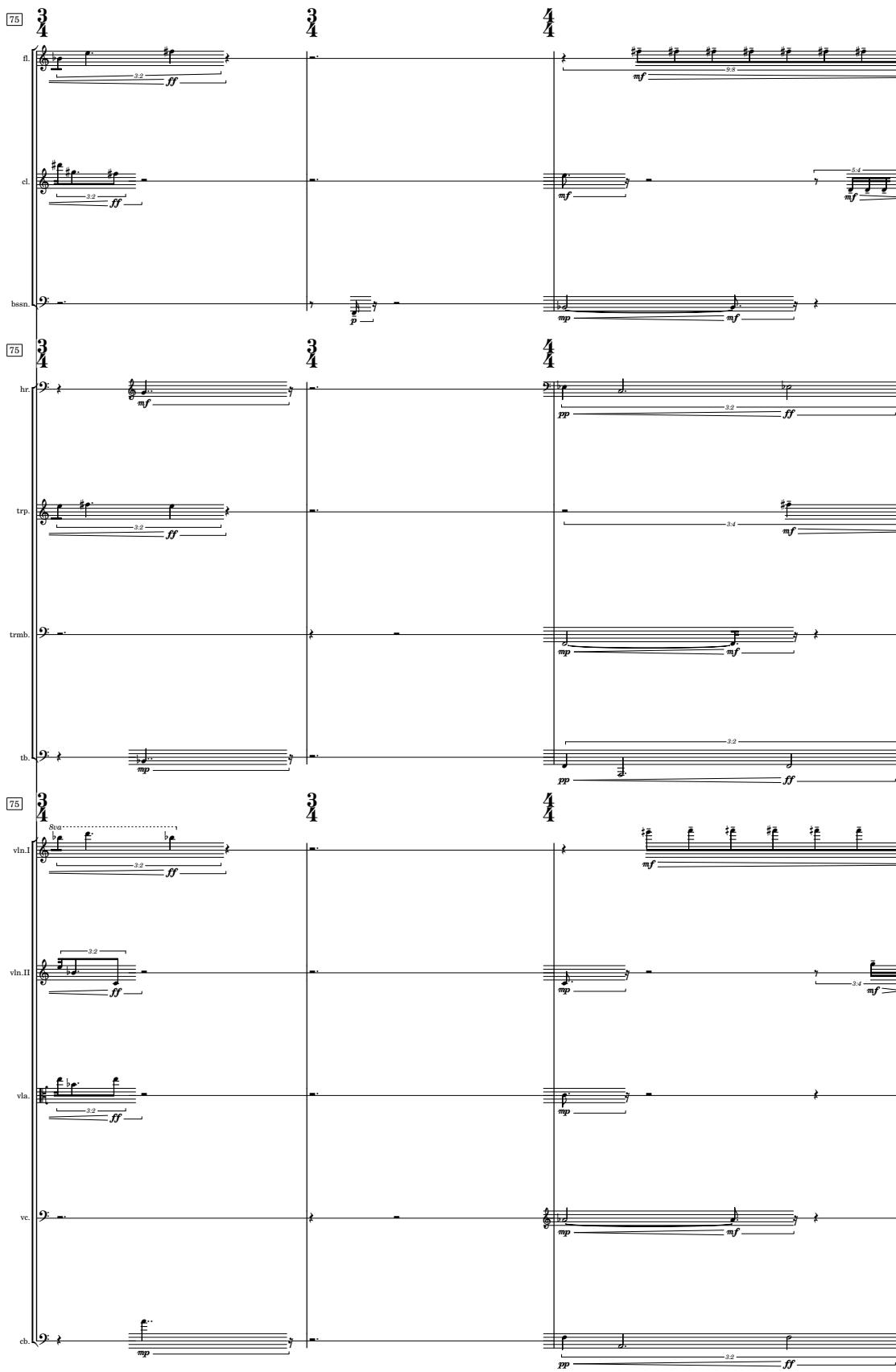
169 

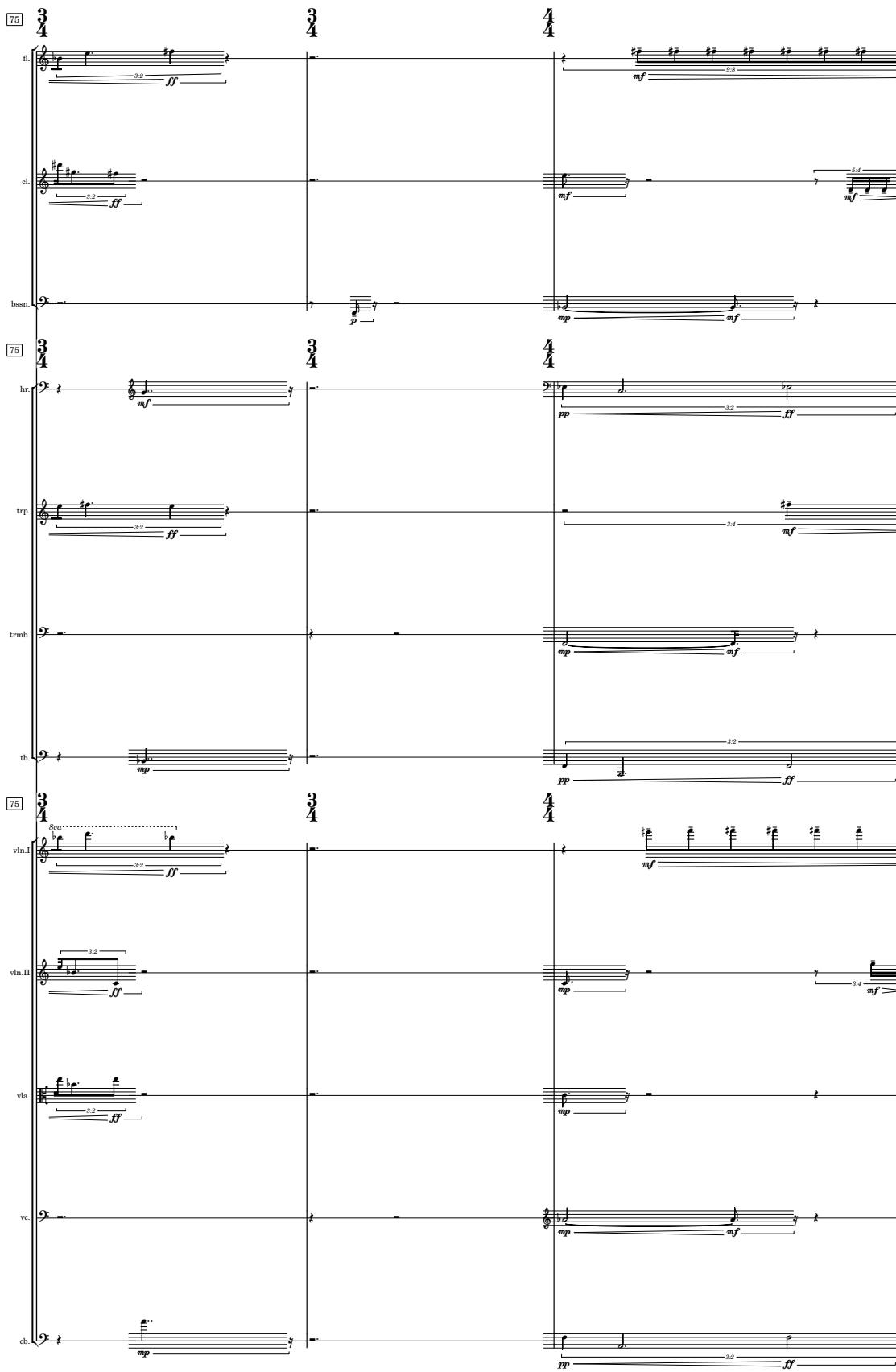
170 

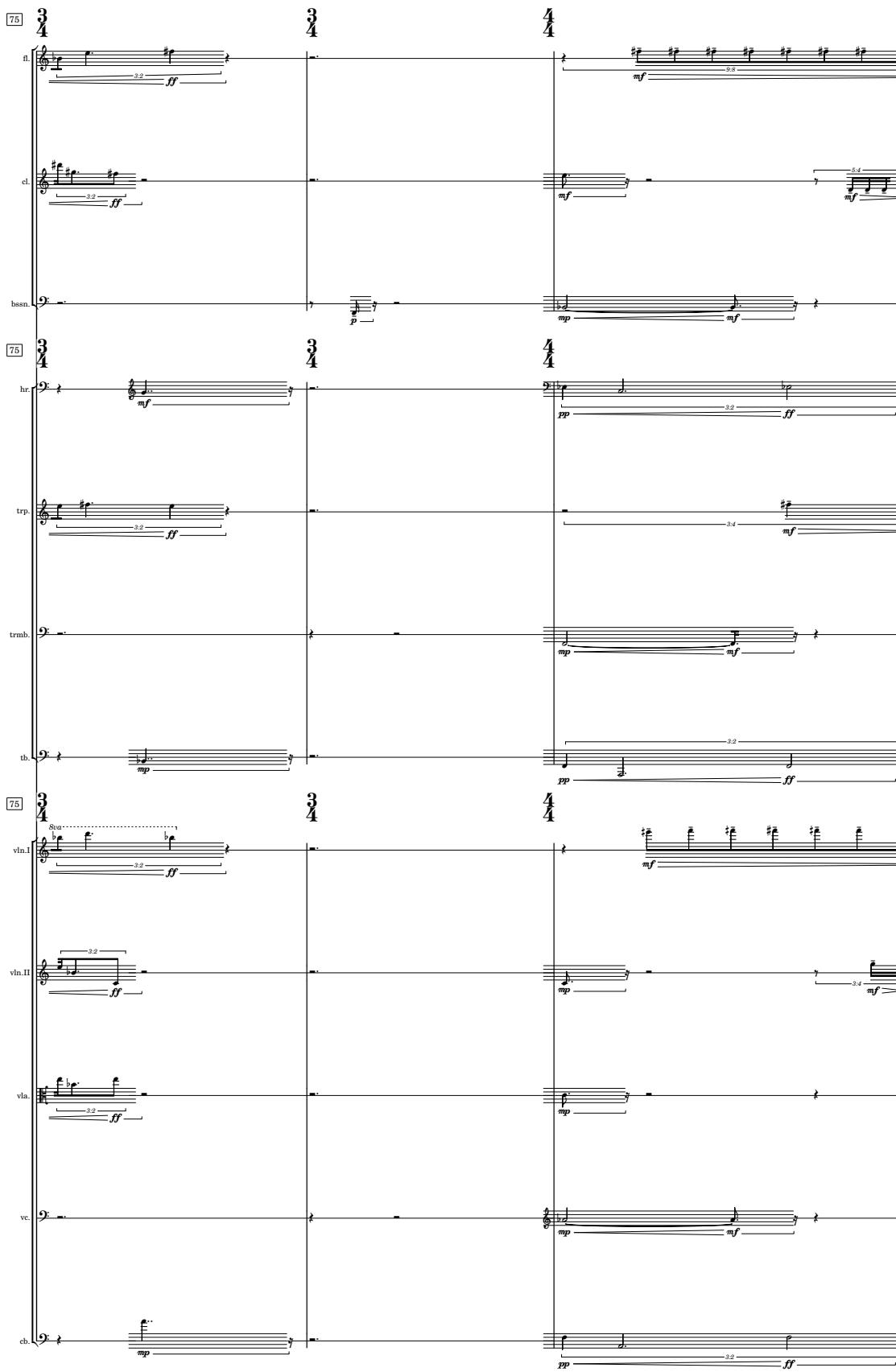
171 

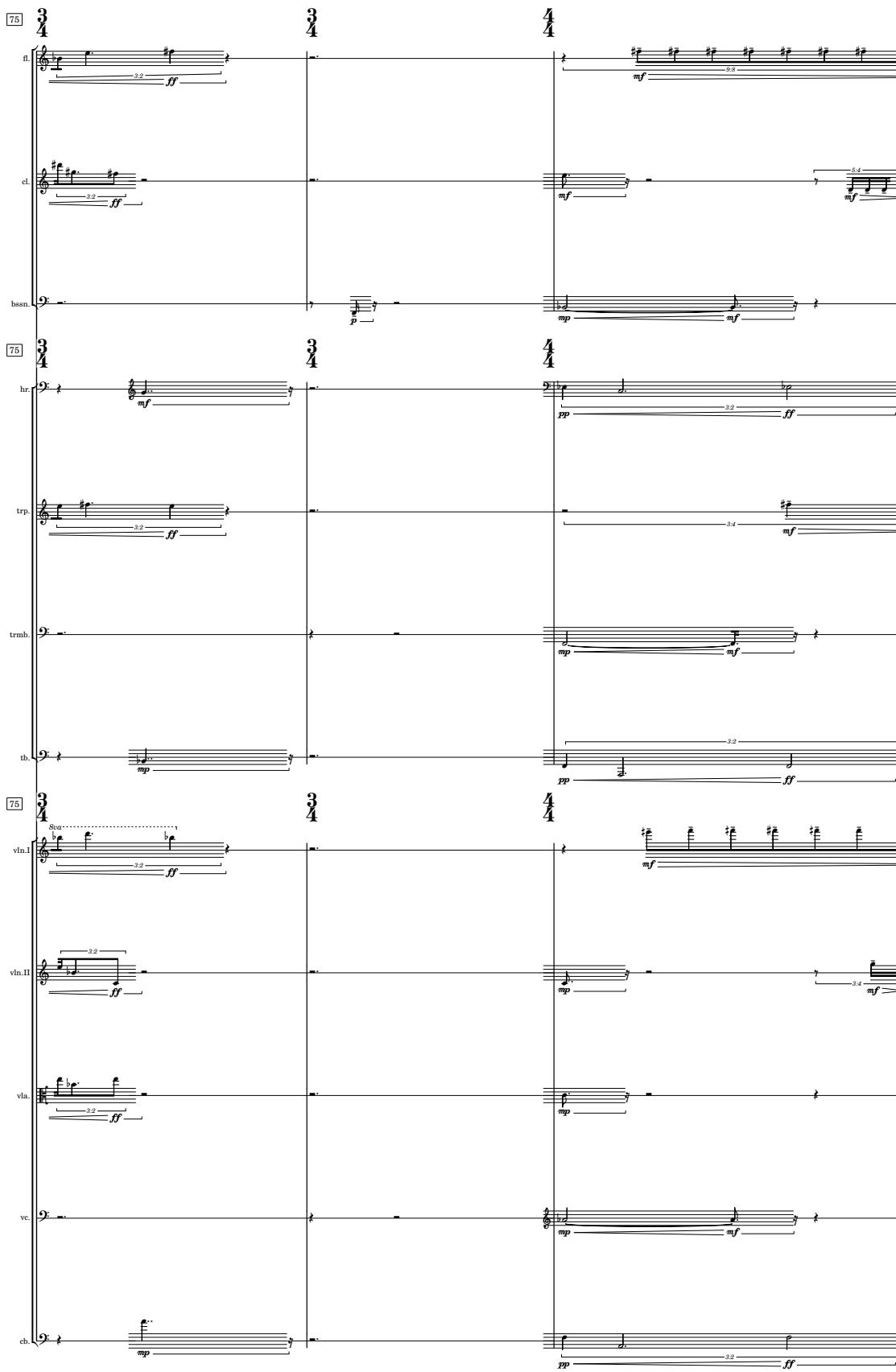
172 

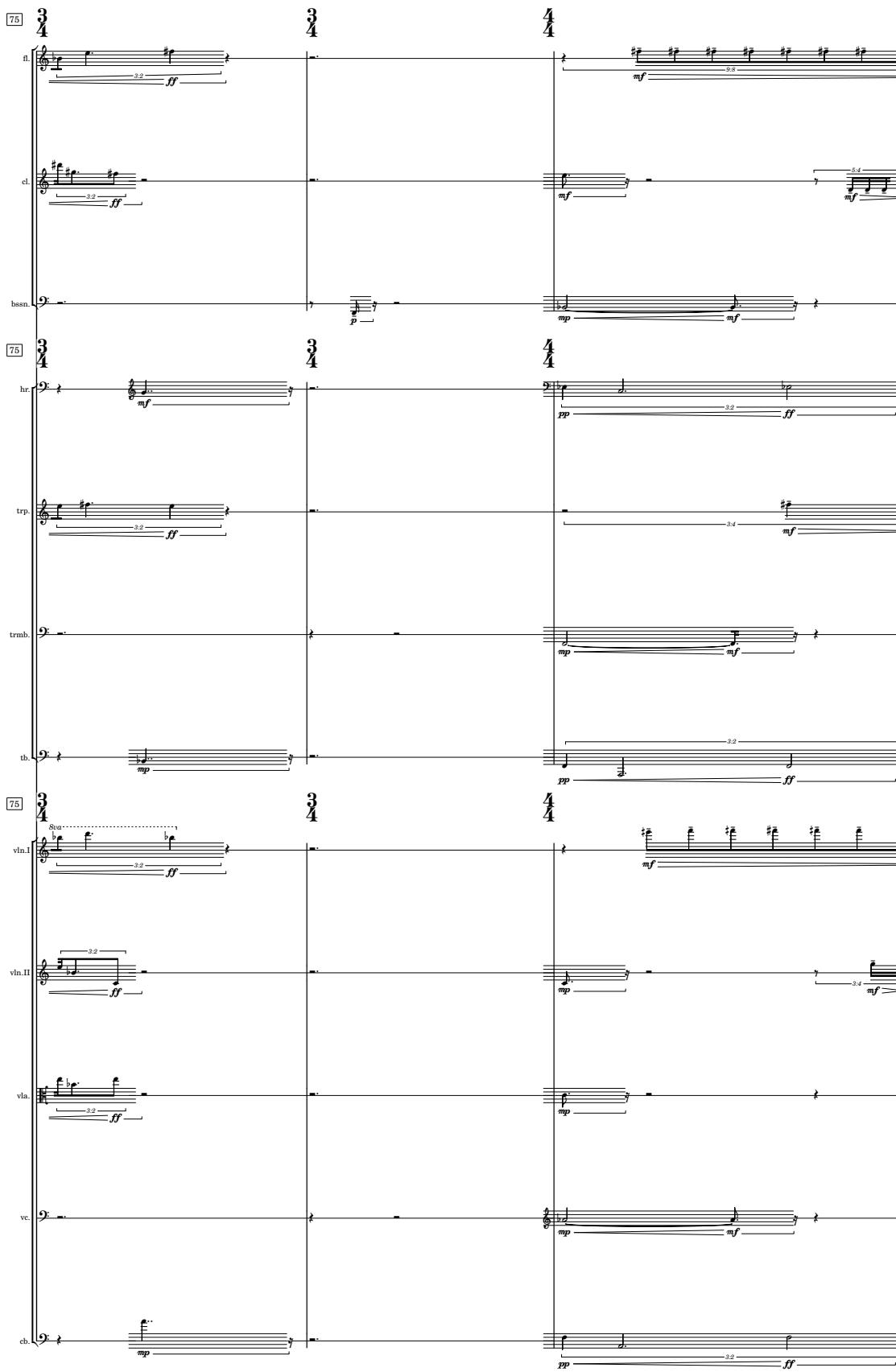
173 

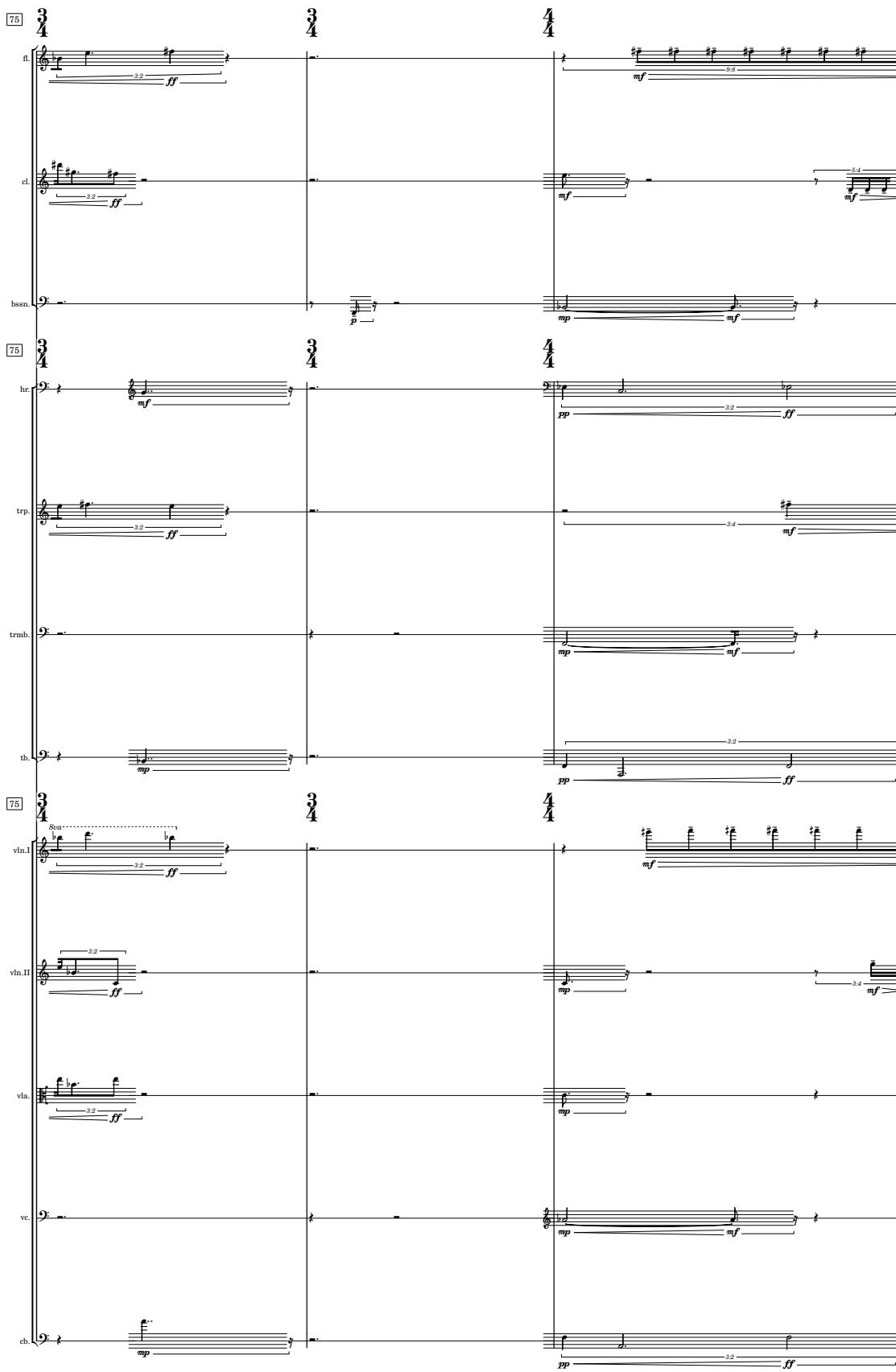
174 

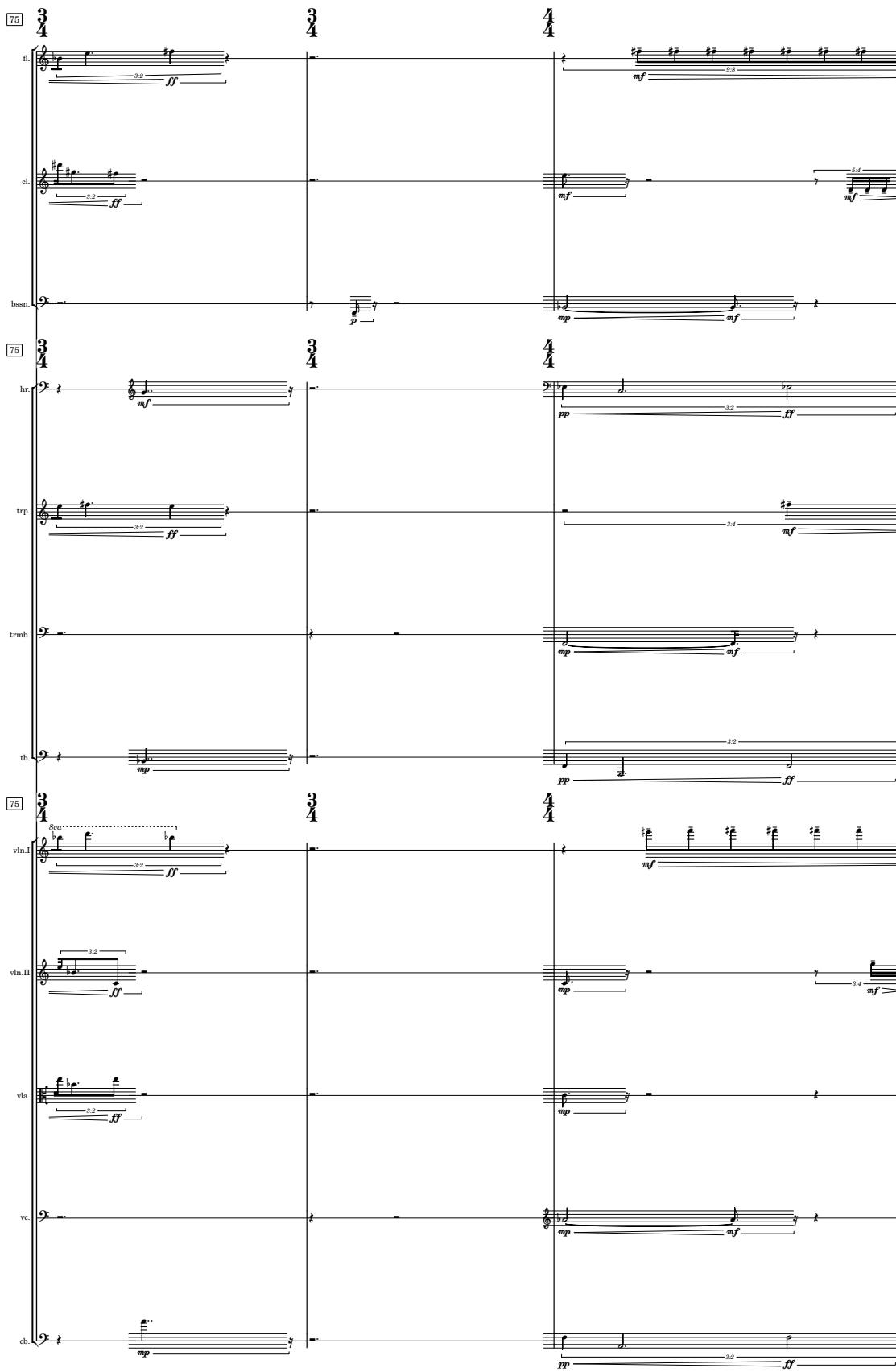
175 

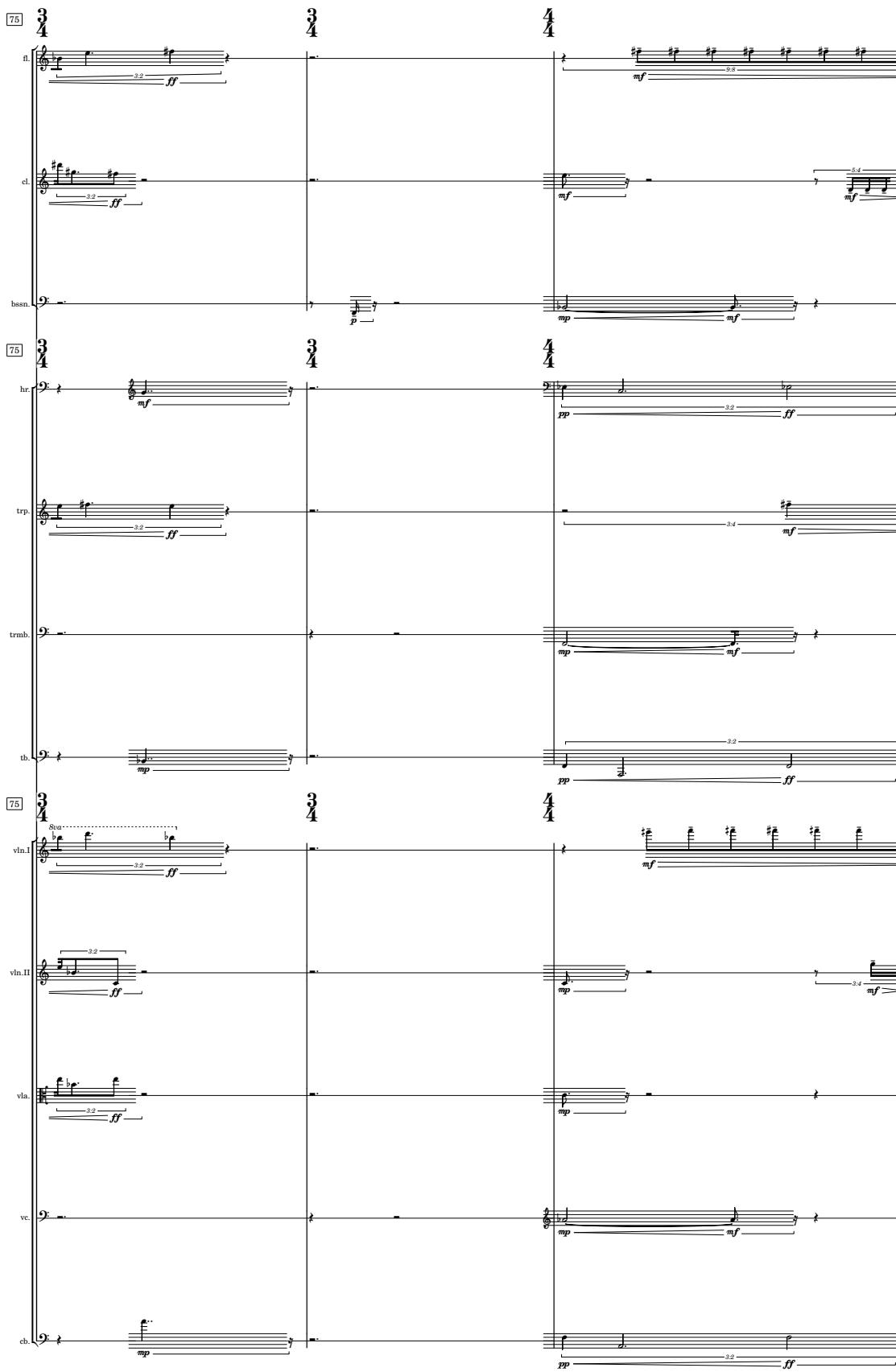
176 

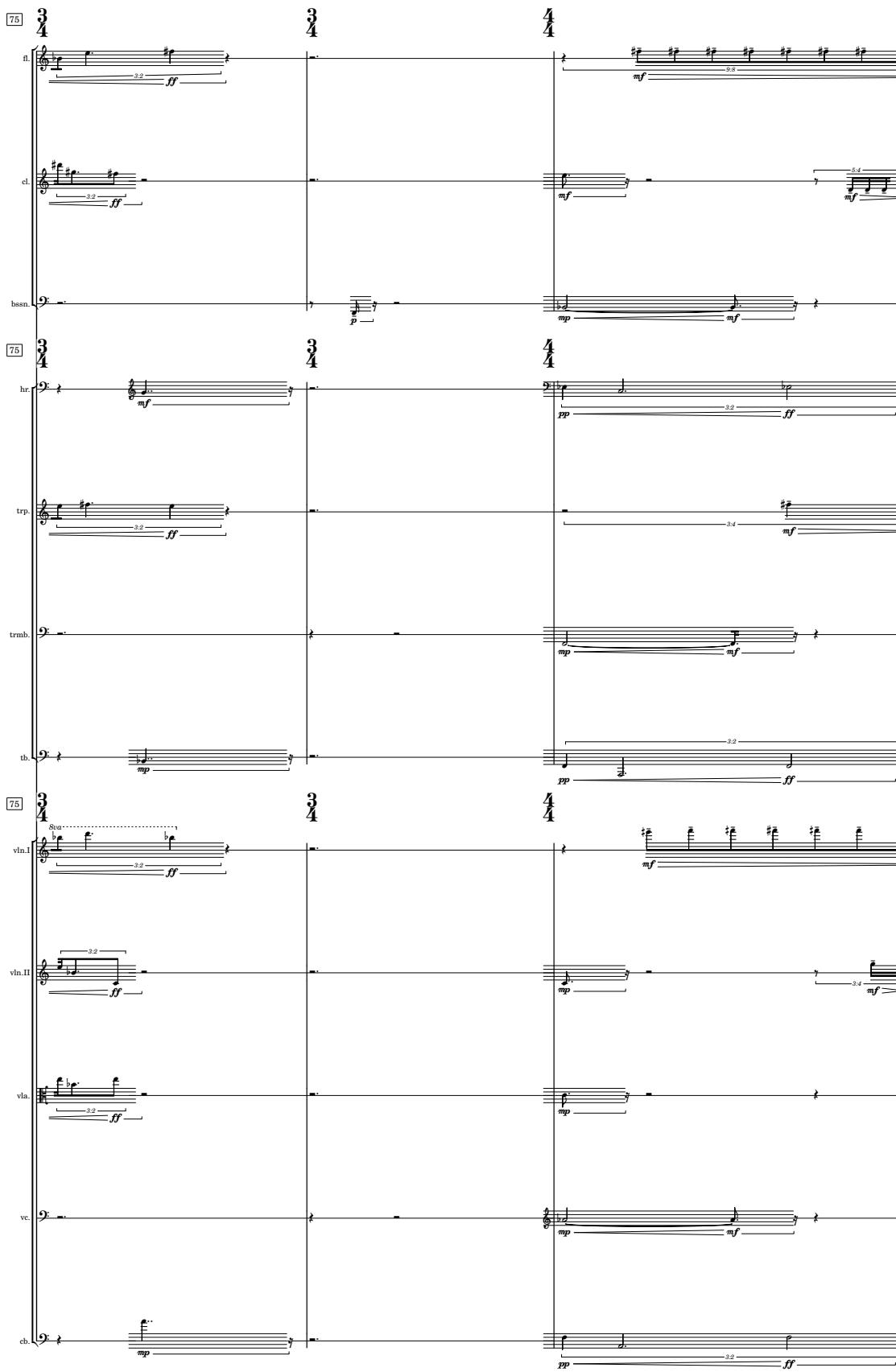
177 

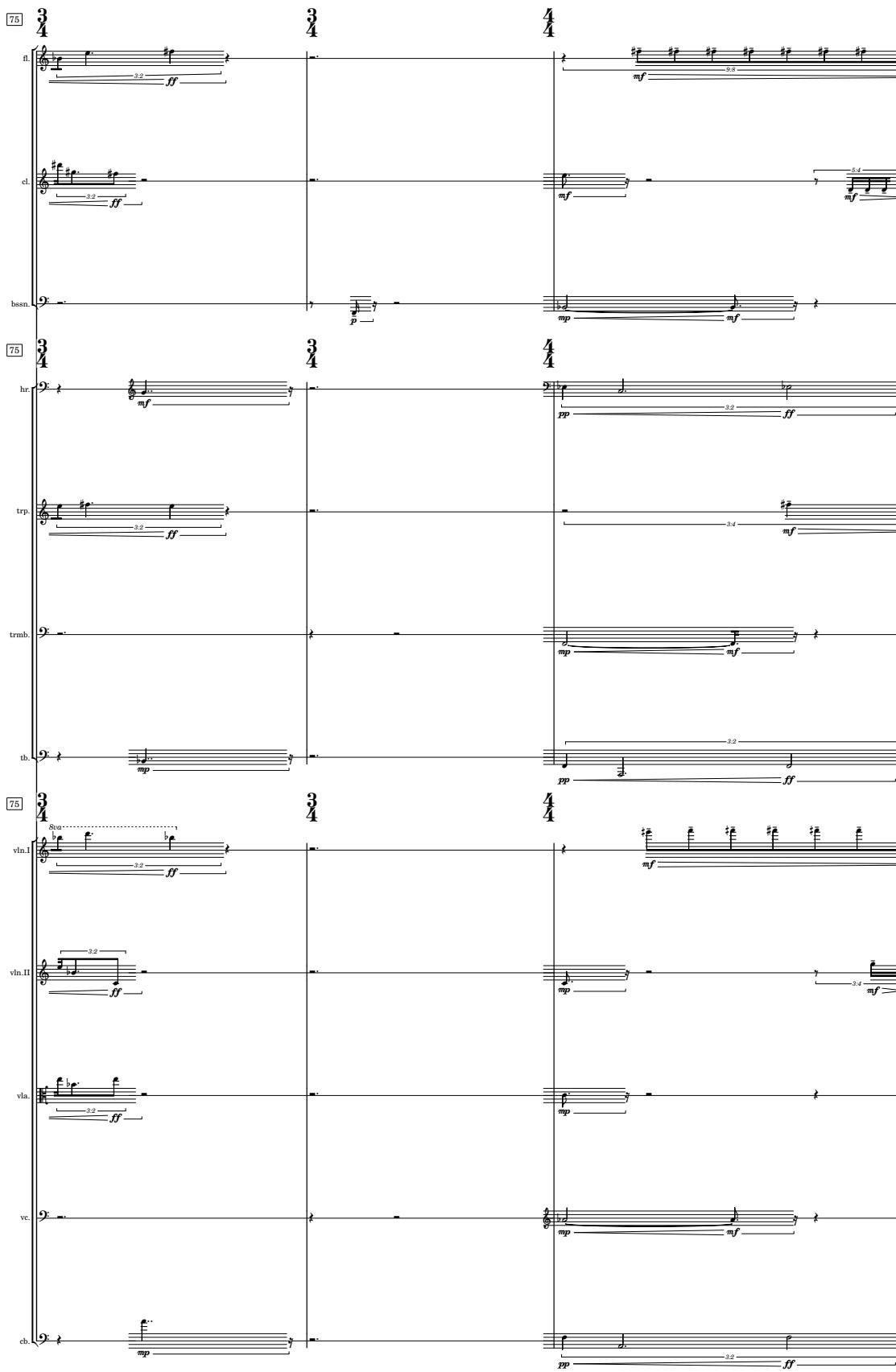
178 

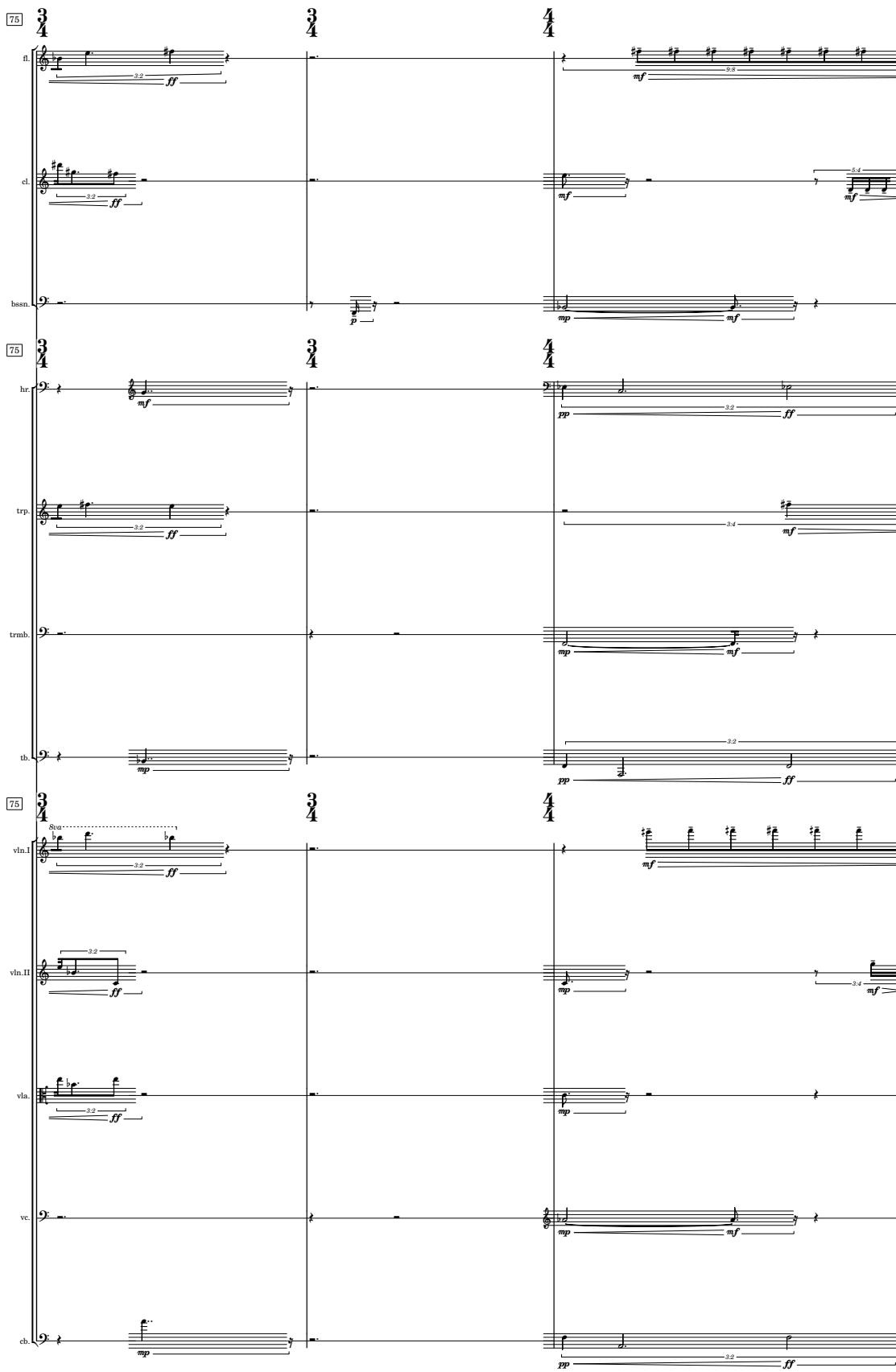
179 

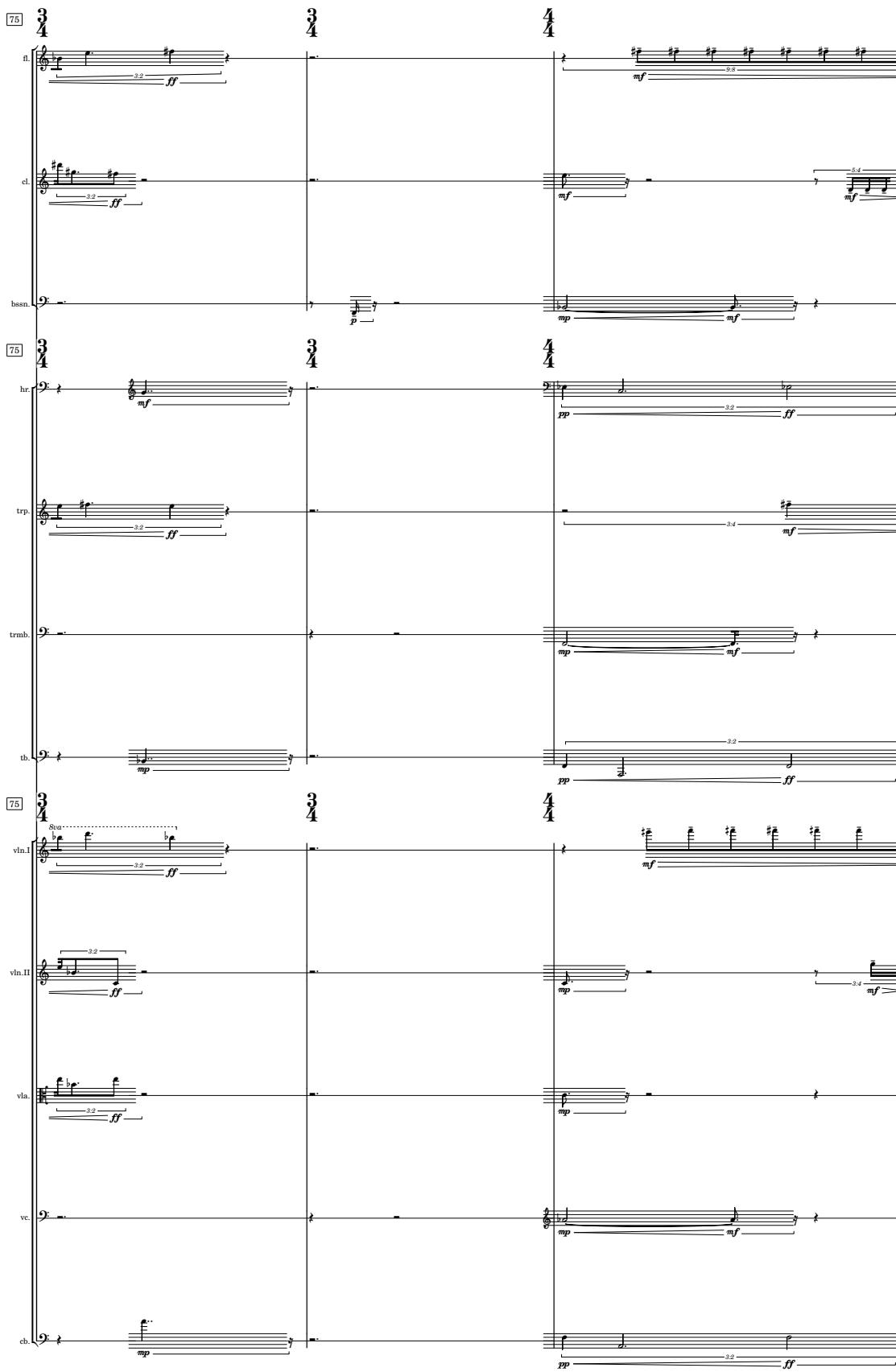
180 

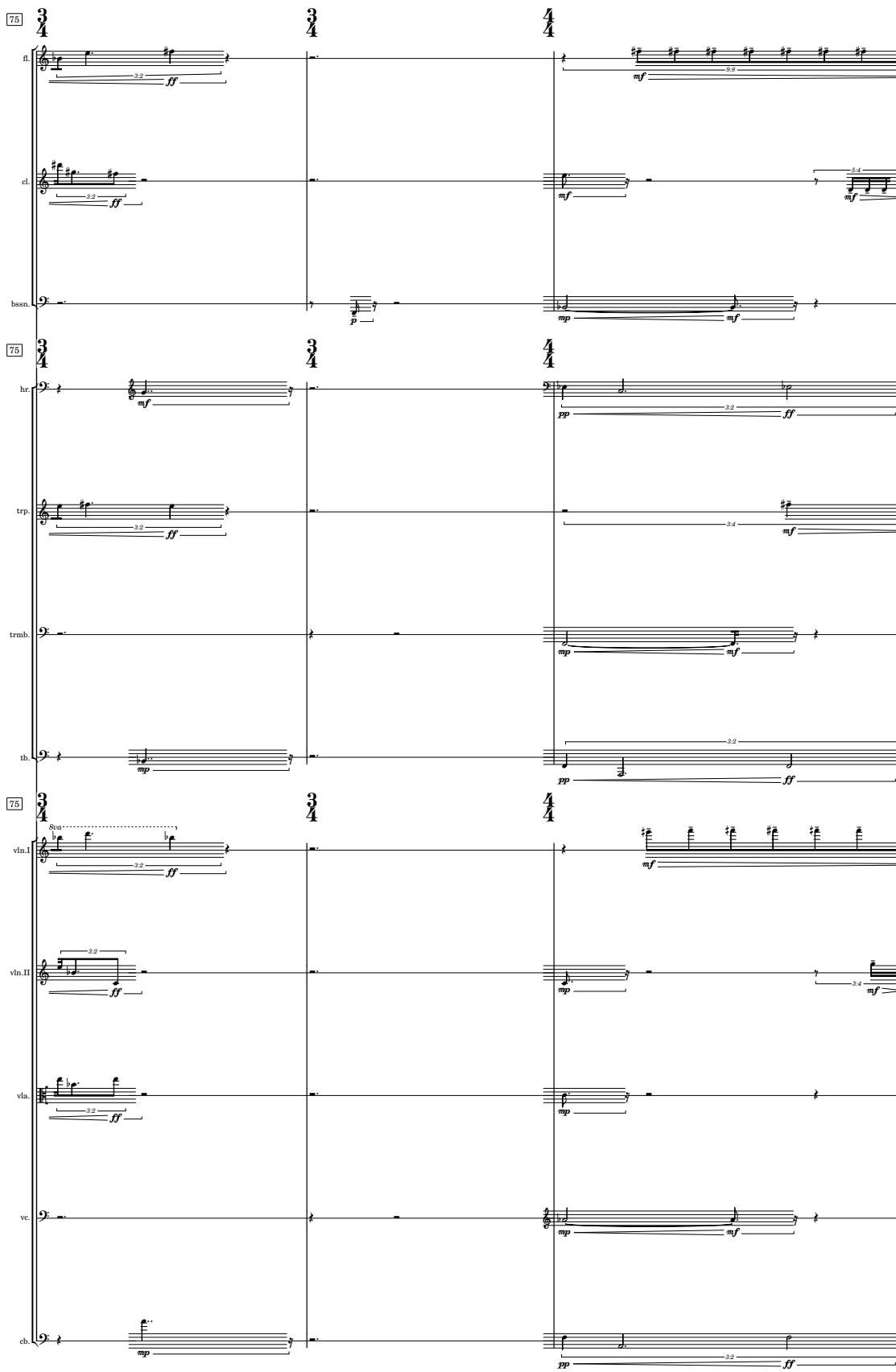
181 

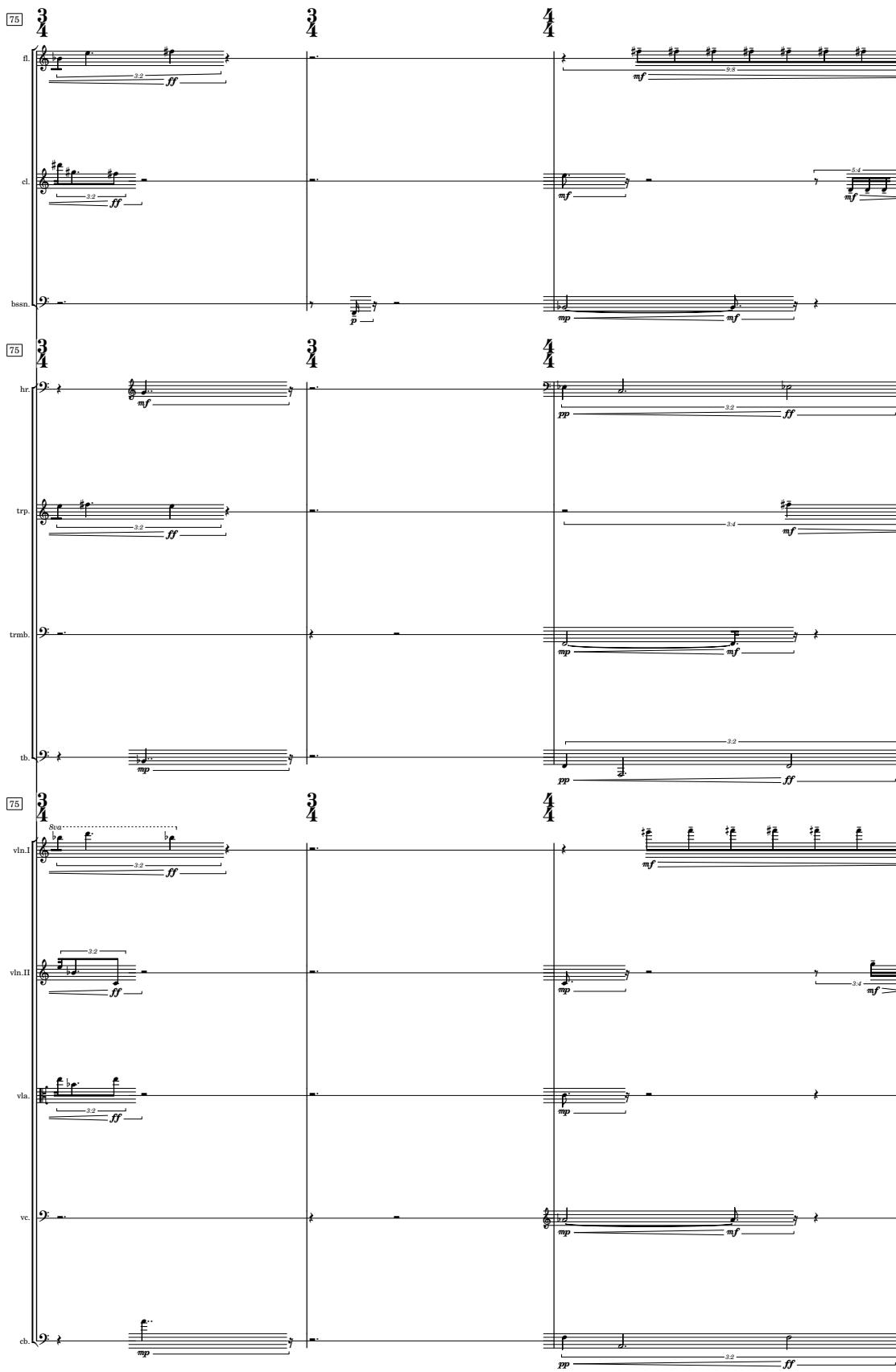
182 

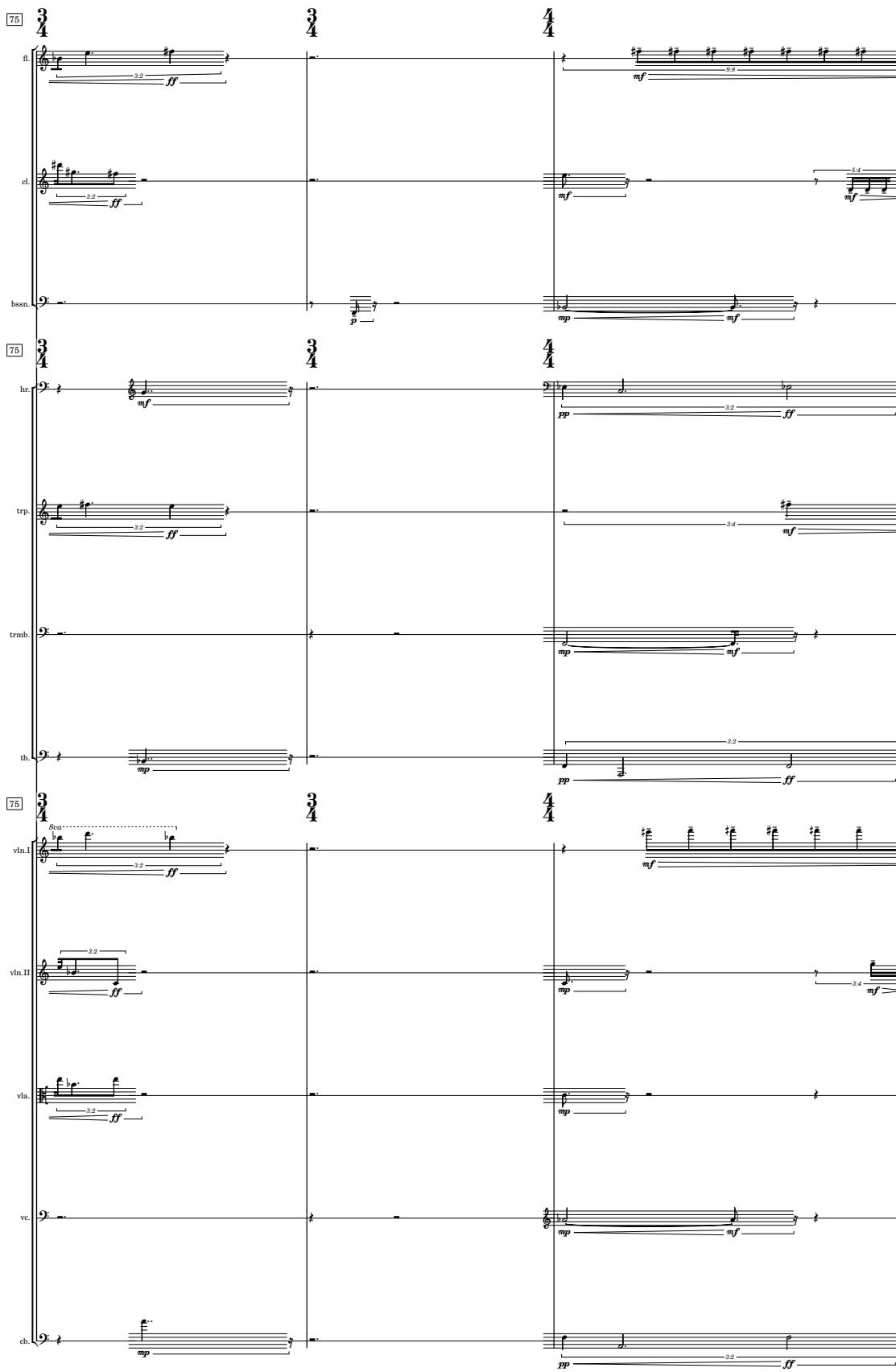
183 

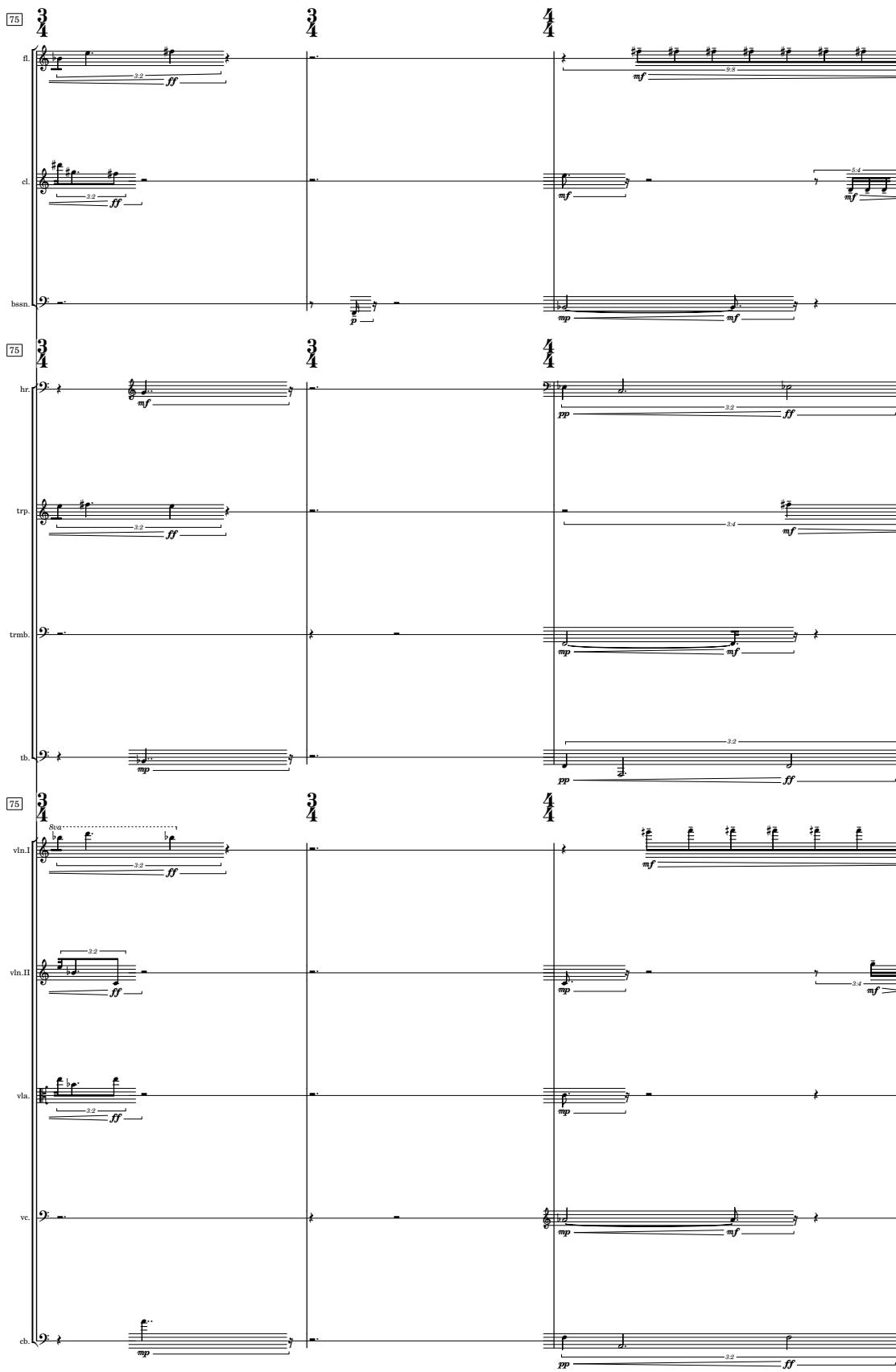
184 

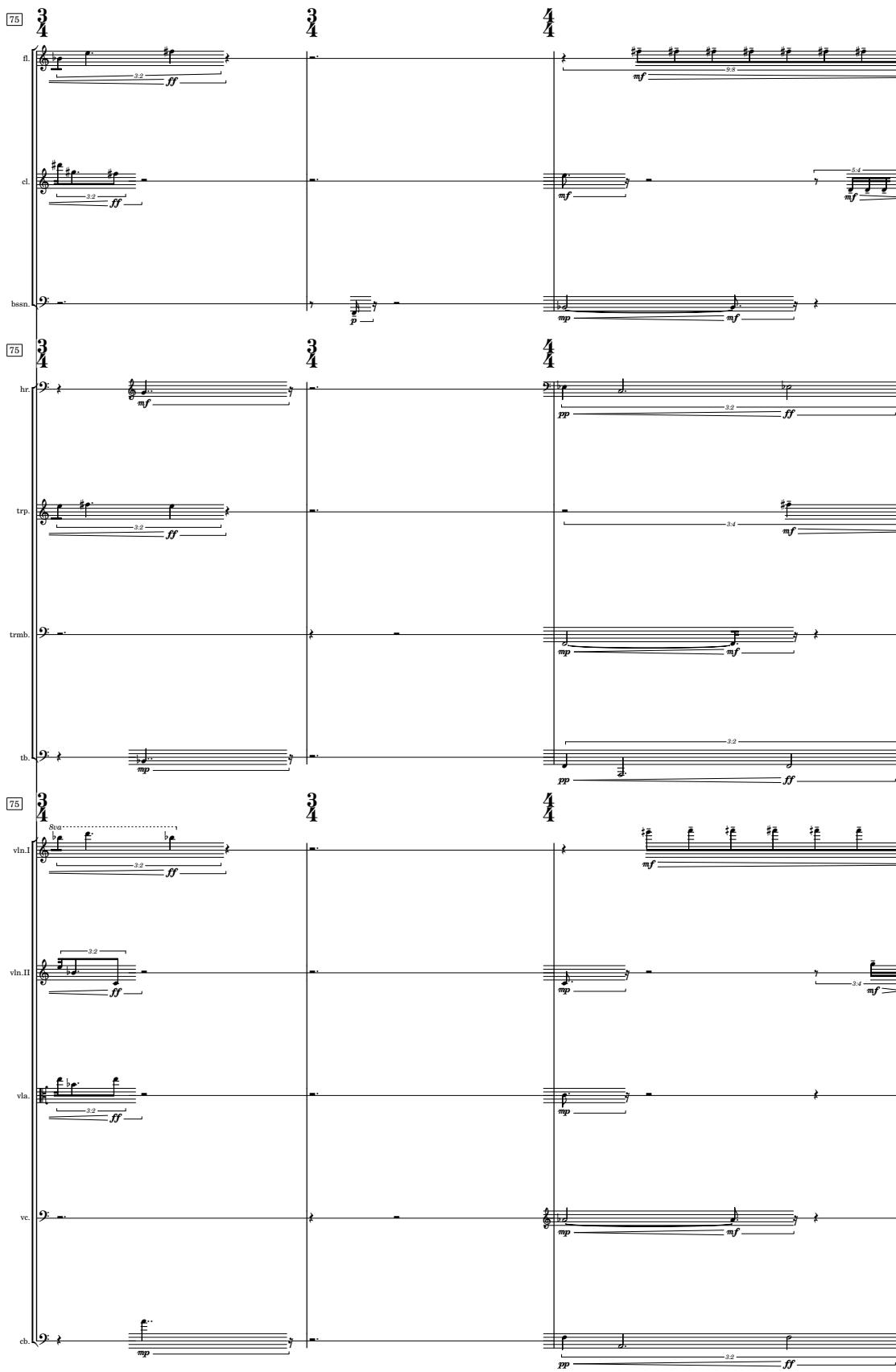
185 

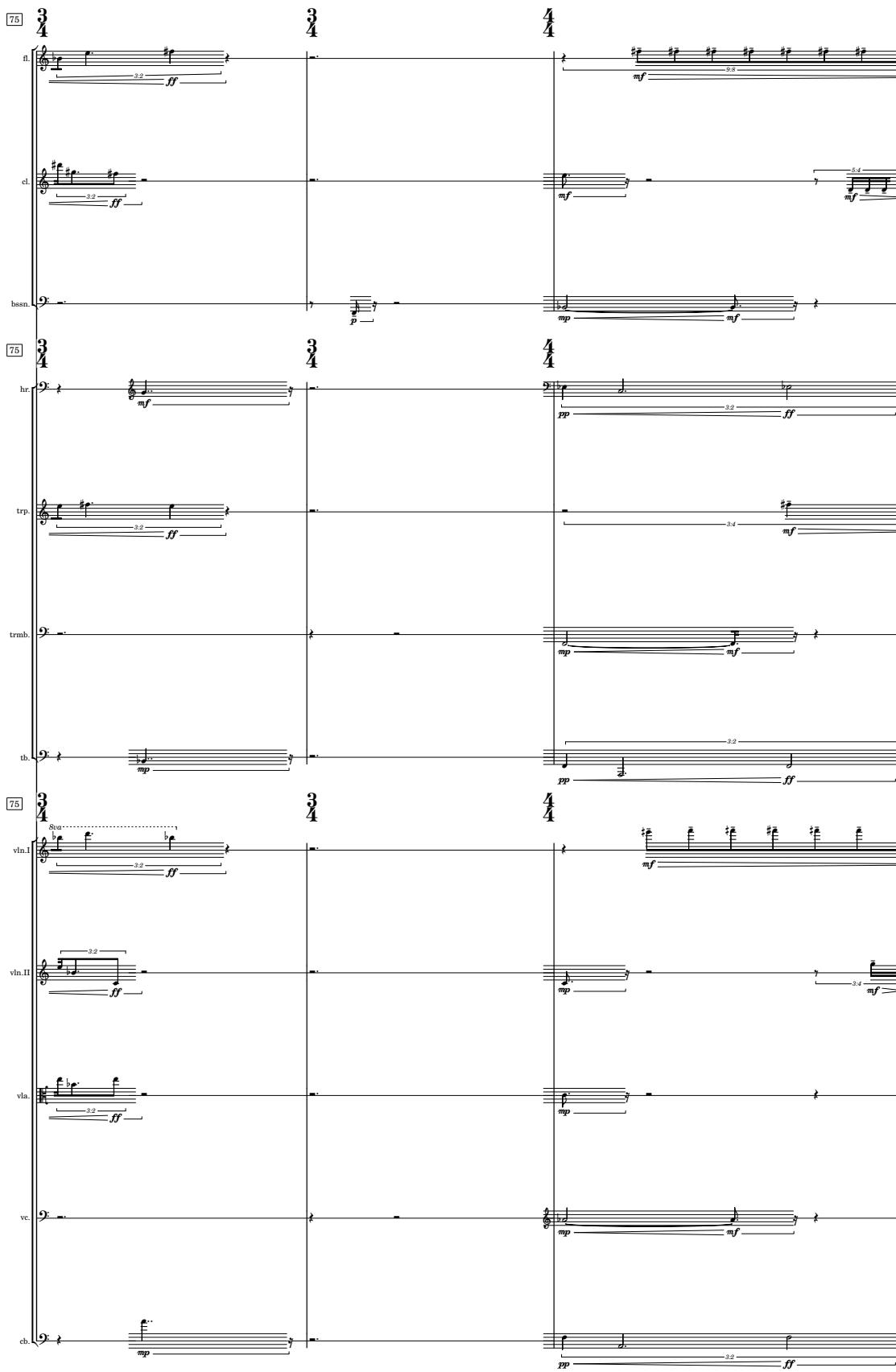
186 

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

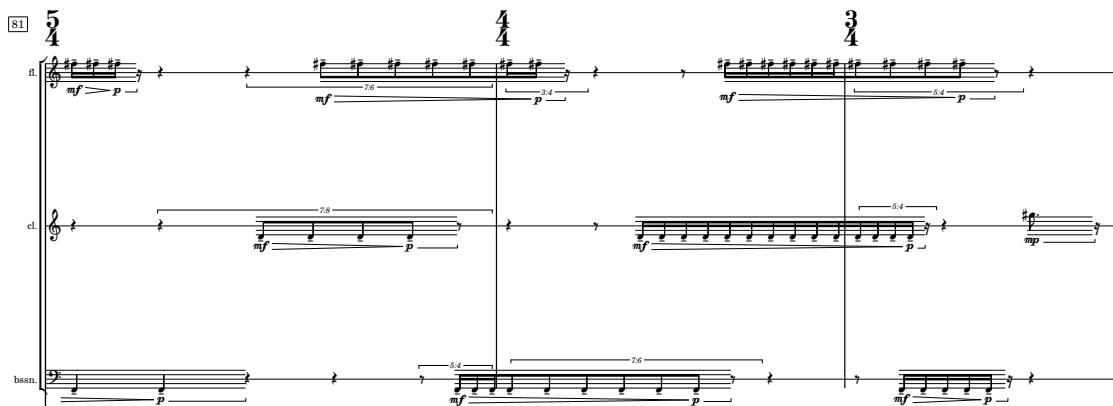
vln.I p 8.021 mf mf 12.12

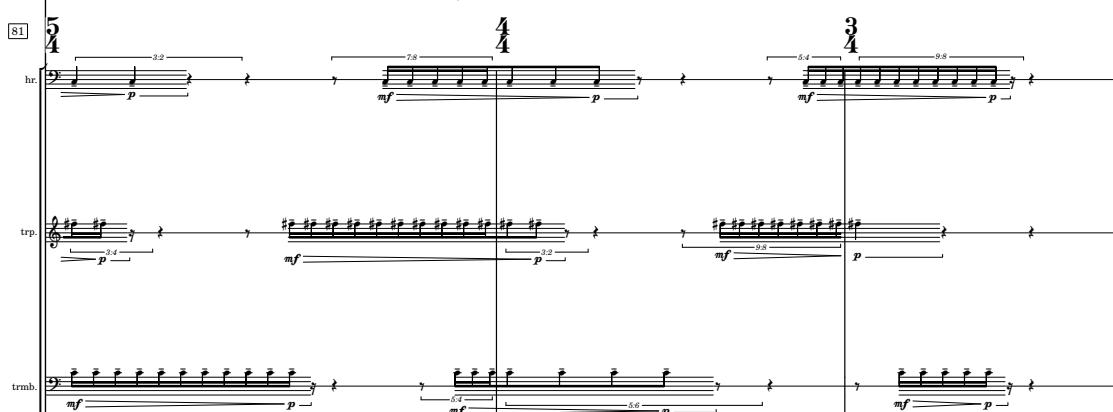
vln.II p mf p 9.8

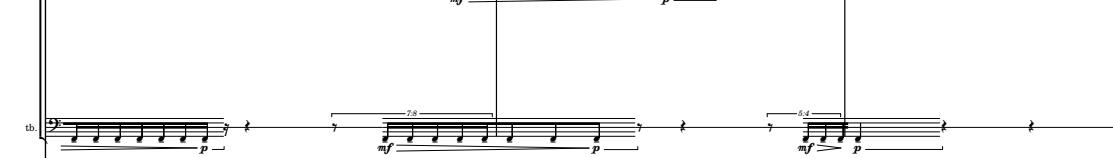
vla. mf p 12.16

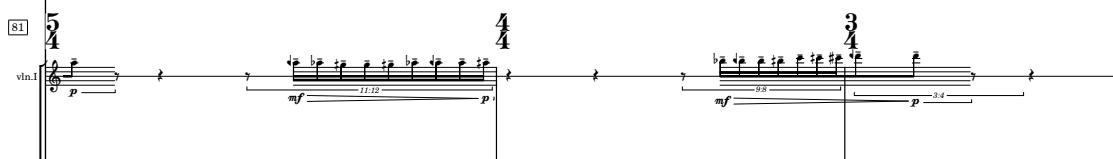
vc. pp ff mf p 9.8 3.2 mf

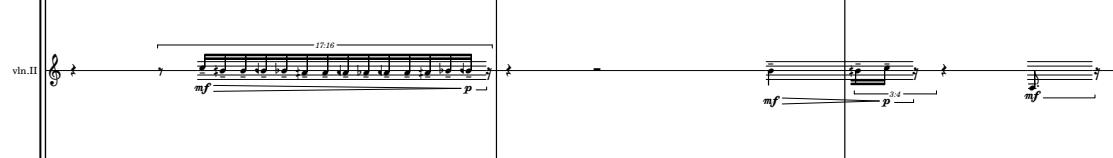
cb. 9 mf

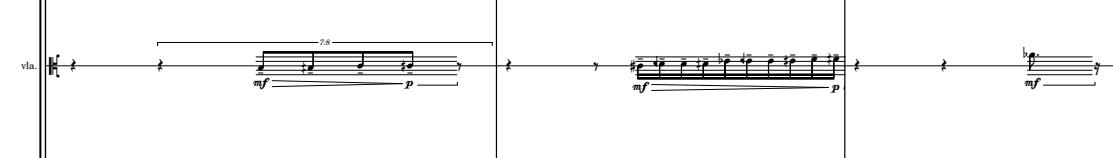
[81] 

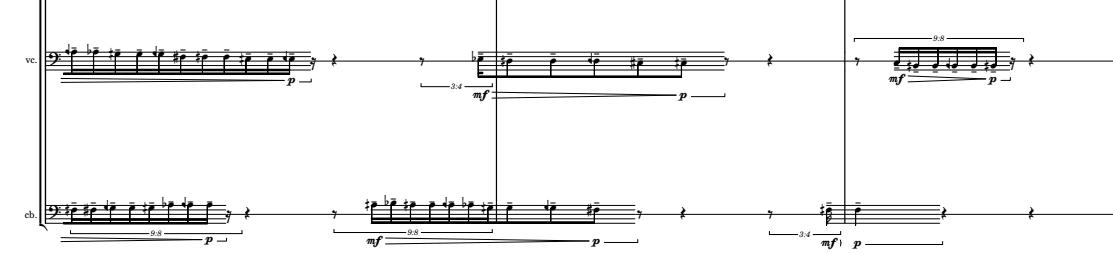
[81] 

[81] 

[81] 

[81] 

[81] 

[81] 

-30-

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.
trp.
trmb.
th.

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

88

f fl.

cl.

bassn.

88

hr.

trp.

trmb.

tb.

88

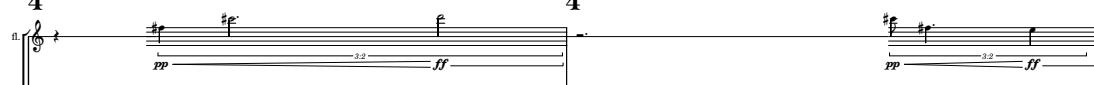
vln.I

vln.II

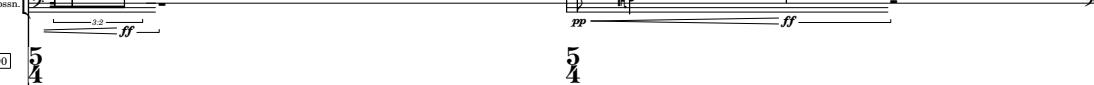
vla.

vc.

cb.

90 

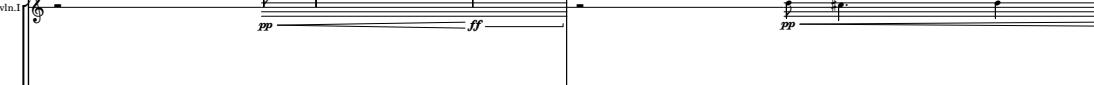


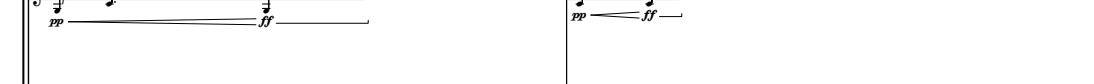
90 







90 







-34-

92

fl.

cl.

bassn.

hr.

trp.

trmb.

tb.

vln.I

vln.II

vla.

vc.

ch.

3

95

fl. cl. bsn.

hr. trp. trmb. th.

vln.I vln.II via. vc. cb.

98 **5**

f. *p* —————— 3.2 —————— *ff* ——————

5 *J = 60*

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

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

2

hr. *f* —————— 3.2 —————— *ff* ——————

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

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

th. —————— 3.2 —————— *ff* ——————

5

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

8va —————— *b2* —————— *f* ——————

5

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

2

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

vc. *f* —————— 3.2 —————— *ff* ——————

ch. *f* —————— 3.2 —————— *ff* ——————

101

fl.

cl.

bassn.

hr.

trp.

trmb.

th.

vln.I

vln.II

vla.

ve.

ch.

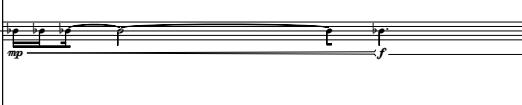
Measure 1 (4/4): fl. (f), cl. (mp), bassn. (mp), hr. (ff), trp. (f), trmb. (f), th. (mp), vln.I (ff), vln.II (mp), vla. (mp), ve. (ff), ch. (mf). Measures 2-3 (3/4): fl. (ff), cl. (mf), bassn. (mf), hr. (ff), trp. (ff), trmb. (ff), th. (ff), vln.I (ff), vln.II (mf), vla. (mf), ve. (ff), ch. (mf).

[104]

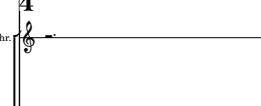
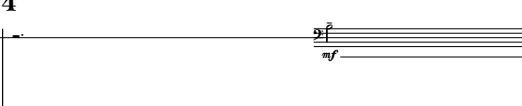
[M]

[104] <img alt

[107]  

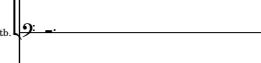
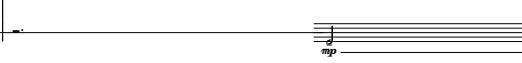
cl.  

bassn.  

[107]  

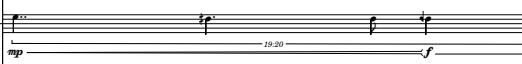
trp.  

trmb.  

th.  

[107]  

vln.II  

vla.  

vc.  

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

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

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

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

	2	3	4	3
fl.	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] $\frac{4}{4}$ ○ $\frac{3}{4}$ $\frac{5}{4}$

[130]
 fl. *f* | *mp* — 12.12 — *f* | *mp* — 7.5 —
 cl. *mp* — 15.16 — *f* | *mp* —
 bssn. *b* | *mp* — 11.12 — *mp* — 13.12 — *f* |
 hr. *b* | *mp* — 8.8 — *f* | *mp* — 7.8 — *f* | *mp* —
 trp. *f* | *mp* — 11.12 — *f* | *mp* — 9.8 —
 trmb. *b* | *mp* — 5.4 — *f* |
 th. *f* | *mp* — *f* | *mp* — 5.4 —
 vln.I *f* | *mp* — 11.12 — *f* | *mp* — 9.8 —
 vln.II *mp* — 17.16 — *f* | *mp* —
 vla. *mp* —
 vc. *f* | *mp* — 11.12 —
 cb. *f* | *mp* — 9.8 — *f* | *mp* —

[130]
 fl. *f* | *mp* — 12.12 — *f* | *mp* — 7.5 —
 cl. *mp* — 15.16 — *f* | *mp* —
 bssn. *b* | *mp* — 11.12 — *mp* — 13.12 — *f* |
 hr. *b* | *mp* — 8.8 — *f* | *mp* — 7.8 — *f* | *mp* —
 trp. *f* | *mp* — 11.12 — *f* | *mp* — 9.8 —
 trmb. *b* | *mp* — 5.4 — *f* |
 th. *f* | *mp* — *f* | *mp* — 5.4 —
 vln.I *f* | *mp* — 11.12 — *f* | *mp* — 9.8 —
 vln.II *mp* — 17.16 — *f* | *mp* —
 vla. *mp* —
 vc. *f* | *mp* — 11.12 —
 cb. *f* | *mp* — 9.8 — *f* | *mp* —

[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] **5**

f.
mf

cl.
mf

bsn.

[134] **5**

hr
mp

trp.
mf

trmb.

th
mf

[134] **5**

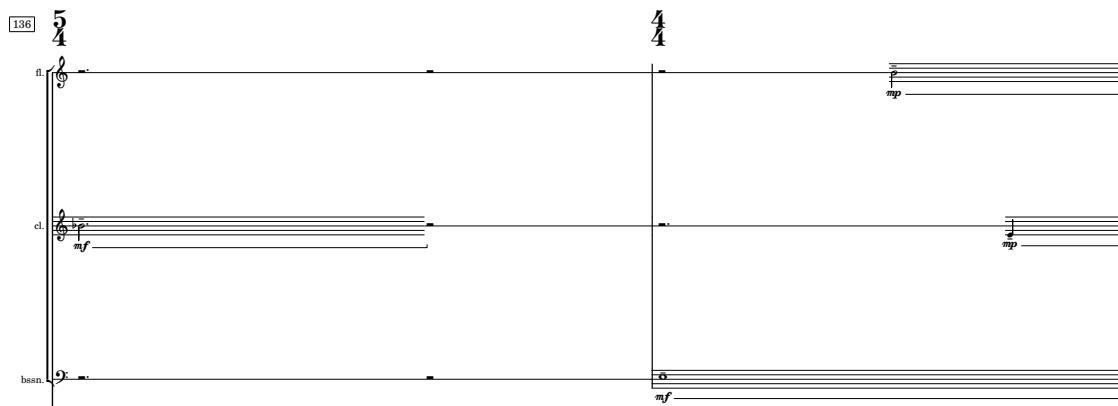
vln.I
mf

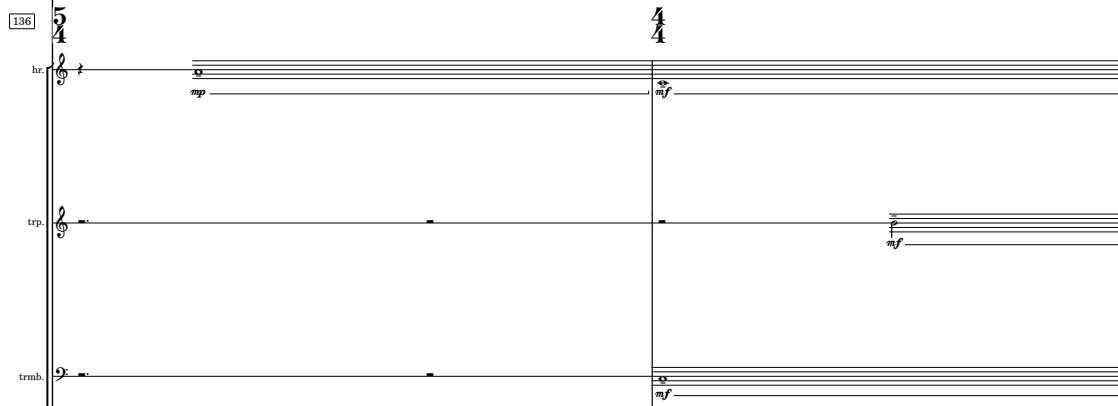
vln.II
mf

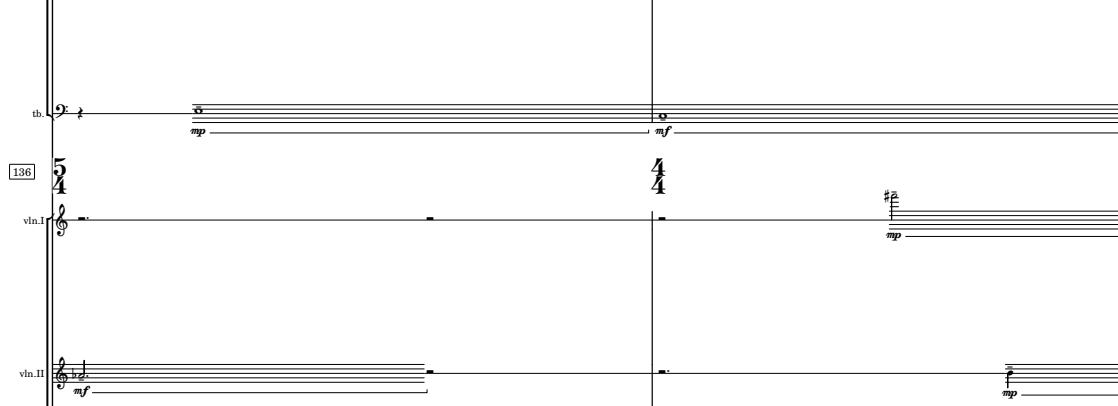
vla.
mf

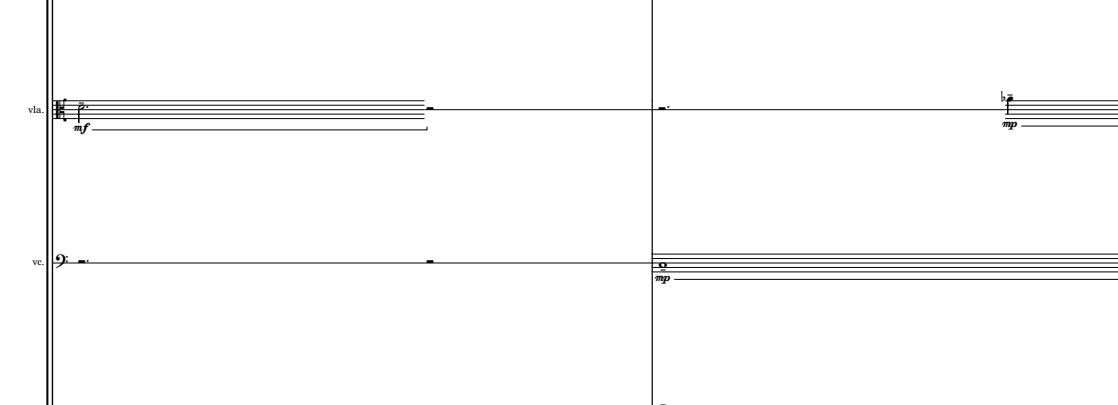
vc
mf

ch
mf

[136] 

[136] 

[136] 



[138] $\frac{4}{4}$ (R) $\frac{5}{4}$

fl.

cl.

bassn.

[138] $\frac{4}{4}$ (R) $\frac{5}{4}$

hr.

trp.

trmb.

th.

[138] $\frac{4}{4}$ (R) $\frac{5}{4}$

vln.I

vln.II

vla.

vc.

cb.

[140]
fl. -
cl. -
bsn. -

[140]
hr. -
trp. -

trmb. -

th. -

[140]
vln.I -
vln.II -

vla. -

vc. -

ch. -

[142]

142

fl. *mf* *f* *f* *f* *f*

cl. *f* *f* *f* *f* *f*

bassn. *f* *f* *f* *f* *f*

142

hr. *f* *f* *f* *f* *f*

trp. *f* *f* *f* *f* *f*

trmb. *f* *f* *f* *f* *f*

tb. *f* *f* *f* *f* *f*

142

vln.I *f* *f* *f* *f* *f*

vln.II *f* *f* *f* *f* *f*

vla. *f* *f* *f* *f* *f*

vc. *f* *f* *f* *f* *f*

cb. *f* *f* *f* *f* *f*

[145] $\frac{4}{4}$ $\frac{3}{4}$ $\frac{5}{4}$

f. $\frac{4}{4}$ $\frac{3}{4}$ $\frac{5}{4}$

cl. $\frac{4}{4}$ $\frac{3}{4}$ $\frac{5}{4}$

bsns. $\frac{4}{4}$ $\frac{3}{4}$ $\frac{5}{4}$

[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] $\frac{5}{4}$ $\downarrow = 120$

f.
cl.
bssn.

[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.
ch.

151

Flute: dynamic ff, measure 9.8

Clarinet: dynamic ff, measure 9.8; dynamic mf, measure 11.12

Bassoon: dynamic ff, measure 11.12

Horn: dynamic p, measure 7.8

Trumpet: dynamic ff, measure 15.16

Trombone: dynamic ff, measure 7.8

Tuba: dynamic ff, measure 7.8

Violin I: dynamic ff, measure 9.8

Violin II: dynamic ff, measure 9.8

Viola: dynamic ff, measure 7.8

Cello: dynamic ff, measure 7.8

Double Bass: dynamic ff, measure 7.8

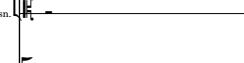
[154]

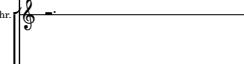
[156]  

fl.

[156]  

cl.

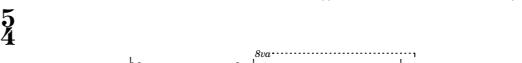
bassn.  

[156]  

hr.

trp.  

trmb.  

th.  

[156]  

vln.I

vin.II  

vla.  

vc.  

cb.

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 

193 

194 

195 

196 

197 

198 

199 

200 

201 

202 

203 

204 

205 

206 

207 

208 

209 

210 

211 

212 

213 

214 

215 

216 

217 

218 

219 

220 

221 

222 

223 

224 

225 

226 

227 

228 

229 

230 

231 

232 

233 

234 

235 

236 

237 

238 

239 

240 

241 

242 

243 

244 

245 

246 

247 <img alt="Measure 247: Flute, Clarinet, Bassoon, Trombone, Trom

163

fl. $\frac{5}{4}$ $\frac{3}{4}$ (1) $\frac{3}{4}$ $\frac{3}{4}$

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]

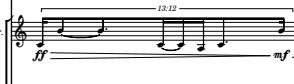
170   

fl.  

cl.  

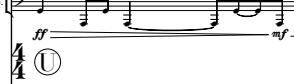
bassn.  

170   

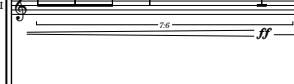
hr.  

trp.  

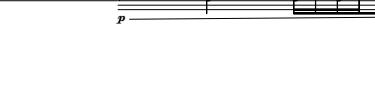
trmb.  

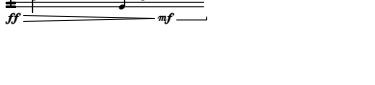
tb.  

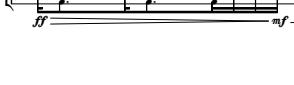
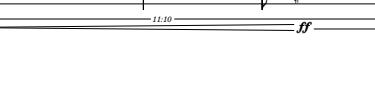
170   

vln.I  

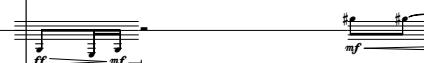
vln.II  

vla.  

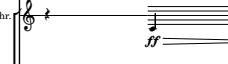
vc.  

ch.  

173  

cl.  

bassn.  

173  

trp.  

trmb.  

th.  

173  

vin.II  

vla.  

vc.  

ch.  

[176] $\frac{3}{4}$ fl. ff — ff = mf —

cl. ff — 5.4 ff —

bassn. —

[176] $\frac{3}{4}$ (V) hr. —

trp. ff — ff = mf — ff > mf —

trmb. — p pp mf ff — mf —

th. — ff = mf ff — mf —

[176] $\frac{3}{4}$ vln.I Sva... ff = mf — ff > mf —

vin.II ff —

vla. ff — 5.4 —

vc. — p pp mf ff — mf —

ch. — ff = mf — mf = ff —

179

fl. ff
cl. mf ff
bassoon ff mf 3.2 ff 2.6
hr. f ff mf ff 5.4 ff
trp. ff mf 7.6 ff
trmb. ff ff mf 3.2 ff 2.6 ff
th. ff ff mf ff 5.4 ff 3.2 ff
vln.I ff ff mf ff 7.6 ff
vln.II ff ff mf ff 7.6 ff
vla. ff ff mf ff 5.6 ff
vc. ff ff mf ff 7.6 ff
cb. ff ff mf ff 5.4 ff 3.2 ff

181 $\frac{3}{4}$

fl. ff — 5.4 —

cl. ff — 5.4 — ff — mf —

bassn. mf — ff —

181 $\frac{3}{4}$ $\frac{5}{4}$ (W)

hr. ff —

trp. ff —

trmb. mf — ff —

th. ff —

181 $\frac{3}{4}$ $\frac{5}{4}$ (W)

vln.I ff — 3.4 —

vln.II ff — 5.4 — ff —

vla. ff — 5.4 — ff —

vc. mf — ff —

cb. ff —

8va.....

[183]

[183]

185 



186 



187 



188 

[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 

189 



189 







191

191

fl.

cl.

bassn.

hr.

trp.

trmb.

th.

vln.I

vln.II

vla.

vc.

cb.

9.8

p pp

7.8

p pp

7.8

p pp

15.16

p pp

5.4

11.12

p pp

9.8

p pp

8.0

p pp

3.4

pp

17.16

p

5.4

11.12

p pp

9.8

p pp

9.8

p pp

7.8

p pp

-73-

194 4 3 98

f fl. # p 5.8 pp

cl. p 5.4 pp

bassn. 13.12 p 15.16 pp

194 4 3 98

hr. pp 5.4 pp

trp. 9.8 p pp

trmb. 13.12 p pp

tb. pp 7.8 pp

194 4 3 98

vln.I Sforzando 16.16 pp

vln.II pp 7.8 pp

vla. pp 17.16 pp

vc. 13.12 p 15.16 pp

cb. pp 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 : Obengaualten (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 , mf , p . Measures 11-12: $5:4$, $13:12$.

Saxophone: Measures 9-10: p . Measure 11: mf , p .

Violoncello: Measures 9-10: p . Measure 11: p .

(3)

fl. (Measure 5): mf , ff . (Measure 6): ff , mp .

sx. (Measure 5): p . (Measure 6): ff , mp .

vc. (Measure 5): mf , p , ff . (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): p . (Measure 9): ff . (Measure 10): ff . (Measure 11): ff .

vc. (Measure 8): mf , p , ff . (Measure 9): ff , mp . (Measure 10): ff . (Measure 11): ff .

Measures 12-13: $15:16$, $21:20$.

[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

mf

28

3

4

3

fl.

sx.

vc.

fff

mp

ff

p

mp

ff

mp

ff

p

mp

ff

mp

ff

mp

ff

mf

ff

mp

ff

mf

ff

mf

ff

mf

ff

31

5

8

4

fl.

sx.

vc.

p

9.10

mp

fff

p

mp

fff

mp

ff

9.10

mf

mp

7.8

ff

33

18

fl. *mp*

sx. *p* *mp*

vc. *p* *mp*

fff *mf*

fff *mf*

34

98

68

fl. *p* *mp*

fff *mf*

fl. *p* *ff* *p*

13.12

13.12

fl. *p* *mp*

36

5

fl. *fff* *mf*

fff *mf*

fff *mf*

13.12

9.8

p *fff* *mf*

Four Ages of Sand ~5~ Evans

37

4

fl.

sx.

vc.

6

39

8

fl.

sx.

vc.

4

3

42

8

fl.

sx.

vc.

4

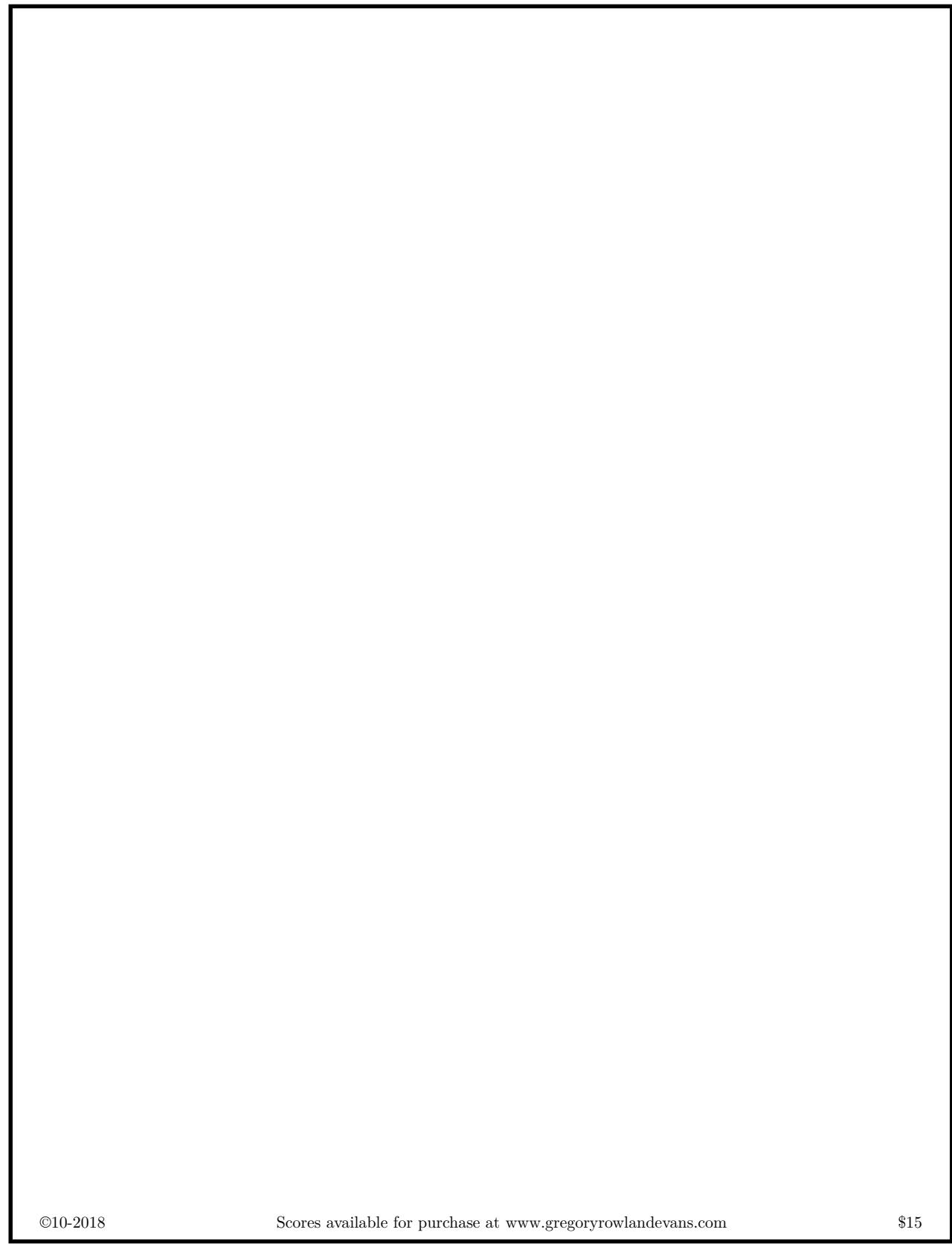
44

11

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

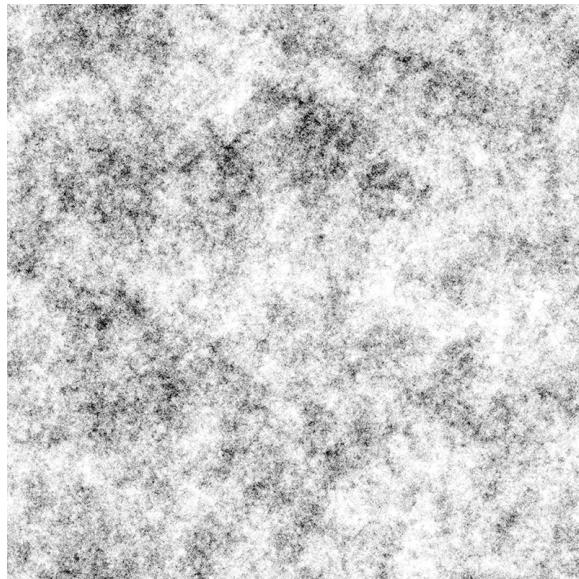


Figure B.1.1: A 2-dimensional random walk of 1000 iterations by Paul Bourke.

THIS THESIS WAS TYPESET using L^AT_EX, originally developed by Leslie Lamport and based on Donald Knuth's T_EX. 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/ or from the author at suchow@post.harvard.edu.