3-10-2020

Finding Music in Chaos: Designing and Composing with Virtual Instruments Inspired by Chaotic Equations

Landon P. Viator

FINDING MUSIC IN CHAOS: DESIGNING AND COMPOSING WITH VIRTUAL INSTRUMENTS INSPIRED BY CHAOTIC EQUATIONS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The School of Music

by
Landon P. Viator
B.M., University of Louisiana at Lafayette, 2015
M.M., University of Louisiana at Lafayette, 2017
May 2020

CONTENTS

ABSTRACT

Using chaos theory to design novel audio synthesis engines has been explored little in computer music. This could be because of the difficulty of obtaining harmonic tones or the likelihood of chaos-based synthesis engines to explode, which then requires re-instantiating of the engine to proceed with sound production. This process is not desirable when composing because of the time wasted fixing the synthesis engine instead of the composer being able to focus completely on the creative aspects of composition. One way to remedy these issues is to connect chaotic equations to individual parts of the synthesis engine instead of relying on the chaos as the primary source of all sound-producing procedures. To do this, one can create a physically-based synthesis model and connect chaotic equations to individual parts of the model.

The goal of this project is to design a physically-inspired virtual instrument based on a conceptual percussion instrument model that utilizes chaos theory in the synthesis engine to explore novel sounds in a reliable and repeatable way for other composers and performers to use. This project presents a two-movement composition utilizing these concepts and a modular set of virtual instruments that can be used by anyone, which can be interacted with by a new electronic music controller called the Hexapad controller and standard MIDI controllers. The physically-inspired instrument created for the Hexapad controller is called the Ambi-Drum and standard MIDI controllers are used to control synthesis parameters and other virtual instruments.

INTRODUCTION

To understand the concepts used in this project, a presentation of background information is necessary to understand concepts used in creating the virtual instruments for the presented composition:

- chaos theory - a field of mathematics that models and calculates chaotic, nonlinear dynamical systems
- physical modeling - an audio synthesis technique that digitally simulates the physics that control the behavior of physical vibrating objects such as strings, bells, and percussion instruments
- mapping - the process of connecting sound-producing sources with an interface for the user to interact within a range that fits the need of the interaction

A literature review is also presented which summarizes compositions and projects that have preceded the current project in the computer music paradigm whose techniques and processes were instrumental in inspiring the techniques and processes utilized in the current project.

Lastly, the composition presented by this project is described focusing on the following concepts:

- an overview of the Hexapad controller and how it interfaces with the virtual instrument
- the conceptual instrument model that the Hexapad-controlled virtual instrument is designed around
- how all the virtual instruments are implemented combining physical modeling techniques and chaos theory
- mapping methods used to control synthesis parameters that are appropriate for the virtual instruments

CHAPTER.1 A BRIEF OVERVIEW OF CHAOS THEORY, PHYSICAL MODELING, AND MAPPING

1.1. Chaos Theory

Chaos theory has been studied extensively for over a century and applied to various fields of study such as economics,[1] business,[2] psychology,[3] and music.[4] Chaos theory is a field of mathematics that models and calculates certain kinds of non-linear dynamical systems. A system is defined as dynamical if the system's state changes over time. Most natural systems are described as dynamical such as ecosystems, traffic patterns, human biology, and the focus of this section which is chaos. Many chaotic dynamical systems display highly sensitive and unpredictable behavior when the system's parameters or initial conditions are changed. Cartwright defines chaos as "order without predictability"[5] which is more easily visualized when viewed in contrast with the mapping of a sinusoidal oscillator in (x, y) space because the differences are immediately obvious. Figure 2.2 on the following page shows the mapping of a sinusoidal oscillator in (x, y) space.

The mapping of a sinusoidal oscillator in (x, y) space draws a perfect circle. This behavior is predictable and will always produce this image regardless of initial conditions. In contrast, Figure 1.1 on the following page shows the mapping of the Peter De Jong chaotic map in (x, y) space. The Peter De Jong chaotic map is specified by the following set of equations:[6]

[1]Mukul Majumdar, "Chaotic Dynamical Systems: An Introduction," *Economic Theory* 4, no. 5 (1994): 641–648, ISSN: 09382259, 14320479, http://www.jstor.org/stable/25054795.

[2]David Levy, "Chaos theory and strategy: Theory, application, and managerial implications," *Strategic management journal* 15, no. S2 (1994): 167–178.

[3]Frederick David Ed Abraham and Albert R Gilgen, *Chaos theory in psychology.* (Praeger Publishers/-Greenwood Publishing Group, 1995).

[4]Costantino Rizzuti, "Mapping chaotic dynamical systems into timbre evolution," in *In Proceedings of Sound and Music Computinc Conference (SMC)* (2007), 22–29.

[5]Timothy J Cartwright, "Planning and chaos theory," *Journal of the American Planning Association* 57, no. 1 (1991): 44–56.

[6]Clifford A Pickover, *The pattern book: Fractals, art, and nature* (Singapore: World Scientific, 1995).

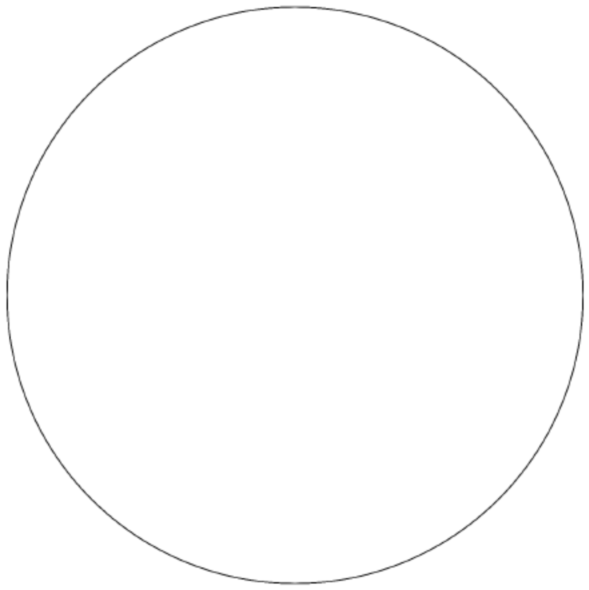


Figure 1.1. A representation of a sinusoidal oscillator state in (x, y) space.

$$x_{n+1} = sin(a * y_n) - cos(b * x_n) \tag{1.1}$$

$$y_{n+1} = sin(c * x_n) - cos(d * y_n) \tag{1.2}$$

The equations determine the dynamics of the system. In terms of (x, y) space, x and y represent coordinates at any given point in time. As the equation iterates each x and y value on the left side of the equation is fed back into the x and y values in the parentheses. Parameters a, b, c, and d are all variable parameters accepting any real numbers that can be changed to affect the behavior of the system and need to be initialized to some values. Since the parameters can be any real numbers, the parameter space is infinite. With certain sets of parameters it is possible for the behavior to be stable, but the edge of chaos can often be razor thin and difficult to control.[7]

[7]Edgar Berdahl et al., "Widening the Razor-Thin Edge of Chaos Into a Musical Highway: Connecting Chaotic Maps to Digital Waveguides," in *Proceedings of the International Conference on New Interfaces for Musical Expression, Blacksburg, Virginia, USA* (2018), 390–393.

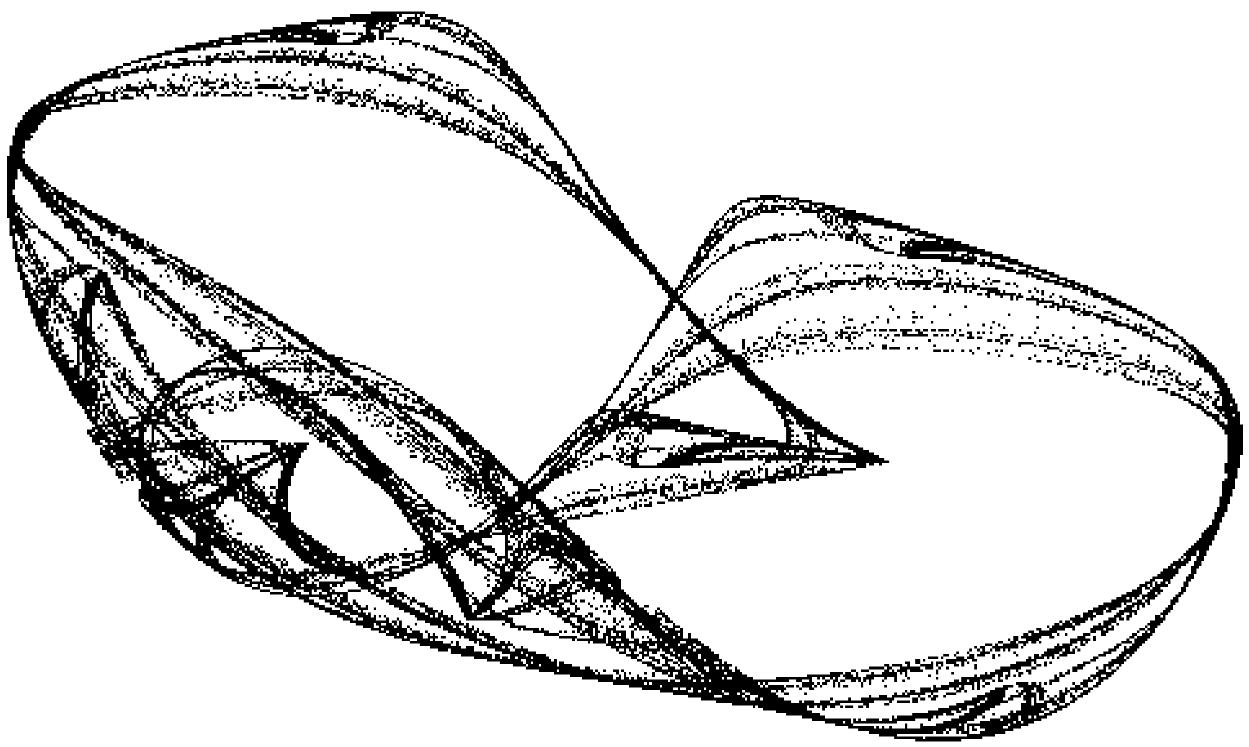


Figure 1.2. A representation of the Peter DeJong chaotic map in (x, y) space where a = 1.3, b = 1.4, c = 0.34, and d = 1.9.

It is important to note that deterministic chaos, which is the type of chaos the Peter De Jong equations describe, is not random. The equations set procedural trajectories and are deterministic according to the parameters passed. When applied to audio at the audio rate (44100 samples per second) or visually mapped such as Figure 1.2 above, it can look and sound random to a human being, but the chaos is following the rules set by the equations based on the initial conditions given or the changing of the parameters.

1.1.1. Circle Map

The Peter De Jong chaotic map is a two-dimensional chaotic map in (x, y) space. In contrast the circle map is a one-dimensional chaotic map. It is specified by the following equation:[8]

[8]M Høgh Jensen, Per Bak, and Tomas Bohr, "Complete devil's staircase, fractal dimension, and universality of mode-locking structure in the circle map," *Physical review letters* 50, no. 21 (1983): 1637.

$$\theta_{n+1} = (\theta_n + \Omega - (\kappa/(2\pi))sin(2\pi\theta_n))\%1.0 \tag{1.3}$$

The circle map contains a structure not found in the Peter De Jong map called an Arnold Tongue.[9] First defined by Andrey Kolmogorov, an Arnold Tongue is a dynamical system model used to describe driven mechanical motors. The circle map can also display phase-locking which means in certain parameter space the map is locked to its driving frequency which is parameter Ω. This means that the phase of the output is equal to the phase of the input, which does not allow phase shifting between the input and output signal. Parameter κ is the coupling strength which is the strength of the force exerted on the system. The standard implementation of the circle map states that θ should be between the values of 0.0 and 1.0 and it represents the phase of an oscillator. Extracting the θ value at any given point in time should be taken modulo 1.0 before the next calculated point. This is done so that as many values for θ are calculated before the map can reach a θ value of zero.

In practice, when parameter $\kappa = 0.0$, the entire section of the equation $\kappa/2\pi$ evaluates to 0.0, so when $\kappa = 0.0$ Ω becomes the controlling entity of the non-linear iteration through the phase of the resulting oscillation. Because of this behavior, κ determines the non-linear behavior of the equation. If $\kappa \neq 0.0$n then Ω can potentially influence the pitch of the oscillation. At the lowest values of κ ($\kappa > 0.0$) the oscillator is bright and tonal in its timbral musical properties, but as κ approaches the largest values of κ ($\kappa < 1.0$) the circle map tends to move toward chaos in a seemingly random path, but still deterministic giving the oscillator a noisy quality. At middle values of κ the oscillation can fail and get stuck on a strange setting. This poses a significant issue of using the map in creative practice such as music composition. To use this map in audio these intermediate κ values need to be avoided if the desired effect is continuous and predictable oscillation.

[9]Philip L Boyland, "Bifurcations of circle maps: Arnol'd tongues, bistability and rotation intervals," *Communications in Mathematical Physics* 106, no. 3 (1986): 353–381.

The circle map is an example of a chaotic map that operates on discrete time. Discrete time systems view parameters as separate distinct moments of time.[10] An analogy that helps to conceptualize this is the process of viewing video. The standard frame rate of cinematic video production is about 30 frames per second. Every second that passes by contains 30 individual consecutive images. There are moments lost between each image if you imagine the motion of a swing. At frame one the swing could be at position $x_n = 1.0$. At frame two the swing could be at position $x_n = 2.0$. The only two frames available are at $x_n = [1, 2]$ but in reality there are an infinite amount of values between 1 and 2 in the swing's trajectory that are not accounted for because of the limit of the frame rate. Therefore, discrete time systems provide a limited view of parameter space.

1.1.2. Hindmarsh-Rose Model

The Hindmarsh-Rose model is the last chaotic dynamical system explored in this project. It was first described by Hindmarsh and Rose in 1982.[11] This model is an adaptation of a previously described model by Fitzhugh.[12] This model describes the behavior of a brain cell discovered in a pond snail that when depolarized by a short pulse of current an action potential (burst) was created with a short after potential decay.[13] To produce the behavior of the cell's bursting behavior the introduction of a small electrical current is used. Figure 1.3 on the following page shows the bursting behavior. The Hindmarsh-Rose model is described by the following differential equations:[14]

[10]Maurício C De Oliveira, Jacques Bernussou, and José C Geromel, "A new discrete-time robust stability condition," *Systems & control letters* 37, no. 4 (1999): 261–265.

[11]James L Hindmarsh and RM Rose, "A model of neuronal bursting using three coupled first order differential equations," *Proceedings of the Royal society of London. Series B. Biological sciences* 221, no. 1222 (1984): 87–102.

[12]FitzHugh, Richard. "Impulses and physiological states in theoretical models of nerve membrane." Biophysical journal 1, no. 6 (1961): 445-466.

[13]STUART H Thompson and STEPHEN J Smith, "Depolarizing afterpotentials and burst production in molluscan pacemaker neurons," *Journal of Neurophysiology* 39, no. 1 (1976): 153–161.

[14]Hindmarsh and Rose, "A model of neuronal bursting using three coupled first order differential equations."

$$\dot{x} = y - ax^3 + bx^2 + I - z \tag{1.4}$$

$$\dot{y} = c - dx^2 - y \tag{1.5}$$

$$\dot{z} = r(s(x - x_1) - z) \tag{1.6}$$

Parameters a, b, c, and d are constants where a = 1, b = 3, c = 1, and d = 5 so that the model is easily graphed. Parameters r and s are also constants that control the bursting behavior of the model. When r = 0.001 and s = 1 the model displays the bursting behavior described previously in the cell. Lastly, I is a variable parameter that represents the applied current across the cell. At small levels such as $I < 1$ the model displays an individual burst followed by a decaying wave which slowly moves to the starting position. At larger levels of current like $I < 2$ the model displays periodic bursts which lower in frequency as the model progresses.

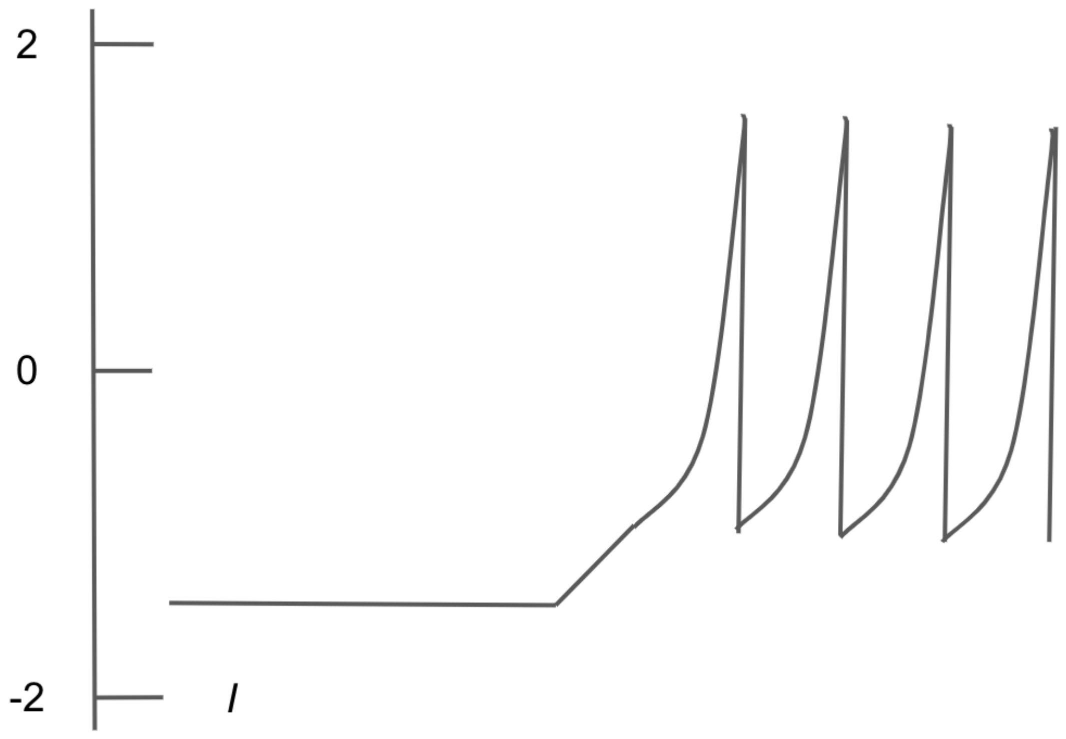


Figure 1.3. A diagram showing the bursting behavior of the Hindmarsh-Rose equations.

Edge Of Chaos

There is a term associated with the exploration of chaotic systems in musical contexts called the "edge of chaos."[15] This refers to the shifting between a stable and unstable state in the chaotic system. The "edge" refers to the position at which the change from stable to unstable occurs. As observed by Berdahl,[16] Bell,[17] Rodet,[18] and others, this edge is often incredibly small and navigating this area can be difficult. Minuscule changes in the system create disproportionately significant effects. This concept is interesting musically because the interpolation between stable and unstable behavior can have a very rich and vibrant sonic character. There are also musical parallels that can be derived from this concept of the edge of chaos that is not based strictly in technology or programming concepts. The edge of chaos is similar to the musical ideas of tension and release. Tension creates the need for resolution and release is the resolution. It is difficult to quantify what exactly musical tension is, but one usually recognizes it when they hear it. The edge of chaos is similar because there is not a clear-cut definition on what is stable or what is unstable. When listening to chaos in a musical context, defining the edge between stable and unstable behavior is often an interpolation of both ends of the spectrum much like the difference between tension and release. For example, as the system starts to approach an unstable state, it exhibits both stable and unstable characteristics such as harmonic/inharmonic tones and distortion. The unstable behavior can begin as infrequent crackles and pops or bursts of noise, but when the system turns completely unstable it is usually completely saturated with noise and distortion. Similarly, musical tension can use the same musical material as the material creating the

[15]Berdahl et al., "Widening the Razor-Thin Edge of Chaos Into a Musical Highway: Connecting Chaotic Maps to Digital Waveguides."

[16]Ibid.

[17]Shawn Bell and Liane Gabora, "A music-generating system inspired by the science of complex adaptive systems," *arXiv preprint arXiv:1610.02475*, 2016,

[18]Xavier Rodet, "Nonlinear oscillations in sustained musical instruments: Models and control," *Proc. Euromech, Hamburg, Germany*, 1993,

release effect such as the same motives, melodies, harmonies, instrumentation, tonal centers, etc. but emphasizing certain emotive characteristics.

1.1.3. Conclusion

Background information about chaotic non-linear dynamical systems is necessary to understand how these types of systems behave and how they have been researched previously. The circle map is a one-dimensional discrete-time chaotic map while the Hindmarsh-Rose model is a continuous-time chaotic system. Both chaotic systems display unique behavior and can be exploited for musical purposes like audio synthesis. This is because of their ability to be implemented as audio oscillators and as chaotic parameter generators. The circle map and Hindmarsh-Rose model are used throughout the rest of this project as sound material for musical composition and audio synthesis. The circle map is used as a low drone by sending noise through a circle map implementation and a variation of the Hindmarsh-Rose equations are implemented as an oscillator and used as a noise source for synthesis engines.

1.2. Physical Modeling

Physical modeling is a type of audio synthesis that digitally simulates the physics that control the behavior of physical vibrating objects such as strings, bells, or percussion instruments. These simulations are calculated by the mathematical formulas that create the same physical behaviors as these objects.[19]. These objects are modeled to behave like the instrument they are modeling, so different parts of the equations that produce the physical model can be manipulated to change the behavior of the instruments based on the desired response. For example, when playing a string instrument it can be plucked to produce a long tone. The string can also be damped and plucked to play a short tone. In physical modeling, there are ways to manipulate the equation to also simulate long tones or short tones.[20]

The main advantage to creating physical models of instruments as opposed to other methods of synthesis is that physical modeling attempts to acquire the same expressive and dynamic qualities inherent in real instruments and allows for intuitive control of the physical phenomena in those models.[21] The main issue with these concepts is that the interface between the performer and the physically modeled virtual instruments (the software) must be as intuitive to use as the real instrument being modeled, but still allow the fine and almost infinite control that the digital realm offers.

The majority of the literature on physical modeling synthesis includes the mathematical examples used in each article, but for the most part lacks practical examples such as program

[19]Sheffield, Eric, "Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments" (2019) *LSU Doctoral Dissertations.* 4918. https://digitalcommons.lsu.edu/gradschool$_{d}$*issertations*/4918

[20]Claude Cadoz, Annie Luciani, and Jean Loup Florens, "CORDIS-ANIMA: a Modeling and simulation system for sound and image synthesis: the general formalism," *Computer music journal* 17, no. 1 (1993): 19–29.

[21]Julius O Smith, "Physical modeling synthesis update," *Computer Music Journal* 20, no. 2 (1996): 44–56.

code to build the models or the visual patching code used in programs like MaxMSP[22] and Pure Data.[23] This document displays every discussed audio example in figures containing MaxMSP patches to allow easy implementation of the discussed audio processing.

1.2.1. Digital Waveguide Synthesis

Digital waveguide synthesis takes a different path to determine the physical model. It solves wave equations to simulate waves traveling through a physical medium.[24] This method is more efficient computationally because previous models require many points of multiplication or addition to calculate the model, but waveguides use only a delay line to simulate waves traveling between two points. In the simulation of this wave movement, the wave travels from one end of the modeled string to the other by one sample of space for every sample of time. This is why the delay lines are the best choice for this type of simulation. The Karplus-Strong algorithm shown in Figure 1.3 is an example of using a digital delay line to simulate a wave traveling through a string.

The Karplus-Strong algorithm was first published by Kevin Karplus and Alex Strong in 1983 to model plucked strings and percussive timbres.[25] This method of physical modeling is an important predecessor to modern physical modeling techniques. This is because it was a viable digital model that required low computing power which allowed it to be implemented effectively on earlier microprocessors before the advent of the powerful laptops. This algorithm uses the averaging of two successive samples of audio which produces a slow decay of the audio waveform. This algorithm also produces a tone whose pitch is controlled by the delay time which is usually in milliseconds.

[22] "MaxMSP," accessed September 1, 2019, https://cycling74.com.

[23] "Pure Data," accessed September 1, 2019, https://puredata.info.

[24] Julius O Smith, "Physical modeling using digital waveguides," *Computer music journal* 16 JSTOR, no. 4 (1992): 74–91.

[25] Kevin Karplus and Alex Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal* 7, no. 2 (1983): 43–55.

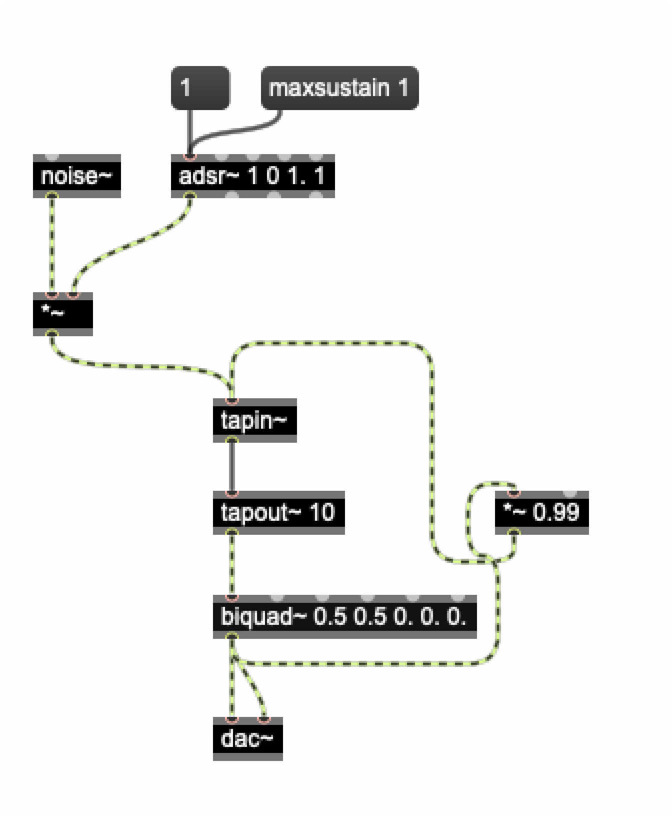


Figure 1.4. Implementation of the Karplus-Strong Algorithm in MaxMSP.

The implementation in Figure 1.4 above uses a burst of noise to excite the model and begin the slow wave decay sound. The `tapin` and `tapout` objects act as the delay line. The input is sent into the `tapin` and then delayed by the `tapout` by the value passed into it as an argument. In this case it is delayed by 10 milliseconds. The resulting signal is passed through the biquad object which is a one-zero filter to reduce harsh high frequencies that would not be as prominent in a real string instrument. The now filtered signal is sent into the input of the same delay line multiplied by a number less than one to avoid a feed-back instability. This final processed signal is sent to the digital audio converter (DAC) along with the initial burst of noise controlled by an envelope object to simulate the initial pluck sound. Digital waveguide synthesis is not limited to string models.

1.2.2. Modal Synthesis

When solid objects are struck like percussion instruments there is a propagation of force throughout the body of the object originating from the point of initial contact. Modal synthesis is a method of physical modeling that uses groups of oscillators encapsulated into a conceptual container to simulate the propagation of force throughout a vibrating object.[26] The oscillators represent the vibrating object excited by an external stimulus or internal resonators excited by an external stimulus called the force model. There are many ways to excite an object such as striking, scraping, and plucking. All of these types of stimuli can be used as parameters in modal models and in the case of real-time synthesis for applications such as video games the external stimulus may be unknown until someone or something interacts with the model.[27] Figure 1.5 below shows a conceptual modal model for a resonating metal box with four resonance frequencies.

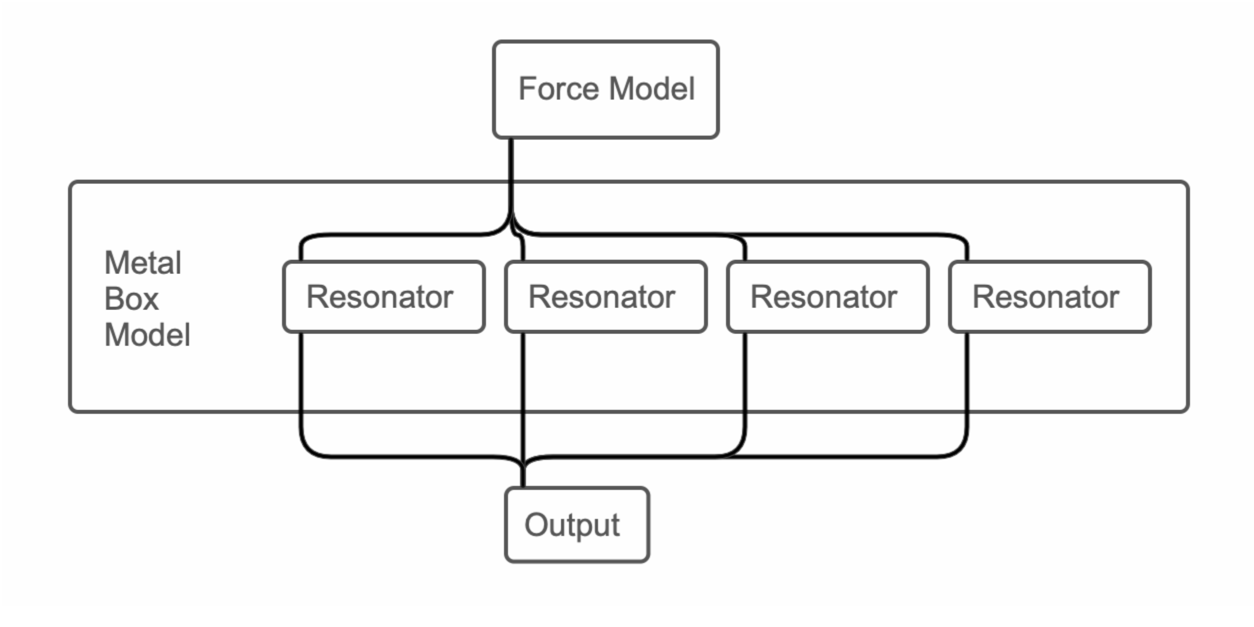


Figure 1.5. Conceptual figure of a modal model of a resonating metal box.

[26]Pirouz Djoharian, "Generating Models for Modal Synthesis," *Computer Music Journal* 17, no. 1 (1993): 57–65, ISSN: 01489267, 15315169, http://www.jstor.org/stable/3680570.

[27]Kees Van Den Doel and Dinesh K Pai, "Modal synthesis for vibrating objects," *Audio Anectodes. AK Peter, Natick, MA*, 2003, 1–8.

In this modal, model a resonating metal box is conceptualized. The force model begins the excitation stimulus which could be anything, but a solid striking object is an appropriate example. The object strikes the metal box exciting each resonator which propagates vibrations throughout the metal box. These resonators could be replaced with any type of oscillator to accomplish a similar outcome, but the main point is to have some kind of sound-producing process that the force model excites. All of the resulting waves are summed together to the output which could then be multiplied by some coefficient in some models, but this simple example omits this coefficient.

Modal modeling is an incredibly powerful tool to model conceptual instruments without having to physically fabricate or find suitable examples in the real world. In the above mentioned example, a metal box struck with a solid object was conceptualized, but the number of possible conceptualizations that can be achieved with modal modeling is infinite. There are many instruments that have not been physically modeled or that have not been effectively modeled with other physical modeling techniques. Modal modeling serves as a viable method of modeling any type of instrument real or conceptual and the current project utilizes modal synthesis to realize a conceptual percussion instrument with resonators such as shown above in Figure 1.4 and digital waveguides.

1.3. Mapping

When designing virtual instruments and physical models of instruments, there are many parameters that affect the resulting audio output of the synthesis engine as well as internal sections of the audio path. If this thought is compared to a standard acoustic instrument there are parallels that can be drawn between them. A violin has many parameters that mirror an audio synthesis engine. There is a volume parameter which is controlled by the pressure of the bow applied by the player where as it could be a knob or dial in a synthesis engine. There is a timbre parameter controlled by many factors, but primarily by the player bowing at different positions with different angles of bowing on the strings and by using different parts of the bow to make contact with the strings. In a synthesis engine, this could be realized with an audio filter control or perhaps a set of filter controls for different types of filters and different parameters for each. Then, there is all of the non-traditional prepared playing and extended techniques[28] such as using non-bow objects on the strings, striking the body of the violin, or bowing with extreme amounts of pressure which can also be re-created in synthesis engines in any way that achieves a similar sound to the acoustic parallel. All of these controls make up the parameters for the synthesis engine/physical model being considered to model an acoustic instrument. The process of mapping connects the intent of the synthesis engine designer's conceptual vision to the real parameters that control the behavior of the virtual instrument. This is realized by either mirroring the interface of the acoustic instrument or not, and how well it describes the behavior of the acoustic instrument through the performance of the synthesis engine.

When considering an acoustic instrument, the interface for interaction and the source of the sound production are usually the same mechanism. For a violin, the strings both produce the sound and are the interfaces to interact with the sound production. Hunt et al. suggest that the same is not true for virtual instruments.[29] The interface in which the performer interacts with the virtual instrument is a separate procedure from the source of the sound production. For example, an oscillator produces sound waves, but the performer interacts with the oscillator in a seemingly separate procedure such as moving a slider or turning a dial. Mapping is the process of connecting the sound-producing source with the interface for the performer to interact with in a range that fits the need of the interaction. An example of mapping in MaxMSP is shown above in Figure 1.6 where a dial whose default range is 0 - 127 is mapped to the new range of 0 - 20000 and a slider whose default range is also 0 - 127 is mapped to the new range of 0 - 60.

[28]Krewer, Mariana, "Extended Techniques for Intermediate Violin Students" (2018). *LSU Doctoral Dissertations.* 4196. https://digitalcommons.lsu.edu/gradschool$_{d}$*issertations*/4196

[29]Andy Hunt, Marcelo M Wanderley, and Matthew Paradis, "The importance of parameter mapping in electronic instrument design," *Journal of New Music Research* 32, no. 4 (2003): 429–440.

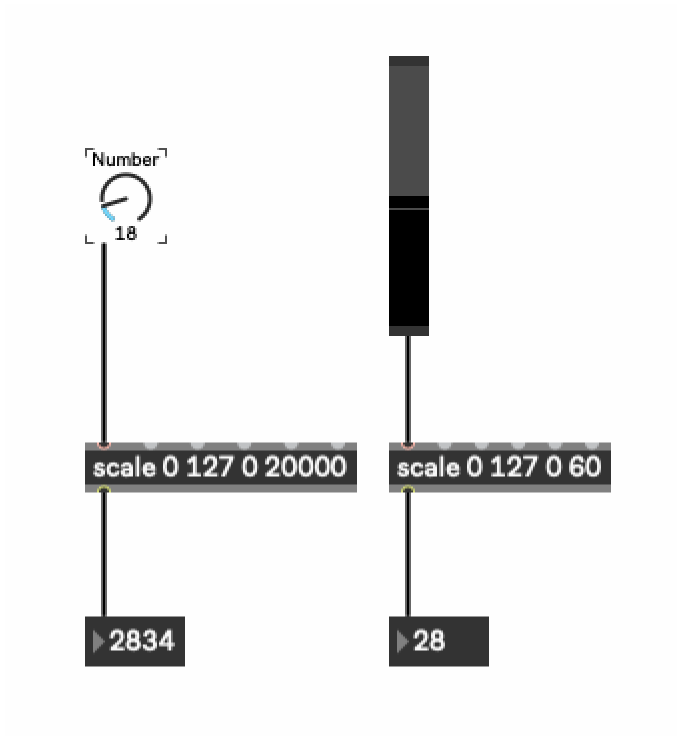


Figure 1.6. Simple example of mapping in MaxMSP.

Choosing A Mapping Method

Hunt et al. suggest that the process of deciding how to proceed with mapping in respect to virtual instrument parameters filters down into the two following choices:[30]

- procedural processes
- explicit performer-defined

The main difference between these two methods is that the procedural option such as neural networks or genetic algorithms uses a procedure to map parameters in an artificially intelligent manner as opposed to having the performer retain direct control of the parameter mapping.

[30]Andy Hunt and Marcelo M Wanderley, "MHunt, Andy, and Marcelo M. Wanderley. "Mapping performer parameters to synthesis engines." Organised sound 7, no. 2 (2002): 97-108.," *Organised Sound* 7, no. 2 (2002): 97–108.

Procedural Mapping

The procedure in the case of the neural network is powered by mathematical algorithms that analyze data and use the data to "learn" how to proceed forward in decision making without a human directly controlling the decision-making process or the procedure past the initial conditions phase. For the case of the genetic algorithm, it's a similar process. Mathematical algorithms analyze data, but a genetic algorithm models evolutionary biology and assigns fitness scores to the data, uses the best fit data to produce a new population of data using a mutation model, and then repeats the process until up to 100% fit data point/data group is produced.

Lee et al. suggested a multi-dimensional neural network to control audio synthesis by analyzing data from a MIDI keyboard controller and using that data to teach the neural network to dynamically control timbre parameters in a synthesis engine. This is an example of a procedural mapping process.[31]

Fels et al. proposed an adaptive neural network interface in the shape of a glove worn on the user's hand that creates and adapts a mapping process based on a training phase of data collected from the performer. This type of adaptive mapping in neural networks is based on the following three features:[32]

- user demonstration of which inputs should lead to which outputs to train the neural network
- new user-provided data to retrain the neural network
- once trained, the neural network runs rapidly and efficiently

With these aspects of neural networks in mind, the authors created an interface in the shape of a glove that is worn by the performer to map hand gestures to a speech synthesis

[31]Matthew Lee and David Wessel, "Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms. ICMC, San Jose," in *Proceedings of the International Computer Music Conference* (International Computer Music Association., 1992), 277–277.

[32]Sidney S Fels and Geoffrey E Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," *IEEE transactions on Neural Networks* 4, no. 1 (1993): 2–8.

engine. This glove is named the Glove-TalkII. The goal of the glove is to allow the performer to control a granular synthesis model that plays back speech files containing 203 words from the English language according to the gestures made by the performer's hand movement. Five neural networks are trained; one for each set of sub-tasks required by the performer's interpretation of controlling the speech synthesis model and produces 8036 levels of weight data used to label each entry of training data.[33] The Glove-TalkII showed high success in synthesizing speech. Only 1% of the synthesized words were incorrect and only 5% of words attempted failed to be synthesized due to gestures not being properly recognized.

1.3.1. Explicit Mapping

Explicit mapping is defined as a mapping method where a user retains direct control of the parameters that control a synthesis model unlike in procedural mapping described previously. Hunt suggests that many performer-controlled virtual instruments are explicitly mapped. These are often considered to be a few-to-many relationship.[34] This few-to-many concept is a type of mapping that is defined as mapping a few performer-controlled parameters to control large amounts of parameters inside of a given synthesis engine. An example of this would be a large array of physically-modeled strings whose delay times are set by one control parameter that the user has access to. By placing a unique scaling equation between each delay line and the user's control each delay line would be mapped to a different and unique parameter space. The user would be changing one control source which then maps each delay line to a specific parameter space. When considering this type of relationship for any two groups of parameters any synthesis engine has any of the three following parameter relationships available:[35]

[33]Fels and Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer."

[34]Hunt and Wanderley, "MHunt, Andy, and Marcelo M. Wanderley. "Mapping performer parameters to synthesis engines." Organised sound 7, no. 2 (2002): 97-108."

[35]Guy E Garnett and Camille Goudeseune, "Performance Factors in Control of High-Dimensional Space.," in *ICMC* (1999).

- one-to-one
- many-to-one
- one-to-many

The one-to-one relationship is defined as a mapping method where one performer parameter controls one synthesis engine parameter at any given time. The many-to-one relationship is defined as a mapping method where many performer parameters control one synthesis engine parameter at any given time. Lastly, the one-to-many relationship is defined as a mapping method where one performer parameter controls many synthesis engine parameters at any given time which is the most common form found in the literature on parameter mapping.

CHAPTER.2
CHAOS THEORY, PHYSICAL MODELING, AND MAPPING: A LITERATURE REVIEW

2.1. Chaos Theory In Music

Chaotic systems have been used in musical applications by controlling numerous parameters at the control rate like delay time values, number generators, and frequency values. Continuous time chaotic systems based on differential equations have been used as oscillators to produce unique and interesting sound synthesis which can then be augmented by DSP models such as amplitude modulation, frequency modulation, and ring modulation.

2.1.1. Real-Time Chua Oscillator

Chua's circuit is a thoroughly researched chaotic system.[1] Choi explored the musical qualities of the oscillation achieved by Chua's circuit.[2] Chua's circuit (which is an analog electrical circuit) is specified by the following three differential equations.[3]

$$\frac{dv_1}{dt} = \frac{1}{C_1}[G(v_2 - v_1) - g(v_1)] \tag{2.7}$$

$$\frac{dv_2}{dt} = \frac{1}{C_2}[G(v_1 - v_2) + i_3] \tag{2.8}$$

$$\frac{dv_3}{dt} = \frac{1}{L_2} * v_2 \tag{2.9}$$

C_1 and C_2 are capacitors and L is an inductor which are all passed a voltage current. v_1, v_2, and i_3 are three discrete signal paths represented by each equation. The non-linear

[1]Takashi Matsumoto, "A chaotic attractor from Chua's circuit," *IEEE Transactions on Circuits and Systems* 31, no. 12 (1984): 1055–1058.

[2]Insook Choi, "Interactive exploration of a chaotic oscillator for generating musical signals in real-time concert performance," *Journal of the Franklin Institute* 331, no. 6 (1994): 785–818.

[3]Matsumoto, "A chaotic attractor from Chua's circuit."

character of Chua's circuit is found in the function of v_1, $g(v_1)$ given by Chua's non-linear function $g(v_1)$.

Chua's circuit contains many control parameters as seen above and many experiments have been conducted with digital implementations of it. Zhong found that $C_2 = 10$ nF and L = 57.4mH put the oscillation frequencies of Chua's circuit into the normal human hearing range. These parameter values put the oscillation frequencies close to the normal frequency range of the piano which is between 27.5hz to 4,186hz. The Chua circuit frequency range from 35hz to 15khz.[4]

The ability to control chaotic oscillation and constrain their frequency response to an audible range makes their use in musical applications feasible.

2.1.2. Chaotic Oscillators In Visual Programming Environments

Yadegari explored the process of generating audio synthesis by finding numerical solutions to differential equations that define continuous time chaotic systems.[5] Solving these differential equations produced oscillations at the audio rate which Yadegari analyzed and plotted three-dimensional graphs for several continuous time chaotic systems. Yadegari implemented the equations in Pure Data with objects such as the `expr` object. This object allows the user to write out mathematical expressions which are solved by the program and allows the user to change variable parameters in real time. Yadegari began with the well-documented and extensively researched continuous-time chaotic system published by Edward Lorenz specified by the following set of differential equations:[6]

[4]Choi, "Interactive exploration of a chaotic oscillator for generating musical signals in real-time concert performance."

[5]Shahrokh Yadegari, "Chaotic signal synthesis with real-time control: solving differential equations in PD, MAX/MSP, and JMAX," in *Proceedings of the 6th International Conference on Digital Audio Effects* (2003).

[6]Edward N Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences* 20, no. 2 (1963): 130–141.

$$\dot{X} = Pr(Y - X) \tag{2.10}$$

$$\dot{Y} = -XZ + rX - Y \tag{2.11}$$

$$\dot{Z} = XY - bZ \tag{2.12}$$

The variables b, r, and Pr are variable control parameters that accept any real numbers which affects the behavior of the oscillation. Parameters X, Y, and Z are the unknown values that are solved for, which generates oscillation and are the initial points for the equations in (x, y, z) space. To solve for X, Y, and Z, the equation takes on the following form:[7]

$$X_{n+1} = X_n + (Pr(Y_n - X_n))\Delta t \tag{2.13}$$

$$Y_{n+1} = Y_n + (-X_n Z_n + rX_n - Y_n)\Delta t \tag{2.14}$$

$$Z_{n+1} = Z_n + (X_n Y_n - bZ_n)\Delta t \tag{2.15}$$

Yadegari found that the implementation of these differential equations yielded oscillations that interpolated between being stable and being unstable in its phase. As discussed earlier, if the parameter space is any real parameters the parameter space is infinite. Yadegari does not specify a way to explore this parameter space, but he did find a set of parameter values that he was able to analyze as an example. The parameters that achieved this result were when Pr = 10, b = 2.66667, r = 18, dt = 0.01 (rate of change in respect to time), and the initial values for X, Y, and Z were 0, 2.3, and -4.4 respectively.

2.1.3. Chaotic Mapping

Cupellini et al. explored the mapping of chaotic systems to control parameters such as filter and spatial parameters for synthesis models that utilized Chua's circuit for the chaotic

[7]Lorenz, "Deterministic nonperiodic flow."

model.[8] The authors suggest scaling the oscillator signal generated by Chua's circuit by a factor that results in the frequency output being forced between 0-20Hz to be used as an low frequency oscillator (LFO) which is a typical control parameter in commercial hardware and software synthesizers. The authors suggest that the resulting LFO can be further manipulated and scaled to fit the needs of control paramters such as filter controls, volume controls, arpeggiation speed, and other typical synthesizer controls. Because the control signal is created by a chaotic system, the authors suggest that the inherent irregularities in the signal are well-suited for musical applications because it is less rigid than and more natural sounding than an LFO created by a sine, square, triangle, or saw wave.

2.1.4. Chaotic Sound Spatialization

Soria et al. suggest a framework for electroacoustic composers to use chaotic systems to create panning trajectories to spatialize sound across large speaker arrays.[9] These trajectories are calculated using the logistic equation. The logistic equation's discrete form is defined as:[10]

$$x_n = rx_{n-1}(1 - x_{n-1}) \tag{2.16}$$

The functional form is defined as:[11]

$$f(x) = rx(1 - x) \tag{2.17}$$

[8]Enrico Cupellini et al., "Exploring Musical mappings and Generating Accompaniment with Chaotic Systems.," in *ICMC* (2008).

[9]Edmar Soria and Roberto Morales-Manzanares, "Multidimensional sound spatialization by means of chaotic dynamical systems.," in *NIME*, vol. 13 (2013), 79–83.

[10]Ibid.

[11]Ibid.

The whole panning orbit is the set:[12]

$$O(x) = \{x_i : ie[1...n]\} \tag{2.18}$$

The authors first designed an algorithm that evaluates the logistic equations for any set of initial conditions and range of values. Then, they wrote a series of algorithms to derive multiple panning orbits from a singular evaluation of the above panning equation.

2.1.5. Connecting Chaotic Systems To Digital Waveguides

Berdahl et al investigated multiple discrete-time chaotic systems to create new musical instruments.[13] The authors note the difficulty navigating the edge of chaos because it can be "razor-thin."[14] Because of this, the authors suggest connecting chaotic systems to digital waveguides to synthesize harmonic tones more easily and to reduce the likelihood of falling off of the edge of chaos. The chaotic systems that the authors investigated were the Peter De Jong map, tinkerbell map, circle map, and standard map. The authors note that all of these systems output the most musically interesting sounds while on the edge of chaos. Changing parameters in the equations that define the chaotic systems allowed the authors to explore complex evolving timbres. The authors suggest that any chaotic system in the following form can easily be connected to a digital waveguide:[15]

$$V_n = \phi(V_{n-L}) \tag{2.19}$$

where V_{n-L} refers to an earlier step in time, ϕ is the chaotic system, and V_n is the final output. Figure 2.7 shows the block diagram of a chaotic system connected to a digital

[12]Soria and Morales-Manzanares, "Multidimensional sound spatialization by means of chaotic dynamical systems."

[13]Berdahl et al., "Widening the Razor-Thin Edge of Chaos Into a Musical Highway: Connecting Chaotic Maps to Digital Waveguides."

[14]Ibid.

[15]Ibid.

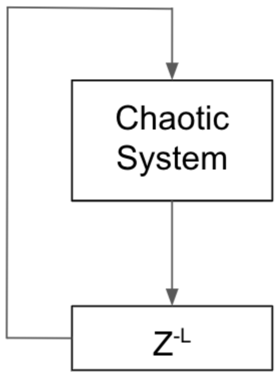


Figure 2.7. Block diagram of a chaotic system connected to a digital waveguide.

waveguide. In this diagram, the output of a chaotic system is fed into a delay line and then the output of the delay line is fed back into the input of the chaotic map. Adjusting the parameters of the chaotic map ϕ allows the user to explore timbre space in the chaotic system and find appropriate ways to map these parameters to user controls to create a musical instrument.

Limiting Instability

As mentioned earlier, some chaotic systems have a boundary beyond which the system goes unstable. One way to avoid this is to bound[16] the chaotic system ϕ with a bounding function such as sin or cos. Another way would be to use a clipping function in the feedback section of the entire system. In MaxMSP this can be easily achieved with objects such as `clip`, `sinx`, and `cosx`. The author points out that adding a clipping function to the tinkerbell map worked well to limit the instability. These chaotic maps also have the possibility of getting stuck at a certain point where audio output is halted at a dc offset or the audio output goes unstable and no longer responds to parameter changes. The authors note that this occurred in the tinkerbell map when all parameters were set to 0. To remedy this, the authors added a small amount of noise to the feedback section to prevent simultaneous instances of 0.

[16]Shankar Sastry, *Nonlinear systems: analysis, stability, and control*, vol. 10 (Springer Science & Business Media, 2013).

2.2. Physical Modeling In Music

2.2.1. *MOSAIC*

MOSAIC is a physical modeling framework that facilitates modal synthesis.[17] *MOSAIC* contains instruments based off of collections of resonant structures that interact with each other by vibrating and exciting the neighboring air. These resonant structures are defined by modal data which represents wooden instrument bodies, bridges, tubes, bells, percussion membranes, strings, and other such vibrating structures. *MOSAIC* also contains a special function called *connections* that allows the user to connect different structures together so that they interact with one another. The connection types are defined by such labels as adhere, strike, bow, pluck, and other actions. To facilitate excitation and control data for these structures, *MOSAIC* contains *controller objects* which allows the user to interact with synthesis parameters. Lastly, sound is achieved by the user placing *virtual pickups* on the created structures to listen to their audio output.

Morrison notes the challenges inherent in controlling parameters in physical modeling because of how large of a data set is needed to specify control information. Some of these go far beyond the typical synthesis parameters such as pitch, envelope, etc. and includes such things as the force applied to bow a string, the force of air traveling through a wind instrument, and the position of a guitar player's finger on the fret board. Morrison states that *MOSAIC* does not intend to solve these control issues directly, but allows the user to easily specify setups for their controls which can then be grouped together and used at other parts of the synthesis model.

MOSAIC contains methods written in c++ that implement the desired physical model by calling the method associated with that model. For example, Figure 2.8 shows a snippet of code that calls a method to create a string model.

[17]Joseph Derek Morrison and Jean-Marie Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal* 17, no. 1 (1993): 45–56.

```
(define my-string
  (make-object  'bi-string
    (modes 40)
    (length .5)
    (tension 150)
    (density 1000)
    (radius .001)
  )
)
```

Figure 2.8. An example C++ method that creates a string model when called.

This method creates a string model that defines the number of modes on the string, the length in meters, the tensions of the string in Newtons, the density of the string in kilograms per cubic meter, and the radius of the string in meters.

2.2.2. ETabla

The *ETabla* is a electronic controller inspired by Tabla drums created by Kapur et al.[18] Tabla are a pair traditional hand drums from the North Indian region and are expressive instruments. There are multiple versions of the drums that are tuned to different pitches such as the Dahana and the Bayan. Each can have their pitch manipulated during performance and each make various types of sounds depending on how the player strikes the drum. The goal of the *ETabla* was to create one electronic controller that allows the user to control a physical model that simulates the sound of each of these drums on one device. Due to the amount of different stroke styles that are used to play the drums and different pitch characteristics, it was a difficult controller to build and a difficult physical model to realize.

To input the various types of data from types of strokes striking the drums such as finger playing, square force sensing resistors were used to input the finger strike forces and long force sensing resistors were used to input the position of the finger strikes. All of this data

[18]Ajay Kapur et al., "The electronic tabla controller," *Journal of New Music Research* 32, no. 4 (2003): 351–359.

is converted to MIDI data and can be used for any purpose, but for the *ETabla* the MIDI data is used to control a physical model synthesis engine.

To design the physical model of the drums banded waveguides were used to simulate a drum membrane. Banded waveguides simulate the behavior of waves dispersing through a membrane by splitting the waves into discrete frequency bands.

The *ETabla* proved its practical value when it premiered in in classical and ritual songs associated with North Indian traditional music which used an atmospheric sound-scape instead of a percussive setting.

2.2.3. *Rhythm'n'Shoes*

The *Rhythm'n'Shoes* is a musical interface created to control virtual percussion instruments in the form of wearable shoes on the user's feet.[19] The user can tap their feet to generate gestures and data for use with virtual percussion instruments. The authors argue that using one's feet to create musical gestures gives the user spontaneous and expressive control. The *Rhythm'n'Shoes* contain four sensors that pick up data from the user's actions and send the data by wireless means to the main user's computer. The sensors are connected to the Arduino Duemilanove Board's analog inputs to sample the input and then sent to the wireless transceiver for wireless transmission. All of the necessary components are housed in a back pack worn by the user with cables connecting the sensors to the pack. The *Rhythm'n'Shoes* also have MIDI and OSC capabilities so that the *Rhythm'n'Shoes* can be used for alternative electronic/virtual instruments. The software used to parse all of the input data is realized in Pd as three modules: tapping detection and converting a tap into a measure of its amplitude, mapping the data to MIDI, OSC, or the synthesis model parameters, and a physically-based percussive impact model which utilizes the detected tapping events. To accurately capture tapping gestures, the signal is fed through a threshold gate which ignores signal noise and

[19]Stefano Papetti, Marco Civolani, and Federico Fontana, "Rhythm'n'Shoes: a Wearable Foot Tapping Interface with Audio-Tactile Feedback.," in *NIME* (2011), 473–476.

small transients of audio caused by rebounding vibrations. Each sensor's data is unpacked so that there are four individual streams of audio signal for each sensor.

The authors included their own sound synthesis engine to accompany the *Rhythm'n'Shoes* which uses the Synthesis ToolKit which is a library for MaxMSP and Pd to generate percussive impact sounds. This synthesis engine simulates a resonator being struck by a mass and then simulating the resulting vibrations from the resonator. The parameters of the synthesis engine are: the mass in kilograms, the resonator's frequencies, and a non-linear spring. These parameters allow the user to shape the instrument's timbral characteristics into a variety of materials like glass, wood, metal, and plastic.

2.2.4. ***Resuscitation***

Resuscitation is a composition for laptop ensemble in which four players play a shared physically-modeled virtual instrument.[20] The model concept is four strings connected to the four corners of a resonant plate which themselves are connected at one point above the plate. Each player excites one of the strings by striking and plucking a piezo-based interface. Figure 2.9 below shows the mental model used to create the virtual instrument created in *Resuscitation.*

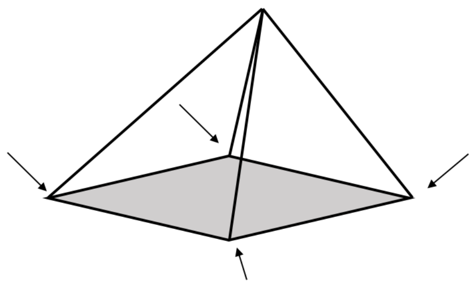


Figure 2.9. A representation of the mental model used to conceptualize the virtual instrument that is used in *Resuscitation.*

[20]Eric Sheffield, "Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments," (2019) *LSU Doctoral Dissertations.* 4918. https://digitalcommons.lsu.edu/gradschool$_{d}issertations$/4918,

Using a local wired network, four MaxMSP patches communicate user inputs to the model where each player's patch is a portion the the entire model. Each patch contains separate audio outputs for the strings and plate model to give each user control over the balance of their dynamics just as an acoustic instrumental ensemble would balance their individual dynamics.

Resuscitation was composed much in the same way as a percussion quartet is composed, by focusing on rhythms passed among the ensemble and built upon while also exploring timbrel shifts among each player. The notation of *Resuscitation* is very close to traditional music notation because each performer's role is similar to that of a traditional percussion quartet performer. Specific actions such as tapping, striking, damping of the plate and plucking of the strings are notated with specific note heads and on different note spaces much like a multi-percussion piece is notated.

2.3. Mapping In Music

2.3.1. Digital Audio Effects Parameter Mapping

Verfaille et al. suggest a mapping strategy that connects gestures to audio effect parameters.[21] This is a novel approach because the authors argue that audio effects are not usually thought of as something to be performed or controlled in real-time like audio synthesis typically is. Zolzer shows that audio effects can be controlled in the context of interactive systems for performers, composers, and sound engineers.[22] The authors focus on two control categories for parameters:

- gestural
- adaptive

[21]Vincent Verfaille, Marcelo M Wanderley, and Philippe Depalle, "Mapping strategies for gestural and adaptive control of digital audio effects," *Journal of New Music Research* 35, no. 1 (2006): 71–93.

[22]Udo Zölzer, *DAFX: Digital Audio Effects* (John Wiley & Sons, 2011).

An example of gestural control is the direct control of an audio effect with a control surface such as but not limited to sensors, knobs, potentiometers, and sliders. These control surfaces output numerical values that control parameters in the audio effect. The goal is to extract the intention of the gesture and convert it into a meaningful control for the audio effect.

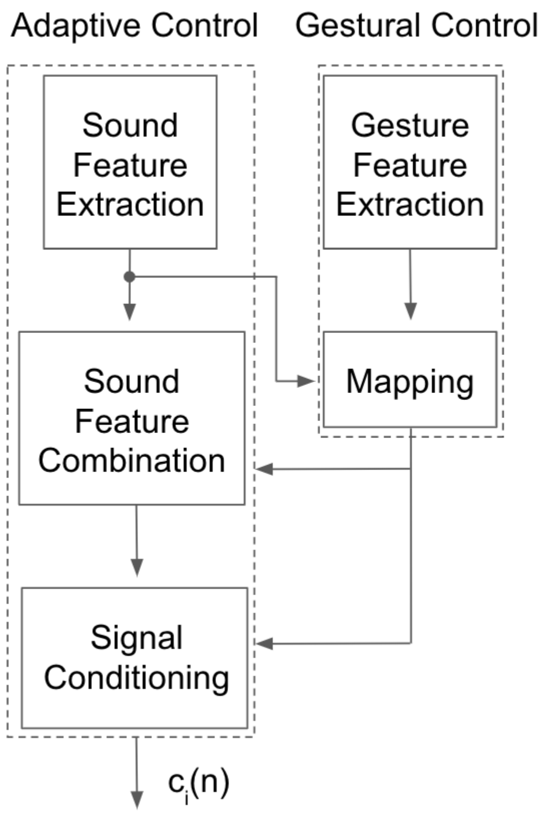


Figure 2.10. Diagram of the two-layer mapping technique connecting a sound feature to an effect control.

Adaptive controls consist of using sound features as control parameters such as pitch correction, compression, and score following. The authors suggest a two-layer mapping strategy to extract sound features and convert them into audio effect controls shown above in Figure 2.10. The authors present six examples of audio effect models which use this two-layer mapping strategy to control audio effects:

- adaptive robotisation
- adaptive granular delay

- adaptive spatialization
- prosody change
- adaptive equalizer
- adaptive spectral tremolo

Adaptive Robotisation

This is a model that allows a user to control the pitch of a robotic voice with their own voice. The control value that must be defined by the user is the fundamental frequency for the robotic voice output and the feature extraction is the RMS value of the user's voice.[23]

Adaptive Granular Delay

In granular delay, the sound input is decomposed into grains and sent to delay lines with varying lengths and amplitudes. In real-time use with granular delay lines, the user is limited to a finite amount of delay lines which also means a finite number of control parameter which is the gain of each delay line. The feature, the input grains, are extracted and normalized to the range [0-1] and quantized to produce a limited number of control parameters which are mapped to the gain of each delay line.[24]

Adaptive Spatialization

In this model a dancer controlled parameters of an eight-speaker sound system which controlled the spatialization of sound with controls such as distance from the listener, elevation, azimuth, and distance. The sound being spatialized was created with a combination of granular synthesis, delay lines, and filters. The dancer's position was captured by sensors and used as the feature to extract and create parameters that controlled the sound position, speed of the sound's trajectory, trajectory shape, or a combination of the three. When the

[23]Verfaille, Wanderley, and Depalle, "Mapping strategies for gestural and adaptive control of digital audio effects."

[24]Ibid.

dancer moved around in space their position was measured by the sensors and used as a trajectory to send the sound synthesis around the eight speakers which mirrored the motion of the dancer.[25]

Prosody Change

The authors describe prosody as the pitch, loudness, and time duration of spoken voice. One can modify the prosody of a voice by modifying these three components. Verfaille connects a mapping layer between these described sound features and a time-varying transformation. The gestural control is used on pitch-shifting, time-scaling (modifying the time duration), and gain alteration. The adaptive control is provided on the pitch-shifting ratio and the time-scaling ratio. In controlling intonation changes and adaptive time-scaling prosody change is facilitated. Further advancement on this work is needed to translate a user-defined emotion to control the system and emulate the given emotion.[26]

Adaptive Equalizer

This equalizer has a time-varying equalization curve based on a vector as the feature. Equalizers are usually used as audio plugins in DAWs to control the timbre of a sound source by boosting or attenuating the gain of certain frequencies. Gestural control is used on the mean, minimum, and maximum values of the filter curve, interpolation of the time scale by using a joystick, and the interpolation type such as complete with no attack present or incomplete with an attack present. All of this is achieved with a slider. This adaptive equalizer was used as an instrument called the *Noisonic* in the piece *Flute Salad* by Verfaille.[27] This adaptive

[25]Verfaille, Wanderley, and Depalle, "Mapping strategies for gestural and adaptive control of digital audio effects."

[26]Ibid.

[27]Vincent Verfaille, "Effets audionumériques adaptatifs: théorie, mise en œuvre et usage en création musicale numérique." (PhD diss., Université de la Méditerranée-Aix-Marseille II, 2003).

equalizer allows the user to modulate timbre, filter, and shape spectral envelopes and time amplitude envelopes with the time interpolation control.[28]

Adaptive Spectral Tremolo

The spectral tremolo is achieved by applying tremolo on each frequency of the STFT (short-time Fourier transform) which is a Fourier transform used to find the sinusoidal frequency and the phase of a signal as it changes over time. Gestures control the bounds of the tremolo rates while sound features control the depths and rates of the spectral tremoli adaptively. With this mapping the user is able to control tremolo speed and the phasing aspect globally.[29]

Wekinator

Wekinator is a software solution that provides a system to apply machine learning techniques to musical performances in real-time.[30] The author's goal was to create a tool that intuitively supports end-user real-time interaction with the fundamental concepts of machine learning techniques such as creation, refinement, and evaluation of the results. Wekinator accomplishes this in real-time, which makes it the first live, performance-ready machine learning system for musical applications. Some of the key features of Wekinator are:

- a centralized GUI where the user can create training data sets, modify training data sets, select sound features, configure learning algorithms, and run in performance from a single application
- evaluation of qualitative performance data in real-time
- native feature extraction for basic audio parameters and OSC compatibility

[28]Verfaille, Wanderley, and Depalle, "Mapping strategies for gestural and adaptive control of digital audio effects."

[29]Ibid.

[30]Rebecca Fiebrink and Perry R Cook, "The Wekinator: a system for real-time, interactive machine learning in music," in *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)* (2010).

- patching of the results from analysis to control synthesis, visuals, or any OSC compatible processes

2.4. Summary

The concept of mapping requires close attention whether custom virtual instruments are being built or parameters for standard audio effects in an audio signal path are being implemented.

Garnett and Goudeseume outline the three basic types of mapping for synthesis models as one to one, many to one, and one to many.

Verfaille et al. shows how mapping can be used to perform musical gestures on audio effects that are not usually included in traditional performance practice.

Chaos can be used to create oscillators, control parameters, spatialize audio, and be used with digital waveguides.

Yadegari's research shows how continuous-time chaotic equations can be implemented to achieve a chaotic oscillation that can be stable or unstable with specific initial conditions.

Cupellini et al. shows that chaotic maps can be applied to the mapping of universal synthesis parameters such as LFO's.

The framework suggested by Soria et al. shows that this method could be applied to any chaotic system that desires exploration. The intent of this research was to provide electroacoustic composers with a new tool for exploring spatialization among large speaker arrays and allows composers to dynamically control parameters that shift between stable and chaotic sound spatialization.

Berdahl et al. demonstrated that interesting musical instruments can be created by connecting chaotic systems to digital waveguides and using music controllers to change parameters of the chaotic systems. The use of digital waveguides with chaotic systems made it easier to synthesize harmonic tones. The authors also show that the edge of chaos can be explored by gradually adjusting parameters.

MOSAIC brought an object-oriented programming approach to physical modeling synthesis and allowed the user to take a modular approach to synthesis with a simple c++ framework.

The *ETabla* made use of the performer's actions more than just as trigger-based events. The force of strikes were used to augment pitched parameters to simulate pitch bends from deforming the drum head and accounted for the various positions a performer could strike the drum head with their fingers to simulate the diverse characteristics of a drum head. The interface created for the current project attempts to capture the expressive nature of percussion-inspired instruments and allow for a more immersive instrument experience.

The *Rhythm'n'Shoes* claims to give the user more expressive control than simpler instruments that are trigger-based. The interface created for the current project uses a simple, trigger-based input system but is designed in a way that allows expressive control over the sound. The *Rhythm'n'Shoes* shows that a multi-channel, trigger-based instrument can yield desirable sonic characteristics when used in conjunction with synthesis models.

Wekinator is a machine learning solution for real-time musical performance where users can set up training data to use for harmony analysis, beat tracking, and many other processes. The data can be evaluated by machine learning algorithms to create new sets of data to apply to the mentioned audio processes.

CHAPTER.3
AN OVERVIEW OF THE HEXAPAD CONTROLLER

The Hexapad controller (Hexapad for short) is an electronic instrument interface that has a built-in microphone array of ten piezo-electric contact microphones each on a discrete audio channel separated from one another. The main focus of the Hexapad is to function as an instrument that generates signals at an audio sampling rate for spatial audio applications. Striking the Hexapad causes vibrations to propagate throughout the body which are received by the contact microphones. Figure 3.11 below shows the Hexapad.

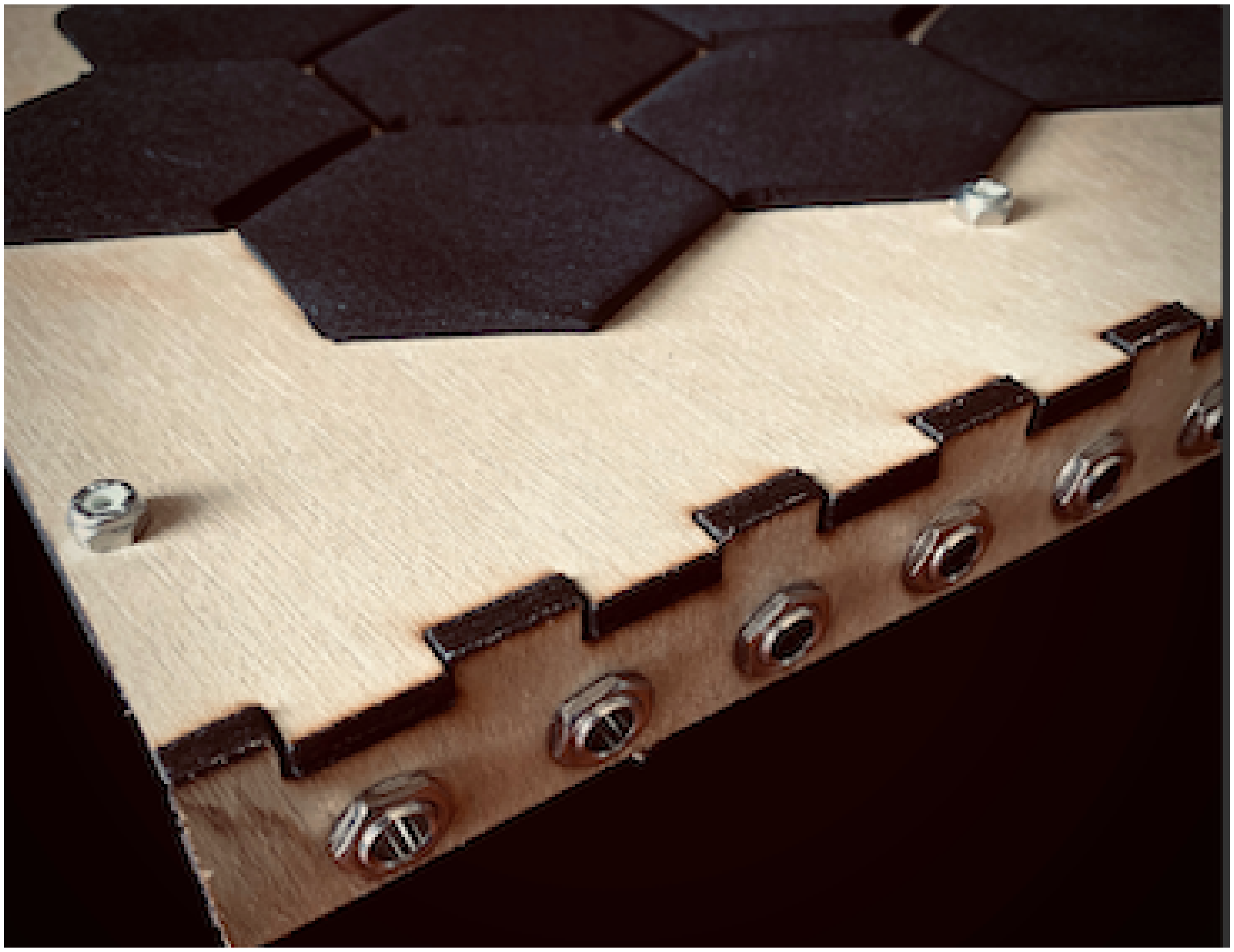


Figure 3.11. The Hexapad audio signal-rate controller.

3.1. Construction

The Hexapad is constructed from laser cut sections of birch wood, neoprene padding, contact microphones, resistors, and audio output jacks. All of these parts are inexpensive, making it easy for any musician to build their own version of the Hexapad. Each microphone is connected to an audio output jack with a resistor connected in parallel with the contact microphone to lower the volume output of each microphone. Since these microphones are inexpensive, they do not all have the same exact volume output so the resistor values vary between each microphone to achieve an even output across the entire array. To avoid direct contact with each microphone, a pad of neoprene sits above each one which prevents clipping the output of each microphone. This smooths out the otherwise harsh, percussive sound output. The neoprene pads also aid in isolating each microphone from one another effectively reducing unwanted vibrations from neighboring ones. This does not cancel out all unwanted vibrations completely, but it reduces them significantly. This is desirable in the Hexapad's application in this project because each pad and microphone function as discrete pitches or discrete audio input channels. Figure 3.12 shows the schematic diagram of each channel on the Hexapad.

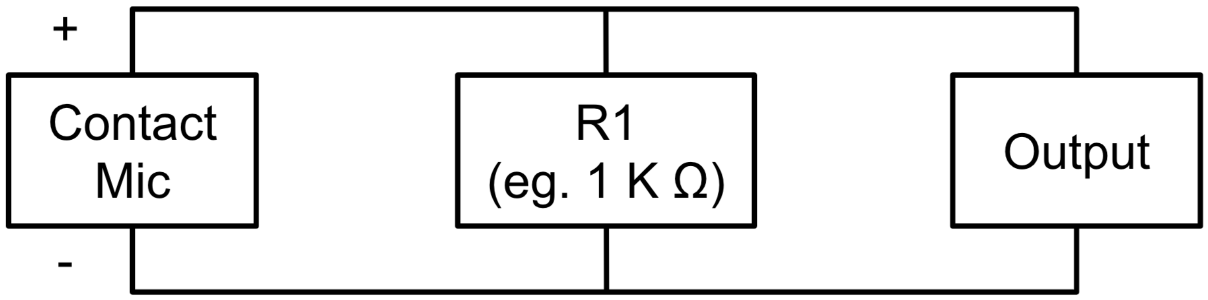


Figure 3.12. Schematic diagram for one channel of the Hexapad.

3.2. Interaction Methods

3.2.1. Fingertips

Using one's fingertips on the pads is a percussive way of interacting with the Hexapad similar to the percussive interaction discussed by Tindale et al.[1] Tindale et al. state that a percussion interface must be able to capture strike events with the resolution to accurately extract parameters such as velocity, angle, and position. By connecting each microphone channel on the Hexapad to a speaker in the same configuration as the pads laid out on the Hexapad, the user's strike position is inherently spatialized because the microphone that is struck is sounded the loudest. Then, the surrounding microphones are triggered by the propagating waves and the respective speakers sound if the vibrations are strong enough to trigger the piezo microphones. The velocity and angle of strike is inherently extracted and present in the audio signal when using contact microphones in this configuration in the Hexapad.

3.2.2. Brush

Using a brush is the primary way that the Hexapad is interacted with in this project. Dragging the brush around the pads creates an image in the speakers that mirrors the brush's direction of motion when each microphone channel on the Hexapad is connected to a speaker in the same configuration as the pads. The audio created is an interesting grinding sound similar to scraping objects along a rough surface such as dragging a heavy object on the ground. The sound of the brush on the Hexapad can also be described as granular since each bristle of the brush individually excites the microphone on a micro level and the aggregate sound of all the bristles combined create the overall texture.

[1]Adam R Tindale et al., "A comparison of sensor strategies for capturing percussive gestures," in *Proceedings of the 2005 conference on New Interfaces for Musical Expression* (National University of Singapore, 2005), 200–203.

3.3. Instrument Type

When presenting a new electronic instrument controller it is important to define how the device fits into the current paradigm of electronic instruments. Miranda et al. discuss how electronic instrument controllers fit into two categories: instrument-like and instrument-inspired.[2]

Instrument-like controllers are modeled after a specific acoustic instrument and their goal is to model that instrument as exact as possible. The authors point out that with modern sensors and electronic components the possibilities of electronic controllers are virtually limitless, so why do designers model existing instruments? It's because the majority of musicians play standard acoustic instruments and if the controller is built like these standard instruments, then the musician does not have to learn new techniques to play the electronic version. The most prolific example of instrument-like controllers is keyboard controllers such as MIDI controllers and keyboards.

Instrument-inspired controllers do not seek to exactly replicate the control surface of existing acoustic instruments, but seek to provide a control surface that allows the player to produce gestures similar to existing instruments. The player can then interact with the controller in a familiar way even though the control surface is not laid out exactly as an existing acoustic instrument. Unlike as with instrument-like controllers, players have the opportunity to extend their techniques past the limitations of their respective acoustic instrument and develop new ways of performing that their acoustic instrument did not allow them to before.

With these concepts in mind, the Hexapad can be considered an instrument-inspired controller. It is inspired by percussion instruments, but the Hexapad can be interacted with in a variety of ways and is not limited to specific objects or gestures to excite the microphones.

[2]Eduardo Reck Miranda and Marcelo M Wanderley, *New digital musical instruments: Control and interaction beyond the keyboard*, vol. 21. Middleton, Wisconsin: (AR Editions, Inc., 2006).

3.4. Extracting Parameters

Once audio is captured by the Hexapad's microphones, it can be sent to a number of different sources, but sending it to MaxMSP allows the user to process the data in many ways.

The simplest processing method is to send the audio directly to the digital-to-analog converter (dac) so that it can be sent to loudspeakers from the computer. Audio effects can be placed in series with the input (Hexapad audio) and output (dac) which is similar to guitarists placing audio effect pedals between their guitar and amp. There are many standard audio effects that can be used in this configuration such as time-based effects like delay, reverb, phasing, or effects like distortion, equalization, compression, etc. Figure 3.13 below shows this configuration for a single channel.

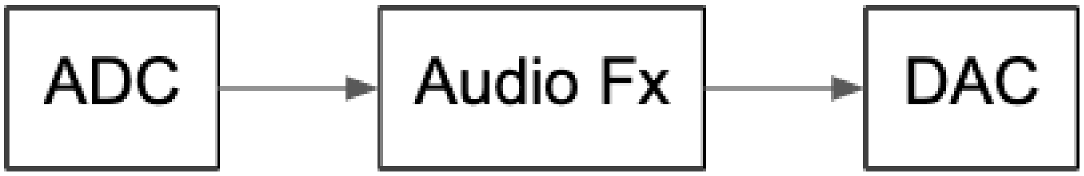


Figure 3.13. The basic audio DSP signal flow for each channel for the Hexapad.

Another method is to convert the audio signal into a MIDI message when it crosses a certain threshold to use as a trigger for operations that use control messages such as sample playback, envelope triggering, or toggling effects on or off. Figure 3.13 shows the Hexapad signal converted into a trigger for control messages. The signal is passed into a `past` object which sends a signal of 1 when the input signal passes above the threshold of the object which is 0.1.

A combination of these two methods is the primary method the Hexapad is used in this project for generating audio for the composition. The Hexapad's direct audio with no processing is used as a musical sound, which passes down the patch line on the left side of Figure 3.14 straight into the `gen` object. The Hexapad's audio is converted into triggers for certain parts of the physical model to excite various parameters and the direct sound of the Hexapad is processed with DSP algorithms similar to Figure 3.13 to achieve interesting musical sounds.

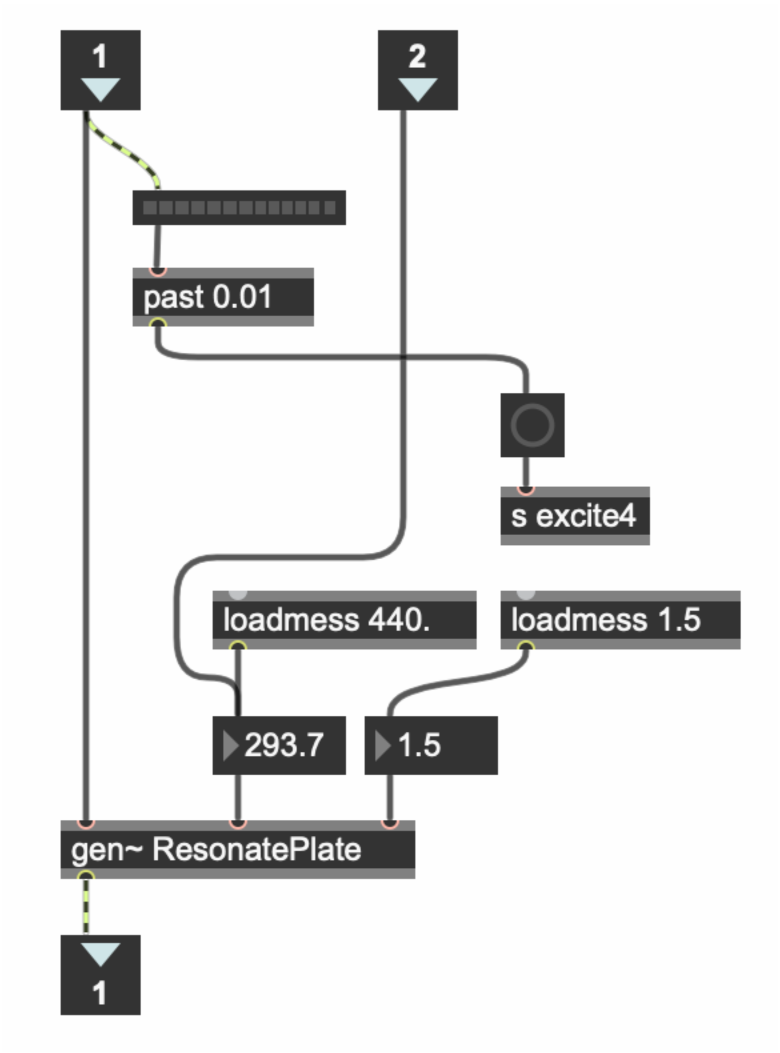


Figure 3.14. Converting the Hexapad signal into a trigger for control messages.

CHAPTER.4
CHAOTIC FACTA

Chaotic Facta is a two-movement composition for Hexapad, guitar, and fixed media. Movement one is titled *VIII Tractuum* and movement two is titled *Ferrum*. It is the culmination of the research done in implementing audio synthesis engines that utilize chaotic equations reliably and in a repeatable modular way that other composers can implement in their own work. *Chaotic Facta* is Latin for 'Chaotic Behavior' which describes the shared theme between the two movements which is chaotic synthesis.

VIII Tractuum is Latin for *Eight Pathways* which represents the fact that the Hexapad is used in an eight-channel arrangement for the composition. In concert, *VIII Tractuum* is connected to eight different loudspeakers. As the performer moves the brush around the pads, rhythms are created in time and in space in the same arrangement as the pads. *VIII Tractuum* focuses on the Hexapad as a solo instrument controlling the conceptual instrument model discussed in the next section. The Hexapad is explored and focuses on how much musicality the Hexapad can create on its own as a solo instrument much in the way solo acoustic instrumental pieces are realized. The wide range of musical sounds and applications of the Hexapad are presented and shows that the Hexapad is a novel instrument in its own right.

Ferrum is for guitar and fixed media and is Latin for *iron*. This movement explores the possibilities of using chaotic sounds and synthesis in a style of music that is traditionally tonal, which is progressive metal. It is inspired by such artists as *Jason Richardson*, *Animals as Leaders*, and *Mike Gianelli* where tonality and voice-leading practices are more traditional than in the current paradigm of experimental and computer music.

4.1. Movement One: *VIII Tractuum*

4.2. Conceptual Model

To have a starting place to incorporate chaotic equations into a synthesis engine, a conceptual instrument model was realized. Taking inspiration from percussion instruments such as steel drums and hang drums, the following instrument was conceived.

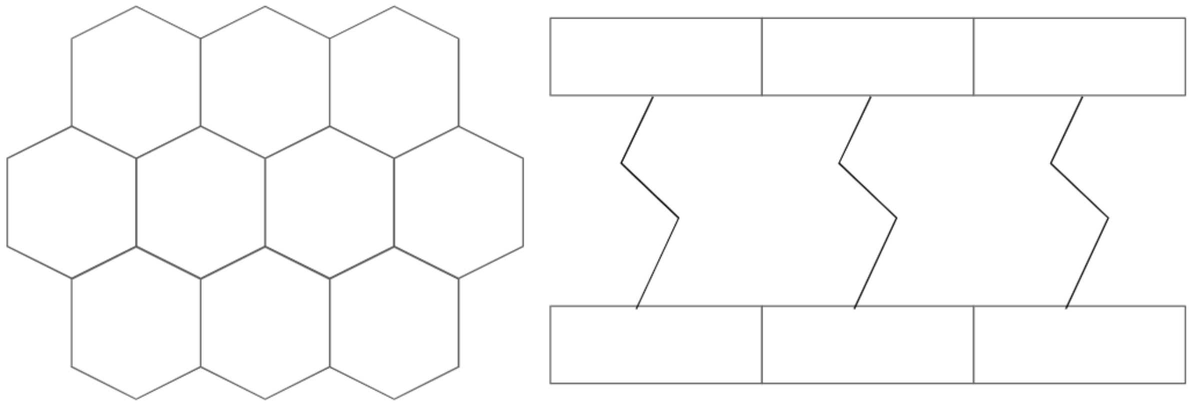


Figure 4.15. The Hexapad conceptual instrument.

The left side of Figure 4.15 shows the surface of an array of harmonic, resonant plates arranged in the same way as the Hexapad tuned to a traditional scale like a steel drum or hang drum. Each resonant plate corresponds to the pad that is in the same position on the Hexapad. Striking the Hexapad excites the resonant plate. The right side of Figure 4.14 shows the side view of the instrument. The top portion is the same surface array of resonant plates that are linked to another set of resonant plates by a chaotic digital waveguide. Exciting the top plates triggers the chaotic strings which in turn trigger the bottom plates that are tuned to overtones of their respective top plate. Connecting digital waveguides to a resonant plate was explored recently by Sheffield in which four players each controlled a single string all attached to one resonant plate.[1].

4.3. Software Realization

Implementing the synthesis engine for the previously described virtual instrument began by Implementing each individual part as a synthesis module. These modules are the top plate, the chaotic string, and the bottom plate. Figure 4.16 below shows the flow diagram that describes the synthesis engine for the Hexapad virtual instrument.

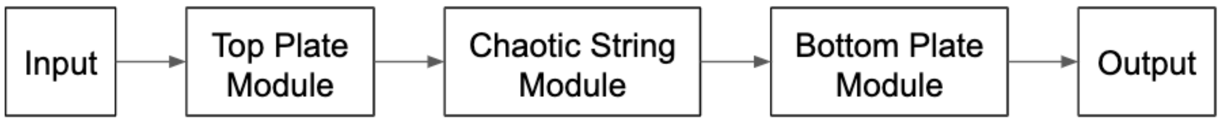


Figure 4.16. Basic flow of the synthesis engine for the Hexapad virtual instrument.

In this diagram audio signal from the Hexapad input (the microphones) flows through each module and the resulting audio from each process is passed to the next module much like in a DAW (digital audio workstation) where multiple plugin effects are loaded onto an audio channel and each process is passed on to the next one in line.

4.4. Top Plate

To realize the top plate module, a resonant filter was implemented to model the resonant nature of the conceptual metal plate. A resonant filter is described by Klatt with the following equations:[2]

$$y(nT) = Ax(nT) + By(nT - T) + Cy(nT - 2T) \tag{4.20}$$

[1]Sheffield, Eric, ”Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments” (2019) *LSU Doctoral Dissertations.* 4918. https://digitalcommons.lsu.edu/gradschool$_{d}issertations$/4918

[2]Dennis H Klatt, “Software for a cascade/parallel formant synthesizer,” *the Journal of the Acoustical Society of America* 67, no. 3 (1980): 971–995.

$$C = -exp(-2 * \pi * bw * T) \tag{4.21}$$

$$B = 2exp(-\pi * bw * t)cos((2 * \pi) * F * T) \tag{4.22}$$

$$A = 1 - B - C \tag{4.23}$$

This was implemented in MaxMSP's `gen` environment which processes at the sample level and contains a `codebox` object that allows the user to write C-style code. Samples from the output of the resonator y(nT) are calculated from the audio input going into the resonator x(nT) by equation 6.5. y(nT - T) and y(nT - 2T) are the last two samples of audio from the output y(nT). A, B, and C are the constants of the difference equation which are associated with a resonator's frequency and bandwidth. Listing 4.1 below shows the implementation of the resonant filter in the `gen` environment's `codebox` object.

Listing 4.1. Gen implementation of resonant filter.

```
History x1, x2, y1, y2;
x = in1;
f = in2;
bw = in3;
t = 1 / samplerate;
    c = −exp(−2 * pi * bw * t);
    b = exp(−2pi * bw * t) * cos((2 * pi) * f * t);
    a = 1 − b − c;
    y = x * a + b * y1 + c * y2;
        y2 = y1;
        y1 = y;
        x2 = x1;
        x1 = x;
out1 = y;
```

The *History* keyword creates a variable that holds a single sample of audio acting as a single sample of memory. It does not matter what value is used to initialize it since it will immediately get overwritten when audio begins processing so it was initialized with zero until it was used.

In the next section, in1 refers to the first input of the `codebox` object which in this case is where the audio is being fed that has been named x. In2 is the fundamental frequency at which the resonator resonates at that has been named f and in3 is the bandwidth of the resonator. Listing 4.1 describes T as 1 divided by the sampling rate and in the `codebox` object the samplerate keyword automatically checks for the projects sample rate setting and passes it without needing to pass the integer manually. In this case 44,100 Hz is the sampling rate. Equations 4.21-4.24 are then implemented directly using the previously named variables. The next equation in Listing 4.1 is the implementation of equation 4.21. Remembering that y(nT) is the output sample and x(nT) is the input sample, this equation can be understood more intuitively as:

$$output = input * A + B * previousOutputSampleOne + C * previousOutputSampleTwo \tag{4.24}$$

Each *History* variable is updated with the last sample calculated and finally the computed audio (y) is passed to out1 of `codebox` which is the output. When this process is complete the `gen` object becomes a modular object than can be duplicated and inserted into any part of the synthesis engine. Figure 4.17 shows the object below where audio can be sent into the first inlet, a number representing frequency can be sent into the middle inlet, and a number representing the bandwidth can be sent into the last inlet. The resulting audio passed out of the outlet on the bottom left of the object.

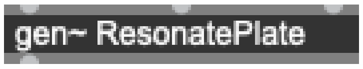


Figure 4.17. Resonant plate gen object.

4.5. Chaotic String

The string module is a digital waveguide that is triggered by the excitation of the top plate.

The module is shown below in Figure 4.18.

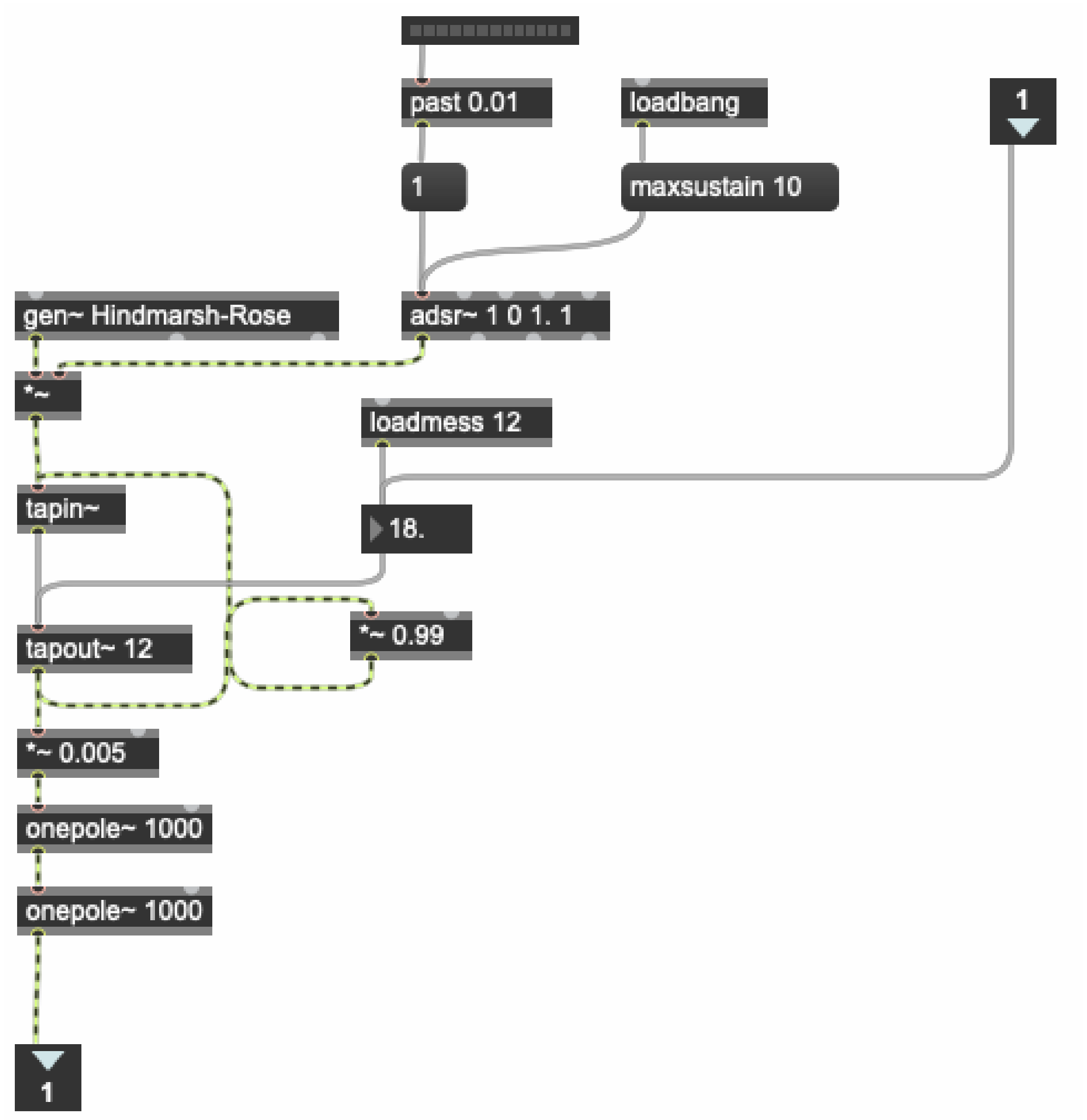


Figure 4.18. Chaotic string module.

At the top of Figure 4.18 above the `past` object, there is an audio meter that measures the peak level of the incoming audio signal. It outputs the value of the audio in the form of floating point numbers and the `past` object sends out a notification when the audio signal is above a defined threshold, in this case it‘s 0.01. This is the trigger that excites the digital waveguide by triggering an `envelope` object. In this digital waveguide, the commonly used `noise` object that was previously discussed in Chapter 3 has been replaced with a chaotic oscillator as the noise source for the digital waveguide. This is how chaos is inserted into a synthesis engine in a stable way that is resistant to exploding. Because it’s being used as a short burst of audio with the `envelope` object and the oscillator‘s parameters are not being changed, it does not go unstable or explode. This chaotic oscillator implements a variation of the Hindmarsh-Rose chaotic equations previously discussed in Chapter 2, which is referred to as the Hindmarsh-Rose variation chaotic oscillator. The Hindmarsh-Rose model is described by the following differential equations:[3]

$$\dot{x} = y - ax^3 + bx^2 + I - z \tag{4.25}$$

$$\dot{y} = c - dx^2 - y \tag{4.26}$$

$$\dot{z} = r(s(x - x_1) - z) \tag{4.27}$$

Just like the top plate algorithms in the previous section, the Hindmarsh-Rose variation chaotic oscillator was implemented in the `gen` environment in MaxMSP. Listing 4.2 on the following page shows the code for the `codebox` object.

[3]Hindmarsh and Rose, “A model of neuronal bursting using three coupled first order differential equations.”

Listing 4.2. Gen implementation of the Hindmarsh-Rose variation chaotic oscillator.

```
History x(.9);
History y(.4);
History z(.1);

Param a(1.);
Param b(3.);
Param c(3.);
Param dt(.17);
Param d(5.);
Param r(.001);
Param s(1.);
Param i(11.);
Param newValue(1);

//Calculate change
dx = y - a * (pow(x, 3)) + b * (pow(x, 2)) + i - z;
dy = c - d * (pow(x, 2)) - y;
dz = r * (s * (x - dx) - z);

x = newValue ? x + dx * dt : x;
y = newValue ? y + dy * dt : y;
z = newValue ? z + dz * dt : z;

out1 = x;
out2 = y;
out3 = z;
```

The Hindmarsh-Rose equations have three values to compute, x, y, and z. A *History* variable was made for x, y, and z so that a single sample of memory was available for the passing of previous audio samples.

All of the *Param* variables refer to the coefficients in the Hindmarsh-Rose equations and were experimented with until an interesting and non-exploding oscillation was discovered. The *newValue* variable allows the object to self oscillate. Otherwise, audio would be required to be sent into it to trigger oscillation.

The next section calculates the change over time for x, y, and z in relations to the coefficients supplied by the *Params*.

The next section updates the *History* values. Taking x as an example, this line increments x by dx * dt if x = 1. This happens for x, y, and z.

Lastly, the samples are passed to the outputs. The Hindmarsh-Rose equations have three output values so the module has three audio outputs also. It was found that the sound of these three outputs are not easily distinguishable from each other so in practice the output of x is the only one that is being utilized. Coefficients r and s are not listed because they are only related to z in the Hindmarsh-Rose equations and the output of x is the only audio being used. The z equation did not output any audio, so it was left out of the x equation. This is why it is referred to as the Hindmarsh-Rose variation chaotic oscillator.

The result of using this chaotic oscillator as the noise source for the digital waveguide is an interesting string sound sound that is ambient in nature and rich in high-end frequencies and harmonic saturation. It is a much more novel string sound than the standard digital waveguide sound. The ambient nature of this chaotic string sound was a serendipitous discovery because it complements the ambient resonance of the top plate sound so well. The conceptual model is inspired by percussion instruments such as steel drums and hang drums which themselves are ambient sounding instruments with natural reverb in the body of the instruments and the chaotic string sounds like it could be a natural part of one of these instruments. Figure 4.17 shows the `gen` object for the Hindmarsh-Rose variation chaotic oscillator which has one inlet that can receive audio signals or *Param* values to change the coefficients in the equations. It has three outlets for output x, output y, and output z.

Exploring the Edge of Chaos

As discussed earlier when working with chaotic systems, the edge between stable and unstable states can be minuscule, so exploring this space can prove challenging. However, it is a fascinating space to explore musically because of the diverse textures, timbres, and overall variety of tones around the edge of chaos. Much experimentation was carried out with the Hindmarsh-Rose oscillator to find the edge of chaos. The Hindmarsh-Rose equations contain many more coefficients than any of the other equations discussed, so finding general boundaries to work within was necessary. The initial conditions of the equations are difficult to explore, but the following is a general set of coefficient values: $dt > .05$ $dt < .18$, $i > -1$ $i < 25$, $a = 1$, $b = 3$, $c > -9$ $c < 17$, $d > 4$ $d < 6$. Since the Hindmarsh-Rose variation chaotic oscillator is being used as the noise generator for the chaotic string, a setting right on the edge of chaos was desirable. The coefficients above achieved this sound, which is right on the edge of chaos. The sonic character is saturated with noise, but the sustained sound has infrequent moments of silence. The result is a randomly rhythmic crackle that is unpredictable but does not explode.

Any values outside of the ranges listed above were found to break the oscillator. When this happens the system inside of MaxMSP cannot resume by simply changing the coefficients to a new value inside of the ranges above. The oscillator has to be re-instantiated. The simplest way to due this is by deleting the object and then running the undo command. Working with these equations and coefficients is difficult. The range listed above is actually dynamic in that changing the value of one changes the stable range of another so working with four variables that all affect the stability of the other is difficult to navigate. Trial and error was used to find values that resulted in interesting sounds that did not break the oscillator and was used for the chaotic string module.

4.6. Bottom Plate

The bottom plate module is almost identical to the top plate module. The same `gen` object is used but the fundamental frequency of the plate is set to an octave above its respective top plate tuning plus or minus four Hz to add a more realistic tonal response. The band width also has a higher setting than the top plate so that it resonates for longer. The audio input for the bottom plate is the output of the chaotic string as opposed to the Hexapad's input. This gives the bottom plate a more ambient and ethereal quality which completes the mental model discussed above in Figure 4.15.

4.6.1. Chaotic Drone

VIII Tractuum uses one more chaotic sound engine which utilizes the circle map discussed in Chapter 1. The circle map is defined by the following equation:

$$\theta_{n+1} = (\theta_n + \Omega - (\kappa/2\pi)sin(2\pi\theta_n))\%1.0 \tag{4.28}$$

The circle map is used as a chaotic drone sound. Audio is fed into it and the chaotic map affects it according to the equation's initial conditions. Figure 4.19 on the following page shows the circle map implemented using standard MaxMSP objects. The sound source used is the Hindmarsh-Rose variation chaotic oscillator discussed previously. The sound source is first fed into a delay line which is the `tapin` and `tapout` objects. The argument in the `tapout` object sets the delay time in milliseconds. This delay line is used so that the output of the circle map can be fed back into itself. The audio is then routed through a network that outlines equation 4.29 above. The two coefficients omega and kappa are controlled using the sliders labeled in Figure 4.18. The sliders have been mapped using the one-to-one method described by Hunt et al.[4] The sliders standard range is 0-127, but a multiplier of .01 has

[4]Hunt and Wanderley, "MHunt, Andy, and Marcelo M. Wanderley. "Mapping performer parameters to synthesis engines." Organised sound 7, no. 2 (2002): 97-108."

been used to map the parameters to a range of 0-1.27 which was found to be an interesting parameter space for the circle map. By applying a low-pass filter to the output of the circle map an interesting chaotic drone is created and serves as a low-end bass instrument for the composition. The cutoff frequency of the low-pass filter becomes a performance parameter and is used extensively throughout the composition.

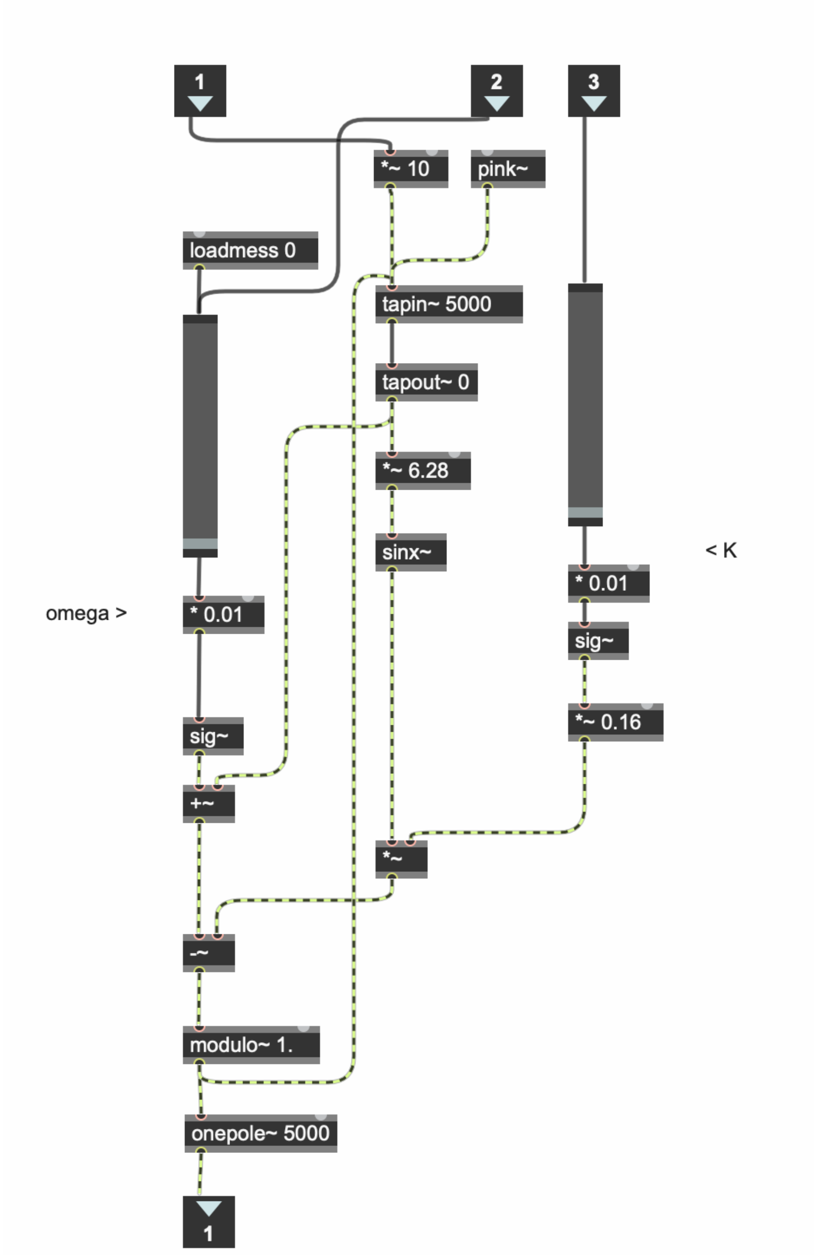


Figure 4.19. MaxMSP implementation of the Circle Map.

4.7. Notation

The notation for *VIII Tractuum* uses many traditional music notation symbols such as note-heads, dynamics, and staves, but a few symbols and gestures have been created to suit the nature of the piece. Figure 4.20 below shows the legend for the symbols in the composition's notation scheme.

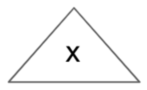
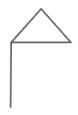
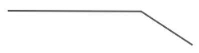
	Pitch Preset Number
	Dejong Pad Strike
	Low-Pass Filter Cutoff Frequency

Figure 4.20. Legend of symbols for *VIII Tractuum's* notation.

The pitch presets are the delay times in milliseconds used to set the pitch of all the Hexapad's outputs when they are being used to trigger digital waveguides as opposed to the pitches and triggers for the *ambi drum*. These delay times are represented by an integer inside of triangular symbols used throughout the score of *VIII Tractuum*. Solid note heads represent the striking of the pads that correspond to the written pitch in the staff. The first pad is note A2 and the last pad is note A3. The slanted line represents the cutoff frequency for the circle map drone which is also indicated by integers in Hertz. Crescendo and decrescendo markings are used in conjunction with these cutoff symbols to indicate movement from one cutoff frequency to another. Score excerpts from *VIII Tractuum* are presented in Figures 4.21 and 4.22 on the following page. These excerpts show an example of each of the notation symbols listed above in Figure 4.20.

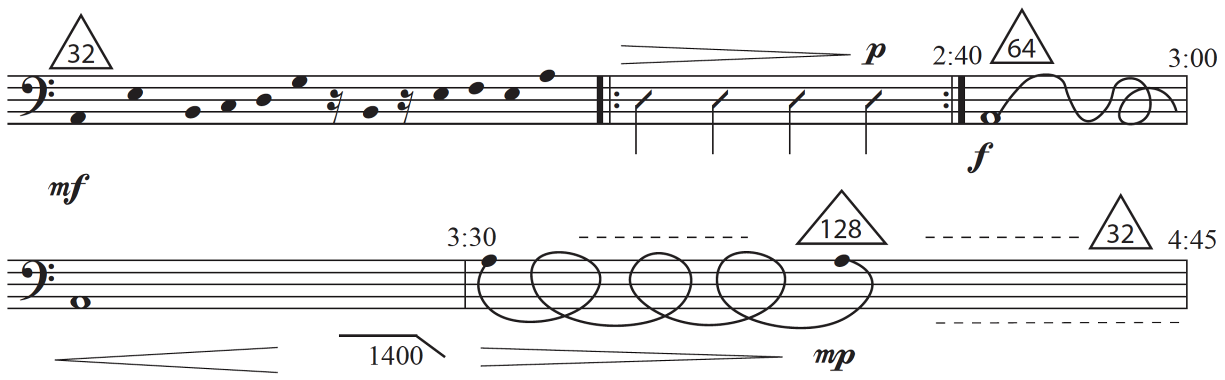


Figure 4.21. Score excerpt one of *VIII Tractuum*.

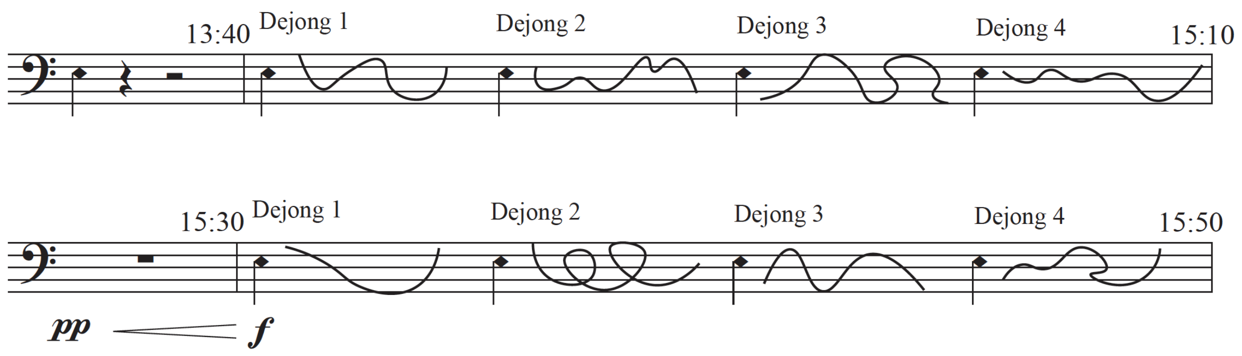


Figure 4.22. Score excerpt two of *VIII Tractuum*.

Time codes are used in place of measure numbers so that the improvisational intent of the piece is easily facilitated allowing the performer more rhythmic freedom. The use of slash notation also facilitates improvisational freedom by indicating repetition of motives. For example, the measure at time code 2:10 containing the stemless note heads indicate pitched improvisation which is then continued indicated by the slash notation. The suggestion is to improve until time code 2:40 comes up. This is the case for all time codes throughout the movement.

Looking at the beginning of the score the curved lines indicate brushing gestures on the Hexapad. This can be interpreted in any way, but the suggested idea is to follow the contour indicated by the trajectory of the lines moving throughout the staff. The pads are tuned to A natural minor for the *ambi drum* sections so the entire piece is notated as if it is in A natural minor for ease. This is why the lowest pitch for the brushed section begins on A2.

The chaotic drone is notated on the A2 space as whole note values. The performer controls the cutoff frequency of a low-pass filter processing the chaotic drone as a musical parameter. The target cutoff frequency for important sections is listed below these notes. Crescendo and decrescendo gestures indicate the change in cutoff frequency, but the performer is free to control the cutoff frequency as they see fit.

Lastly, the pitch presets indicated by the triangle markings are linked to a digital waveguide that processes the input of the Hexapad giving the natural audio of the Hexapad a pitched character and acts as a transitional sound between the raw Hexapad audio from brushing gestures to the *ambi drum* setting.

The result is an ethereal composition containing a wide array of rich timbres, ambient drones and plucks, and minimalist looping motives that covers much musical ground and shows how chaotic synthesis can be controlled and used without breaking.

4.8. Movement Two: ***Ferrum***

Ferrum utilizes the Hindmarsh-Rose variation chaotic oscillator from the previous section with different settings to achieve a different character from the oscillator. Two different settings are used and then sampled to be used by an electronic sampler in Logic Pro. The goal is to use the chaotic oscillator in a pleasing and tonal way that mixes well with the guitar and fixed media parts. The first setting is named the *lead synth* and the second setting is called the *transition synth*. The *lead synth* is given a melody through the MIDI roll in Logic Pro and the sampler automatically matches the pitches written. This way pitch shifting is not controlled by parameters in the Hindmarsh-Rose equations which is prone to breaking when certain parameters are reached. This allows for precise pitch control in stable and reliable way. The *lead synth* is used primarily to double guitar parts or provide harmony melodies for the guitar part. This shows the Hindmarsh-Rose variation chaotic oscillator's ability to be used in a traditionally musical way and this process can be reproduced by any composer in most digital audio workstations. Free third-party sampler plug-ins are available for download if a certain digital audio workstation does not contain a native sampler instrument.

The *transition synth* is used in one section of the piece that bridges two consecutive sections. This synth is not controlled by the sampler, but a sample of it was taken from MaxMSP and imported into Logic Pro as an audio track and manipulated within. The quality of the sound can be described as evolving, grainy, droning, and dynamic. Filtering is used to delete unwanted frequencies in the signal and is also used as a musical effect. A low-pass filter is applied with a low cutoff frequency to begin with a dark tone. The cutoff is then gradually raised throughout the length of the sample to raise the brightness of the sound as it reaches the end of its length. Unlike the *lead synth* the *transition synth* is not utilized for its melodic properties, but for its rhythmic and textural qualities which mix with the piece well and serves as a link between the *lead synth* and the drums in the fixed media portion.

4.9. Software Realization

4.9.1. Lead Synth

Figure 4.23 below shows the MaxMSP code that implements the *lead synth*. The sound source is the same Hindmarsh-Rose `gen` object that is used in the chaotic string model. All of the values listed in this figure in the number box object could be hard-coded in the gen object's `codebox`, but it would not be as easy and intuitive for exploring parameters and the sounds that they produce. In this configuration the user can quickly scroll through values and use trial and error experimentation to find interesting sounds. The numbers in this figure set the variables in the `gen` object's `codebox` through the `message` objects which are the four objects in a row second from the top. The variable names are written in with a `$1` appended to them, which tells the `gen` object to assign the incoming values to its respective variable in the `codebox`. An output multiplier of .05 is applied to the signal output to attenuate the volume to an appropriate range and the `sfrecord` object records the audio of whatever is sent in to it which was used to save the sample for use in Logic Pro.

4.9.2. Transition Synth

The same procedure was used for the *transition synth*with different parameter values which is shown below in Figure 4.24.

4.10. Notation

The score for *Ferrum* is notated in a traditional fashion which is shown by an excerpt from the beginning of the piece in Figure 4.25 below.

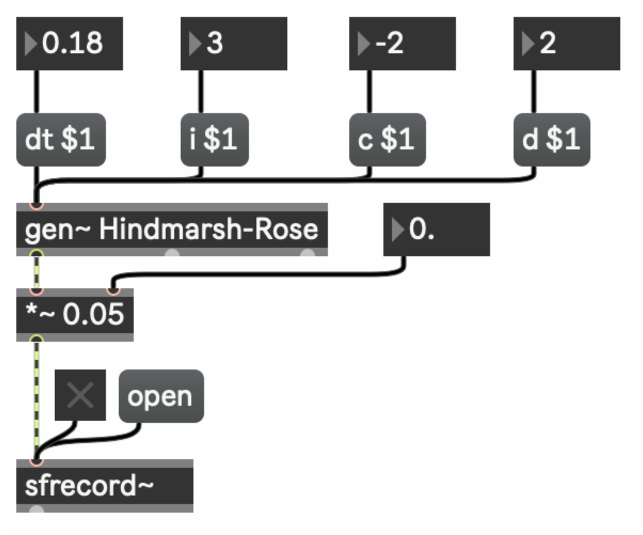


Figure 4.23. MaxMSP code for the Lead Synth.

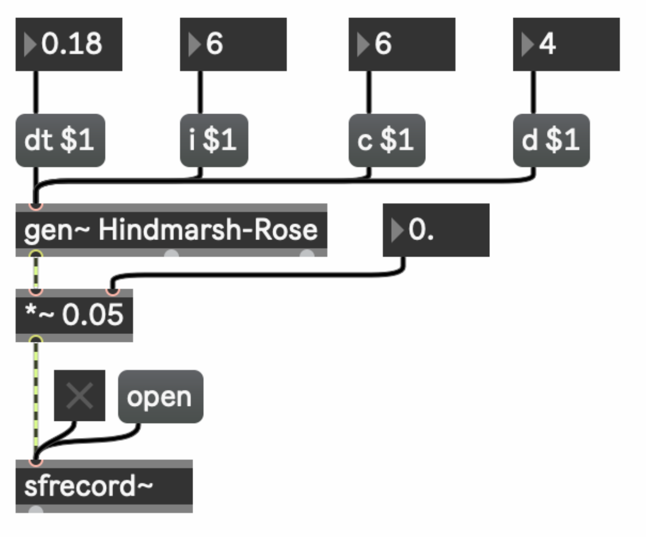


Figure 4.24. MaxMSP code for the Transition Synth.

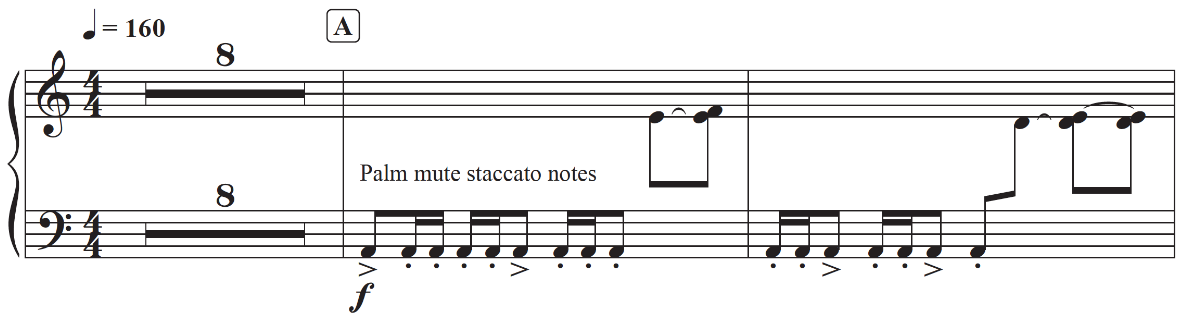


Figure 4.25. Score example from *Ferrum*.

CONCLUSION

The process of connecting chaotic synthesis algorithms to individual parts of synthesis engines proves to be a musical way of using chaotic processes in a reliable and modular way. The Hindmarsh-Rose chaotic oscillator is encapsulated in a MaxMSP gen object. It can be downloaded and added to anyone's MaxMSP project and used in the same way as the current project or in other creative ways such as frequency modulation, amplitude modulation, ring modulation, or sampled and loaded into a buffer for granular synthesis. The audio created by this object for the chaotic string in the *ambi drum* model adds a novel element to the digital waveguide sound that is familiar to so many computer musicians and proves an excellent sound source to compose with. More compositions need to be written for this instrument to explore settings and interaction methods yet to be discovered by the current project.

The chaotic drone created with the circle map is an effective virtual instrument used as a bass instrument in this project, but it is merely one setting for the circle map. It is clear that there is more potential for the instrument. Other computer musicians can download it and change variables to generate their own versions of the instrument to be used in ways currently not realized.

Implementing chaotic equations as audio oscillators and effects is achieved with minimal issues by experimenting in MasMSP's `gen` environment. This environment facilitates uncomplicated digital signal processing with a coding interface for simple c-like mathematical programming. The relative ease of this process is unexpected, but other issues are present after the implementation process. The challenge that arises unexpectedly is the exploration of parameter space in the chaotic equations. This behavior is sometimes encountered when working with chaotic equations and their parameter space. This is the behavior of exploding when undesirable values are computed by the chaotic equations. There is another unexpected behavior exhibited by the chaotic equations when exploring parameter space which is the unpredictable sections of parameter space that transition between stable and unstable behavior. For example, taking the circle map into account because it only uses two variables a single number box was connected to both variables. As the numbers progressed up in a linear trajectory in one direction at a constant speed the behavior in the equations transitioned between stable and unstable conditions. This means that it was not as simple as having one known range that exhibits chaotic behavior and one range that exhibits stable behavior. This made the mapping process of the virtual instrument implementation more difficult than previously expected, but this was avoided by simply finding a small range to work in by way of trial and error experimentation. Once a desirable sound setting was found, the range of variables was clipped to that range so that the chaotic equations never exploded.

It is believed that using some type of machine learning process could be used in future research to explore this parameter space more effectively. For example, a genetic algorithm process could create parameter boundaries, listen to the sound produced, assign it a fitness value based on how desirable the sound is, and then use the most fit members of that gene pool to produce new parameter boundaries to explore. The issue that this poses however is that it is possible this process will produce values that cause the chaotic equations to explode. Checking for chaotic equation exploding with JavaScript and writing a script that re-instantiates the MaxMSP chaotic equation object when this occurs could be a way to remedy this issue and allow the genetic process to proceed independent of a controlling user who would otherwise need to manually re-instantiate the chaotic equation object and interrupt the genetic process.

APPENDIX.A
XIII TRACTUUM SCORE

VIII Tractuum

For Hexapad

Landon Viator

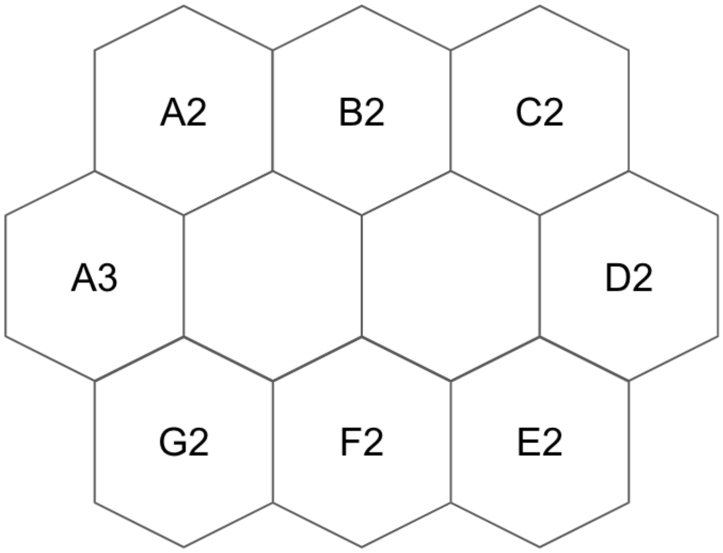
Fall 2019

Introduction

VIII Tractuum is Latin for Eight Pathways, which represents the fact that the Hexapad is used in an eight-channel arrangement for the composition. In concert, *VIII Tractuum* is connected to eight different loudspeakers. As the performer moves the brush around the pads, rhythms are created in time and in space in the same arrangement as the layout of the pads. *VIII Tractuum* focuses on the Hexapad as a solo instrument controlling a conceptual percussion instrument model inspired by hang drums. The Hexapad is explored and focuses on how much musicality it can create on its own as a solo instrument much in the way solo acoustic instrumental pieces are realized.

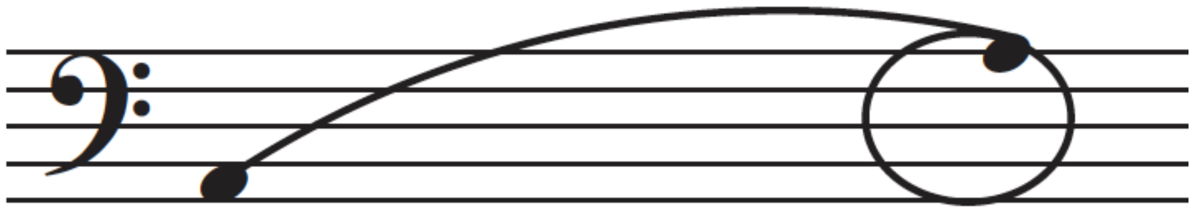
Legend

Time codes are used in place of measure numbers so that the improvisational intent of the piece is easily facilitated allowing the performer more rhythmic freedom. Each pad on the Hexapad is tuned to a note in the A minor scale beginning on the first pad with A2 and ending on A3 on the last pad. The figure below shows the note mapping that the Hexapad is tuned to. Solid note heads represent the striking of the pads that correspond to the written pitch in the staff. These pitches can be realized in any rhythm, which is why they are stemless. The whole notes in the A2 space indicate the drone sound which is automatically turned on by the patch. Text above the staff will indicate to turn a looping buffer on or off that automatically repeats a certain amount of audio. This button is implemented in the patch with a labeled toggle. The text will also indicate turning certain synth parameters on or off with their own toggle buttons as well.



Paths

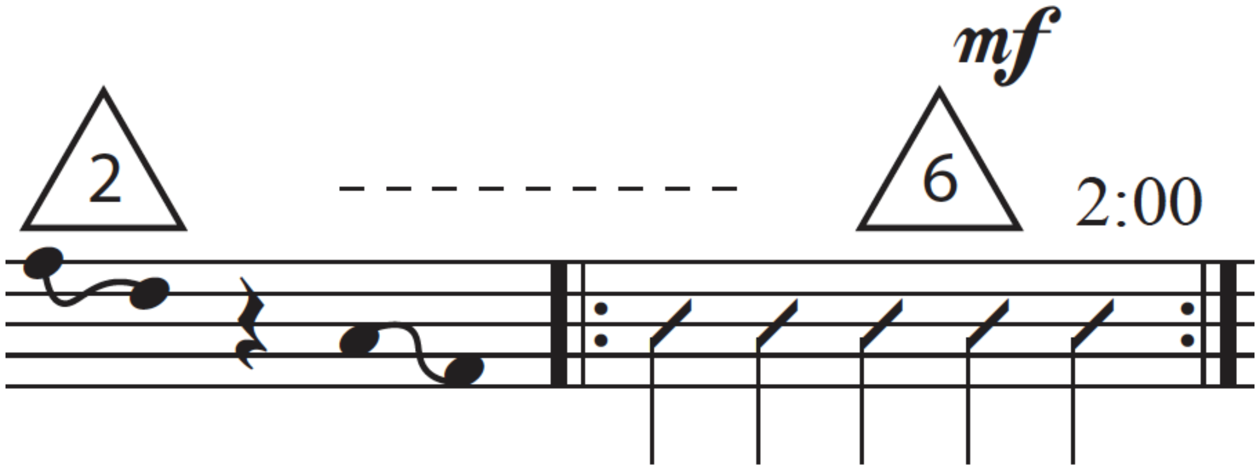
One of the main interactions with the Hexapad in *VIII Tractuum* is brushing of the pads in discrete paths with a brush. Lines are drawn in the staff to indicate the direction/pattern of the brush strokes. The figure below shows an example of this from the score.



Beginning on the first pad, this figure shows a clockwise brushing motion up to the last pad with a quick motion back towards the first pad and quickly brushing back up to the last pad. The performer is meant to follow these paths with the brush across the pads in the direction that the path indicates.

Pitch Presets

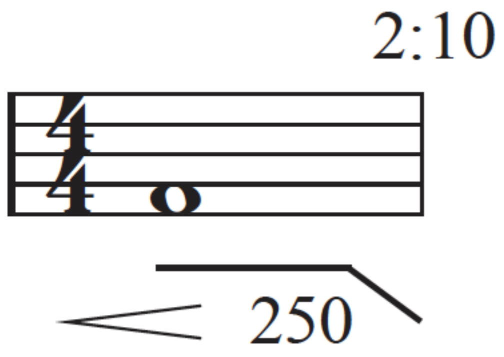
There are points in the piece where the pitch of the Hexapad's pitches are changed by the performer in the form of a pre-defined delay line in the MaxMSP patch attached to a control dial. The figure below shows how these pitch changes are indicated.



This figure shows the triangular shapes that indicate the pitch change. The dial has a range of 0 - 127, so the triangles will indicate a value in that range to select. This figure also shows the slash notation which indicates to improve until the timestamp on the next measure.

Filter

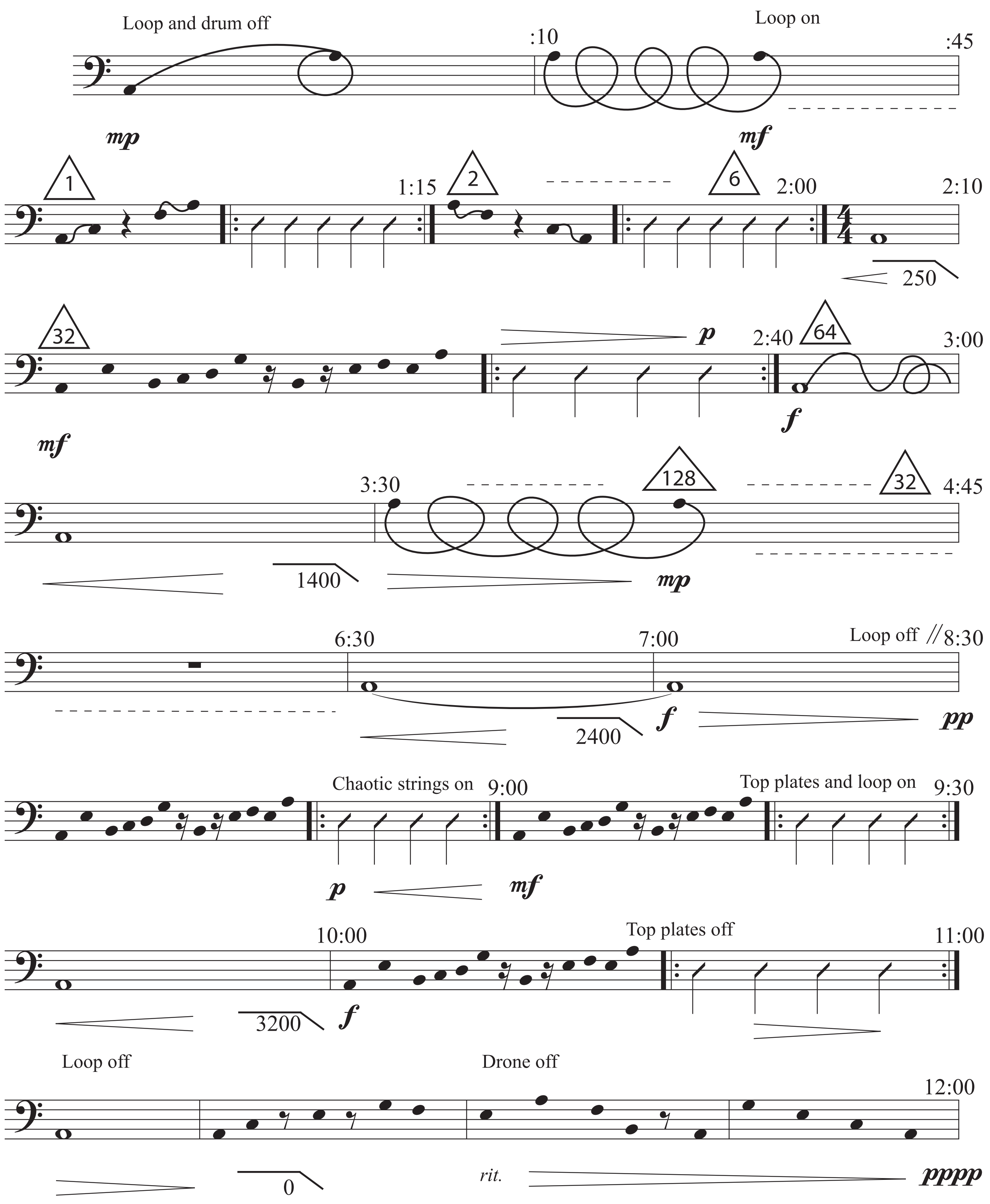
The slanted lines represent the cutoff frequency for the filter that is applied to the drone sound, which is indicated by integers in Hertz. Crescendo and decrescendo markings are used in conjunction with these cutoff symbols to indicate movement from one cutoff frequency to another. The figure below shows an example of this filter symbol.



In this figure, the filter's cutoff frequency is changing from 0 to 250 Hz.

VIII Tractuum

Landon Viator



©

APPENDIX.B

FERRUM SCORE

Ferrum

For Guitar and Fixed Media

Landon Viator

Fall 2019

Introduction

Ferrum is Latin for iron, which is related to the fact that it is inspired by progressive metal music and draws inspiration from such artists as Jason Richardson, Animals as Leaders, and Mike Gianelli where tonality and voice-leading practices are more traditional than in the current paradigm of experimental and computer music. *Ferrum* is for 8 string guitar and fixed media.

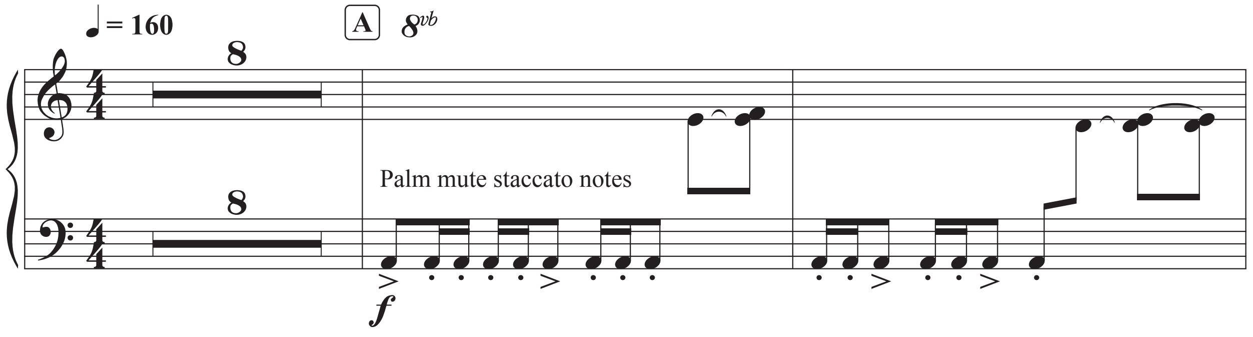
Legend

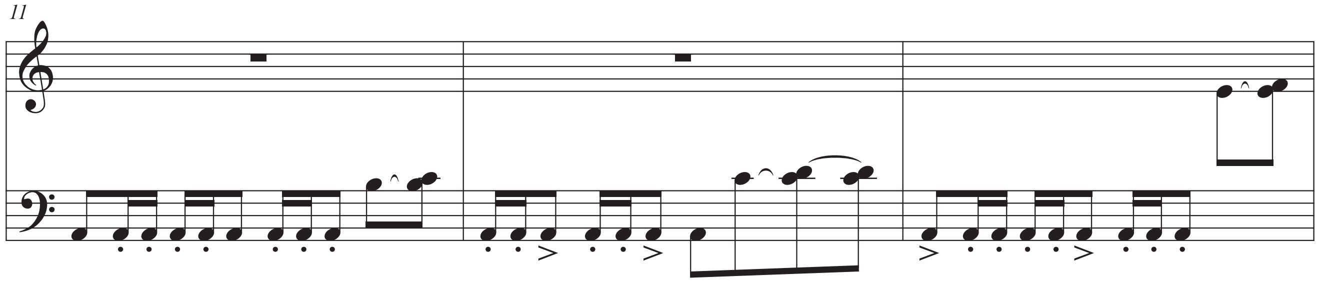
The fixed media part is methodically scored out and does not allow for the same improvisational actions as *VIII Tractuum*. Since this is for 8 string guitar, all notated pitches should be played an octave below what is written so that the low A string on the guitar is note A2 in the bass clef. All stacatto notes are to be played by using palm muting as close to the bridge of the guitar as possible. The slash notation from measures 149 - 152 indicate a solo section that the guitar can improvise or prepare material for. The length of the measures and repeats need to be performed exactly to line up rhythmically with the fixed media part.

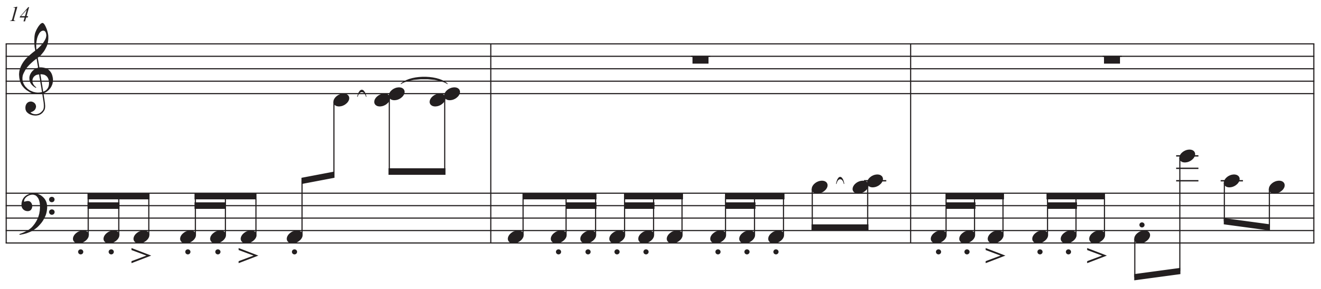
Score

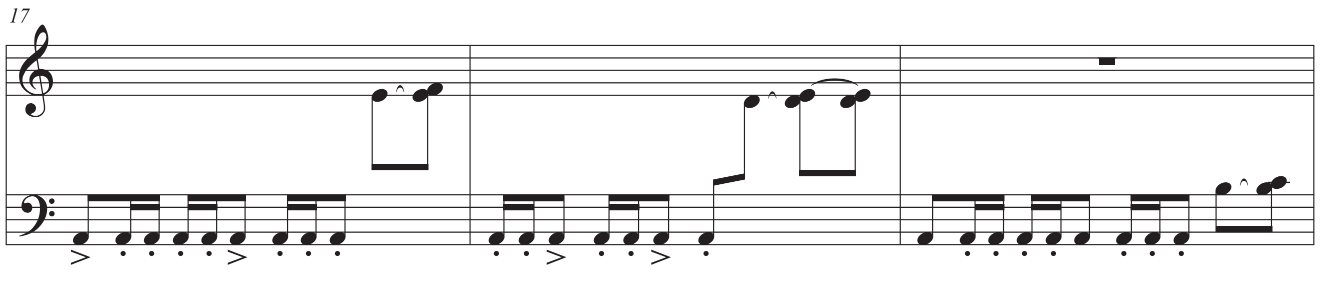
Ferrum

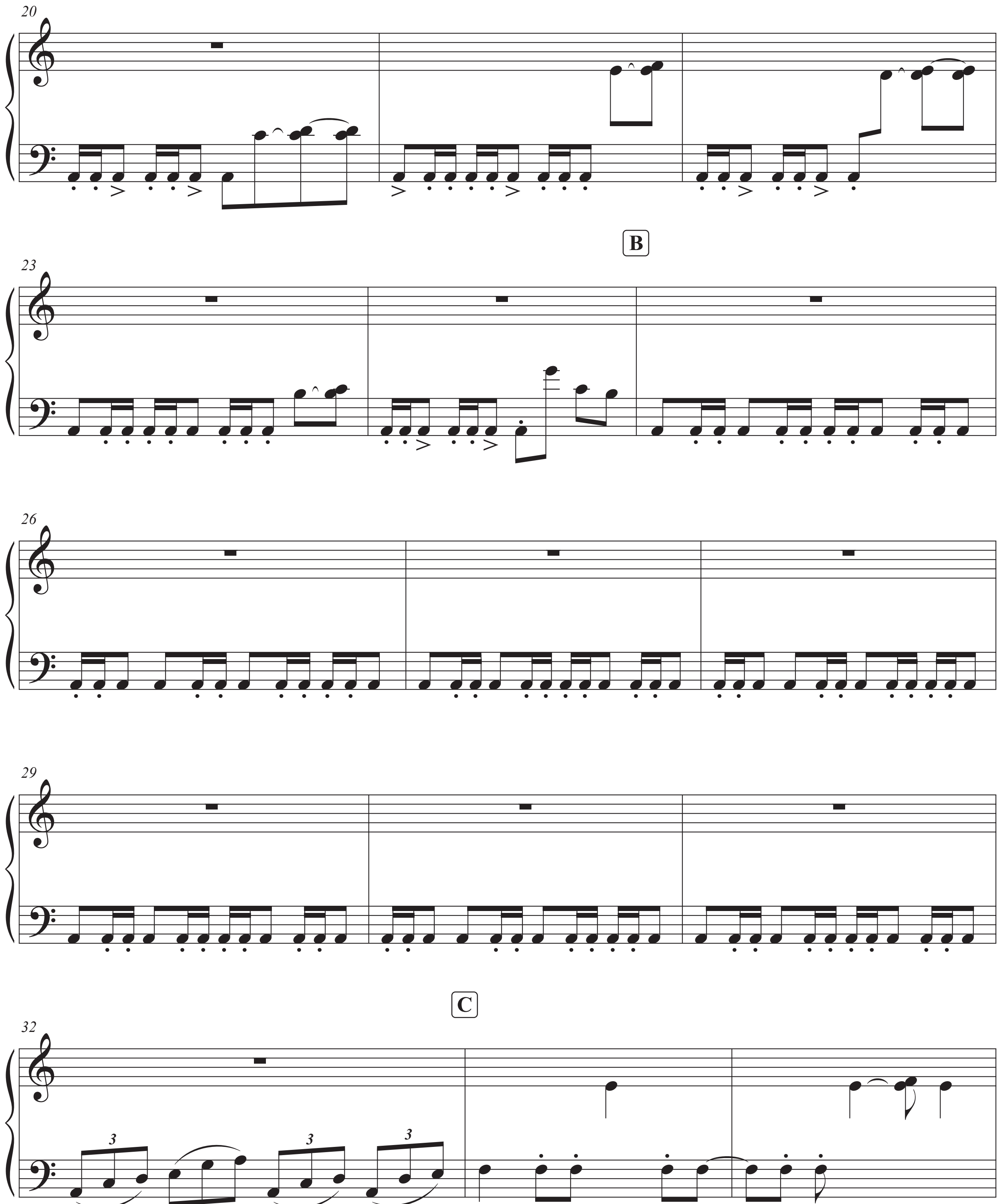
Landon Viator

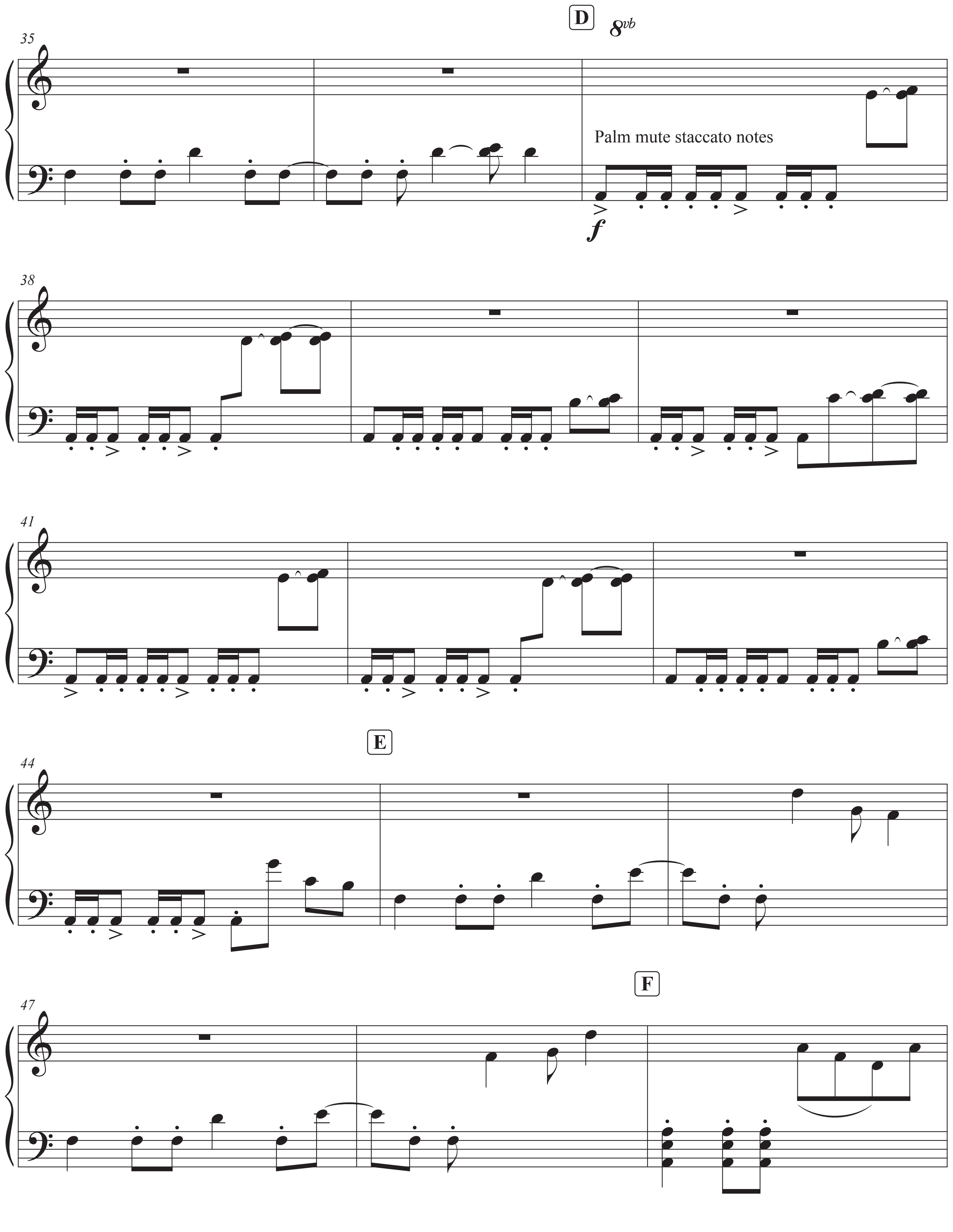


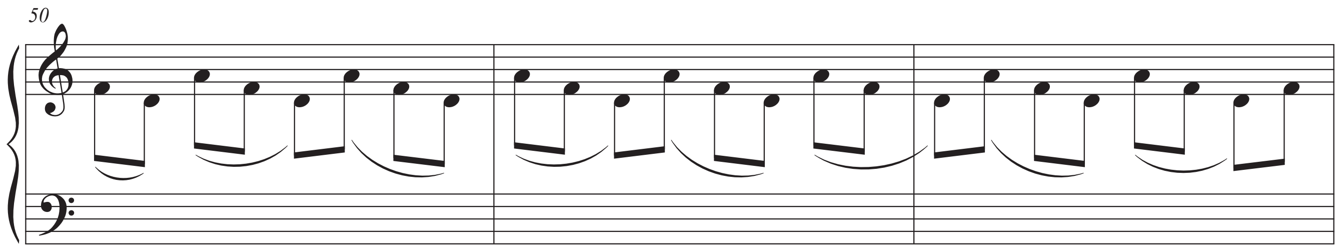


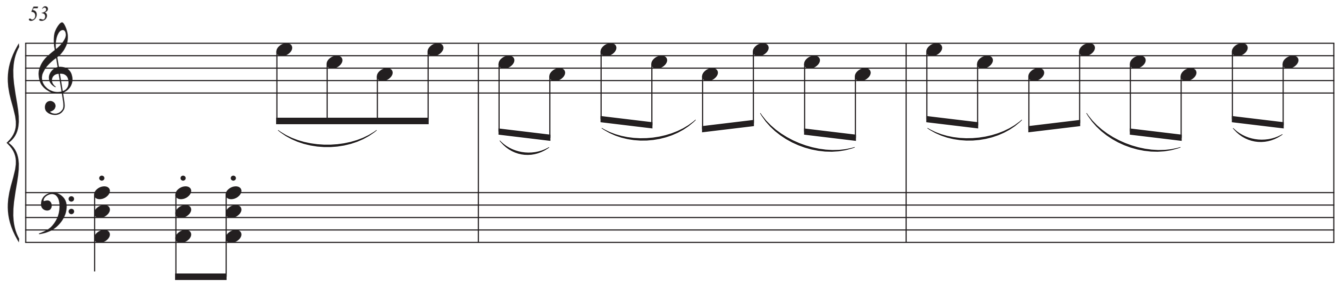


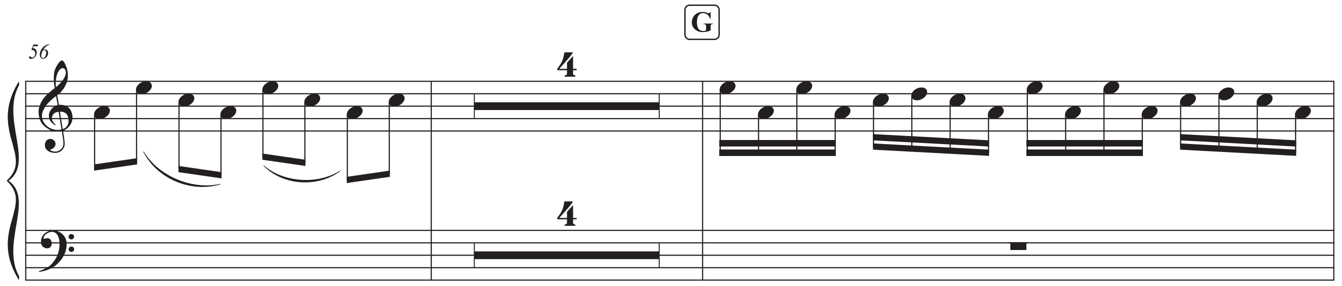


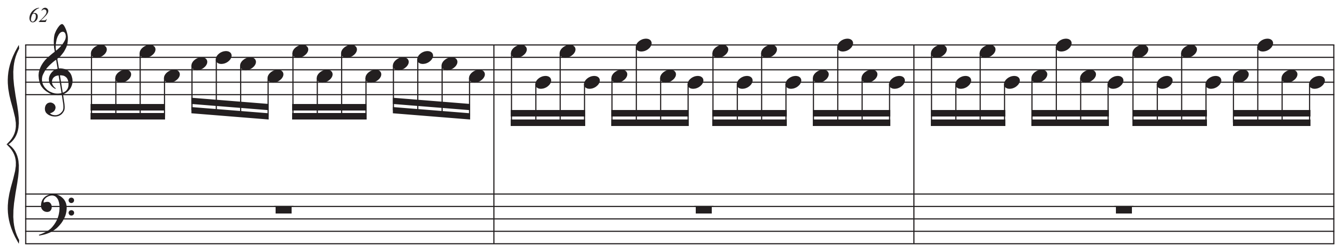

20
23
B
26
29
C
32
3

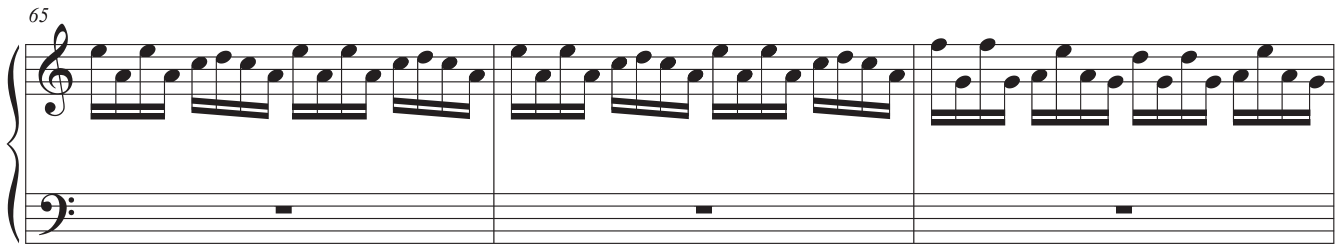

D
8vb
Palm mute staccato notes
f
E
F
35
38
41
44
47

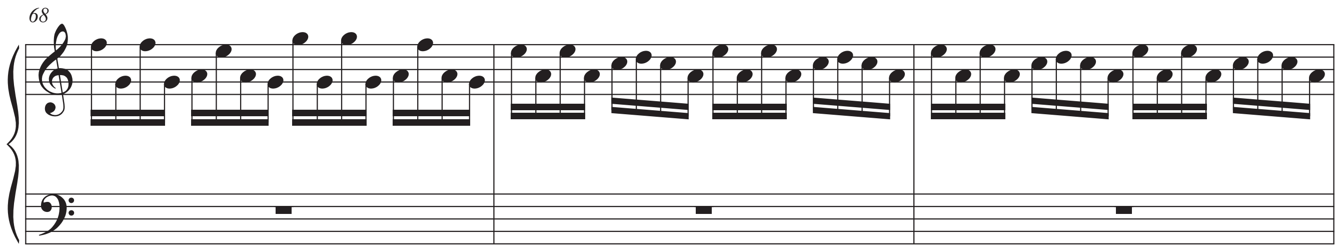


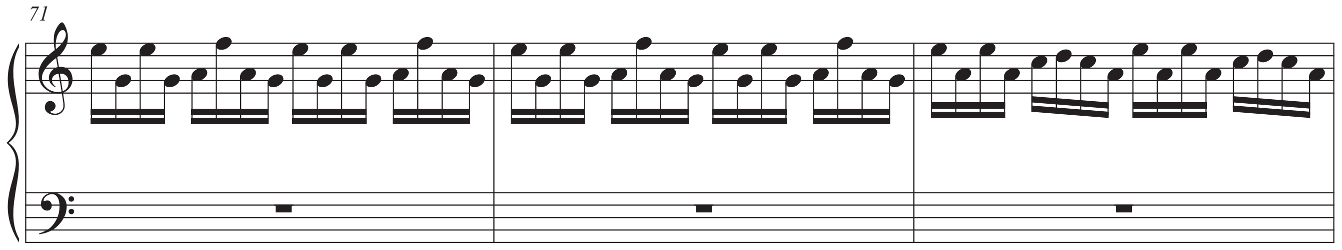


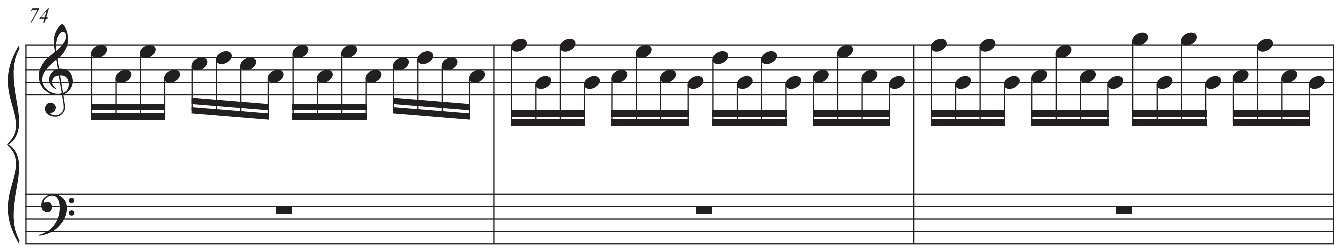

G
4
4

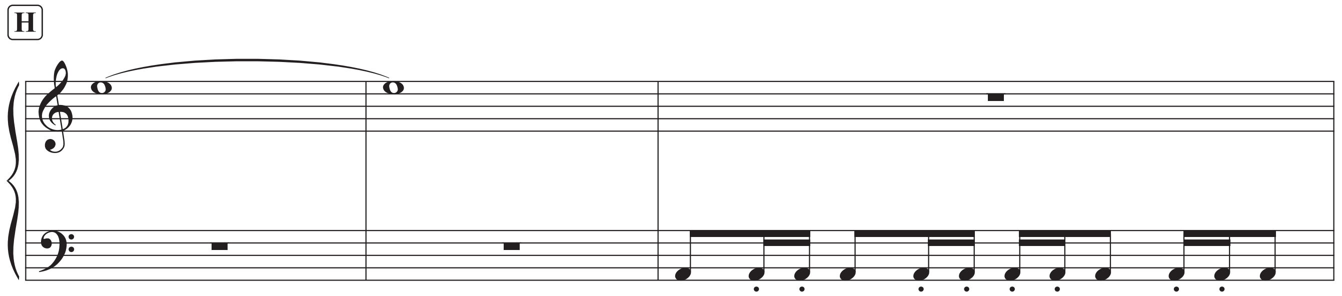


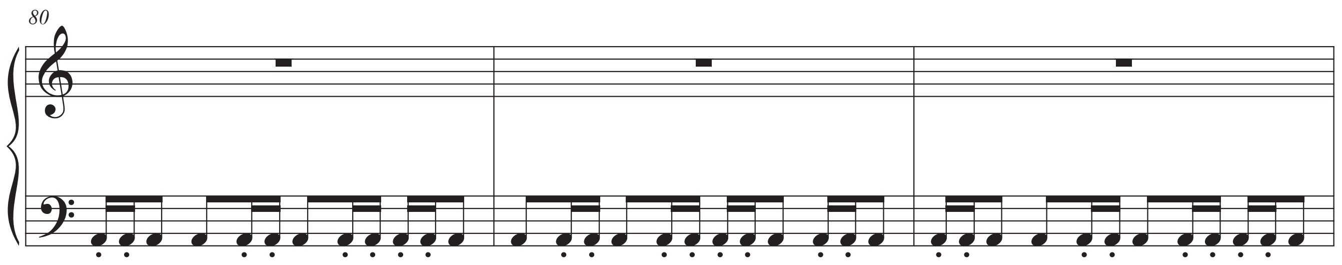


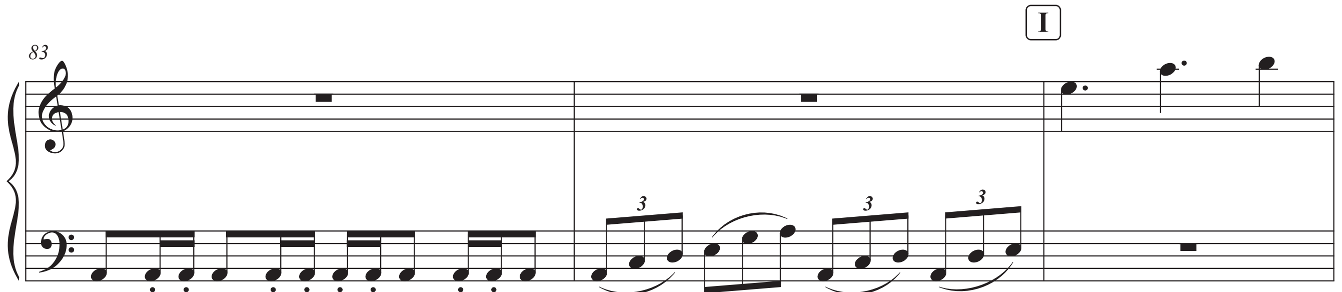

68

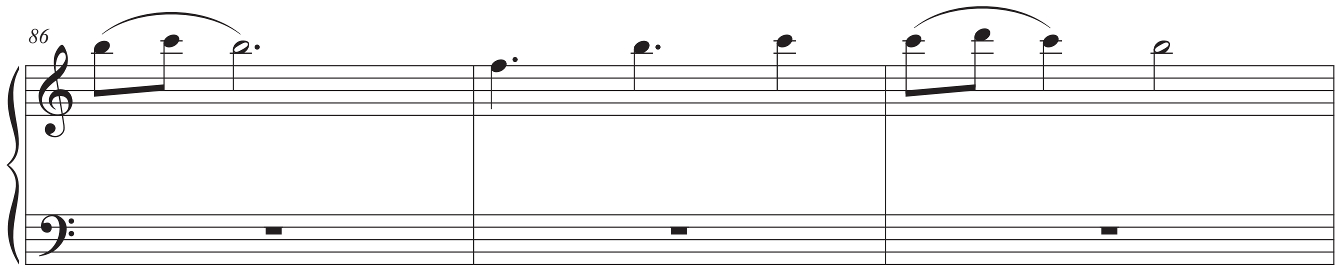

71

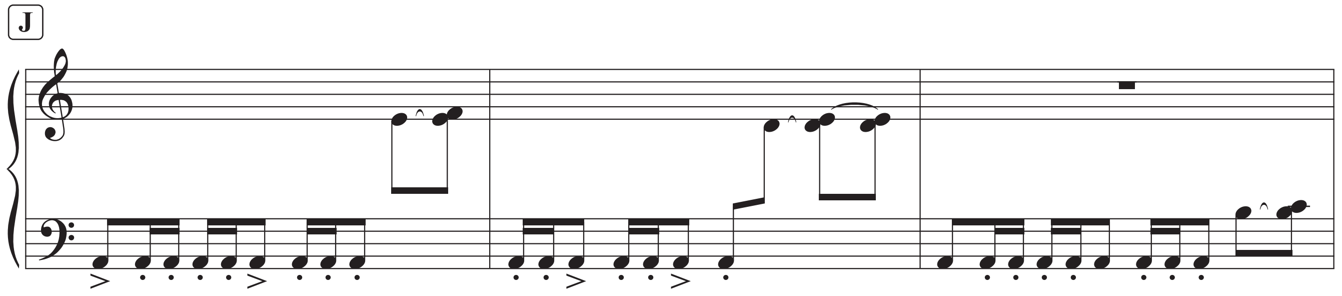

74

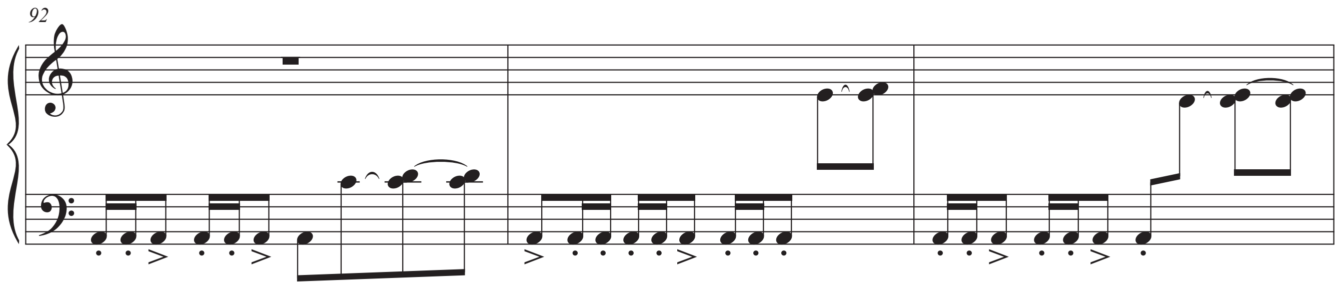

H

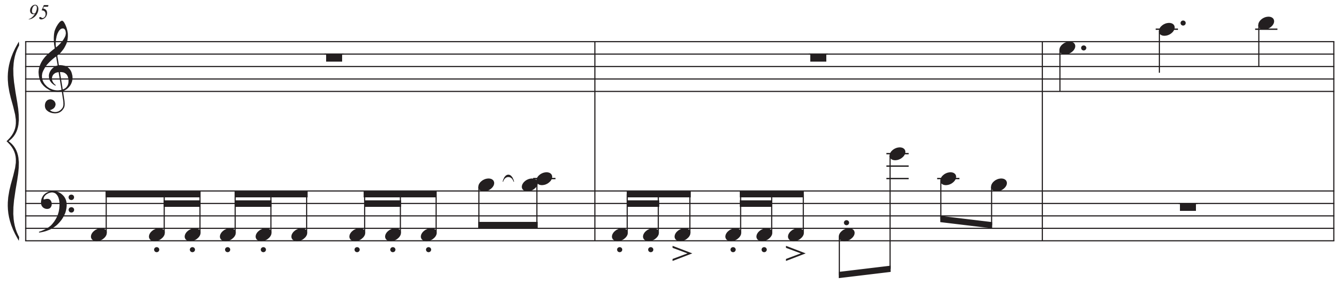

80

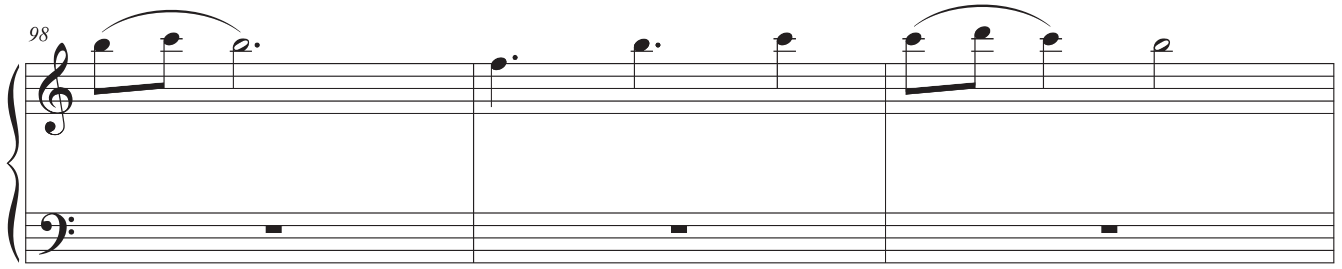

83
I
3
3
3
3

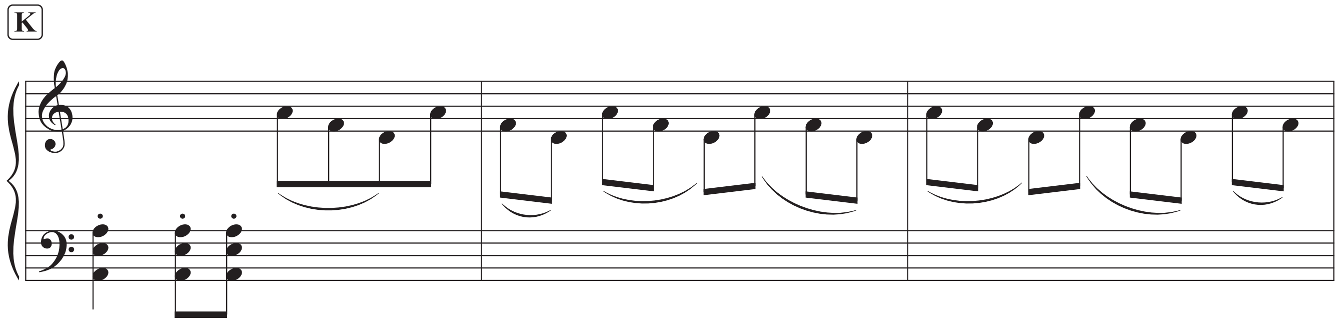

86

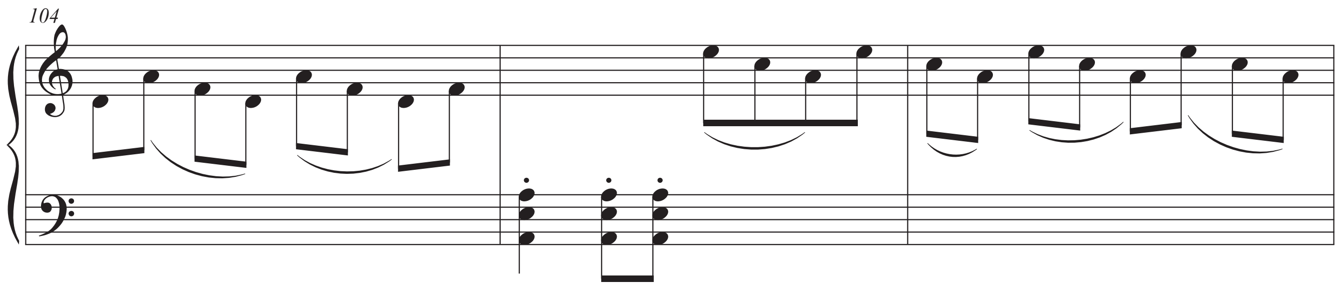

J

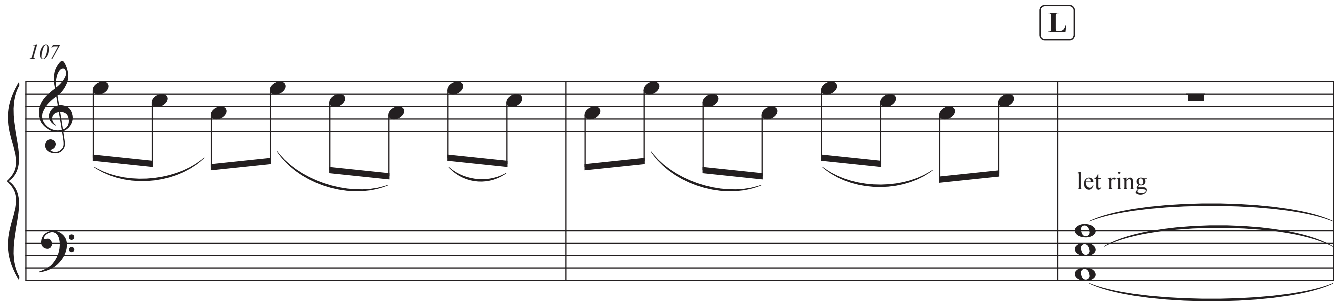

92

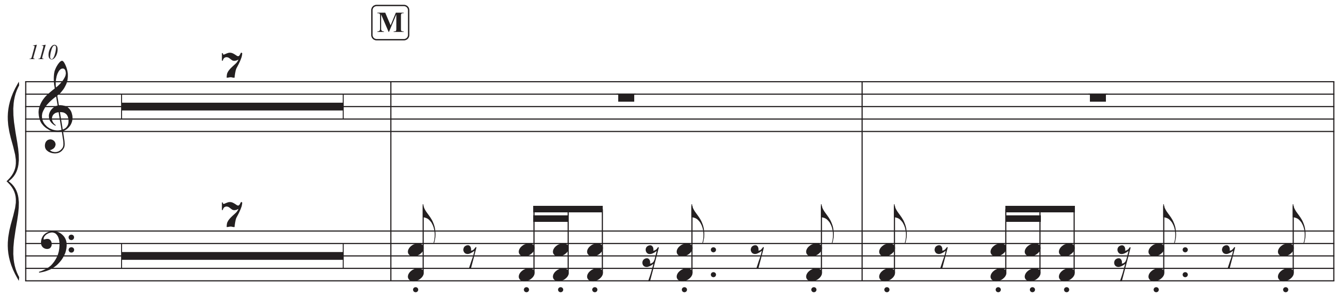

95

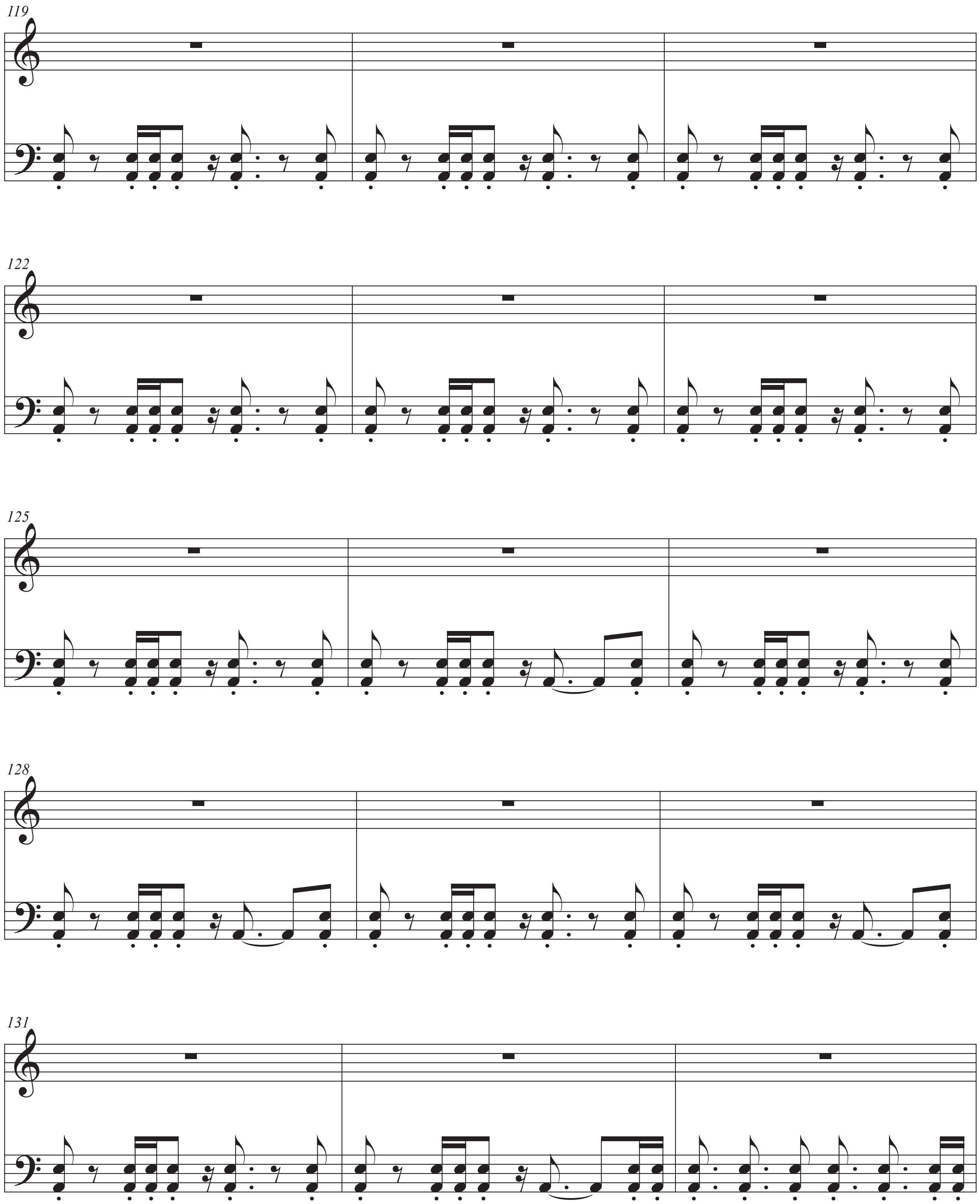

98

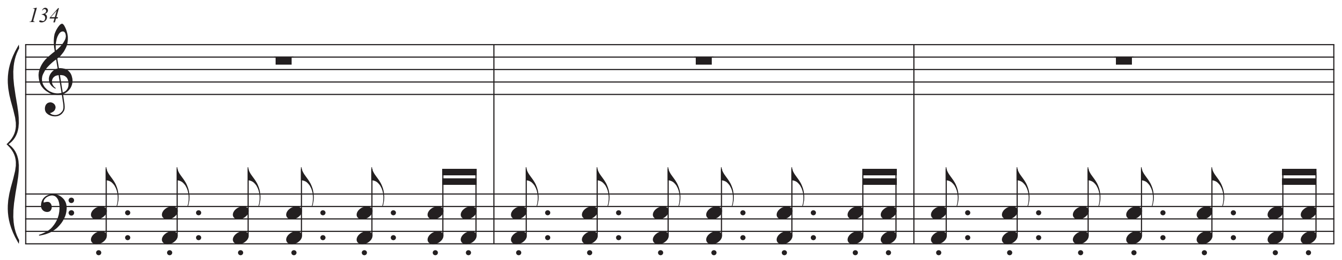
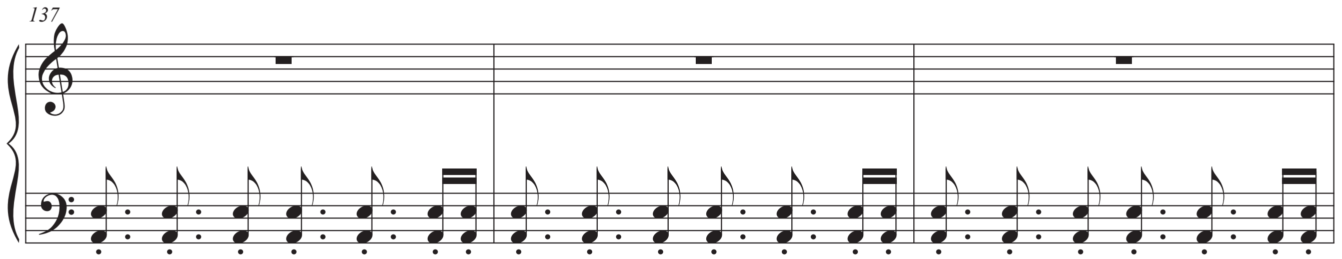
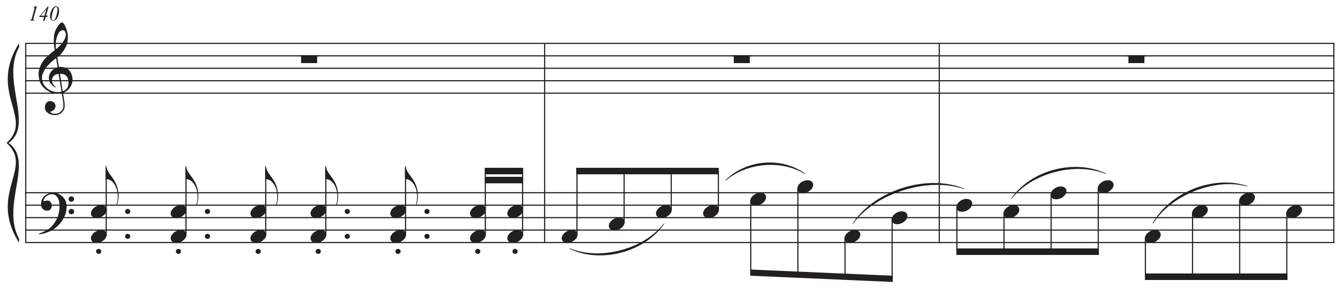
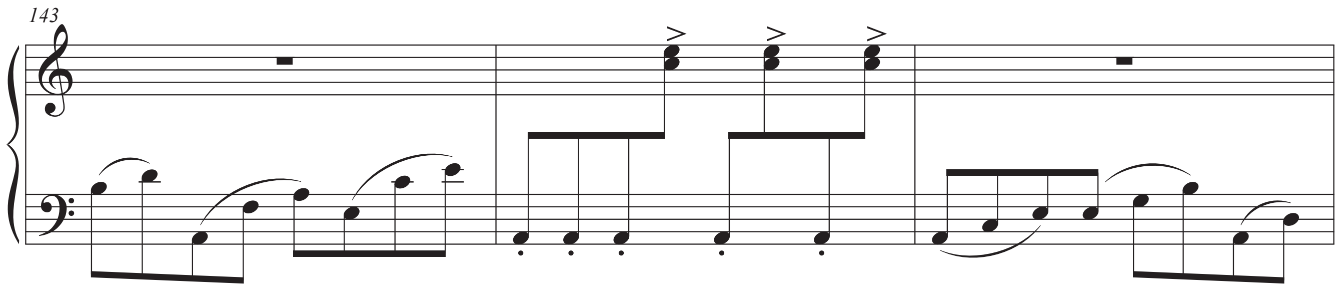
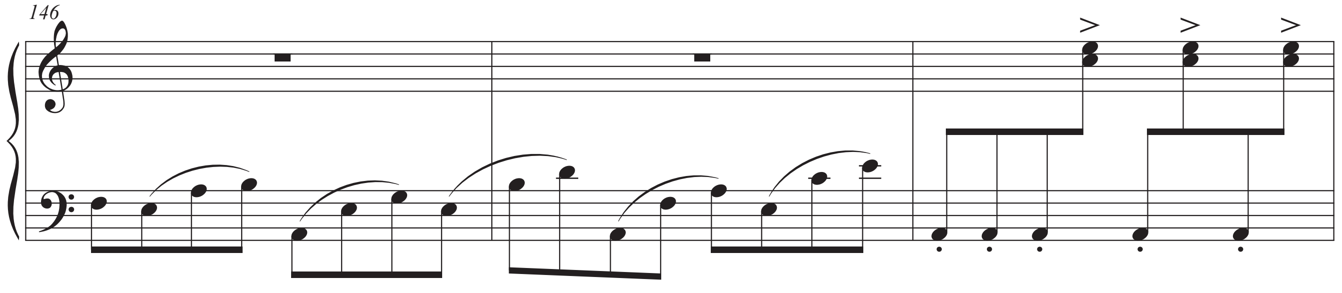

K

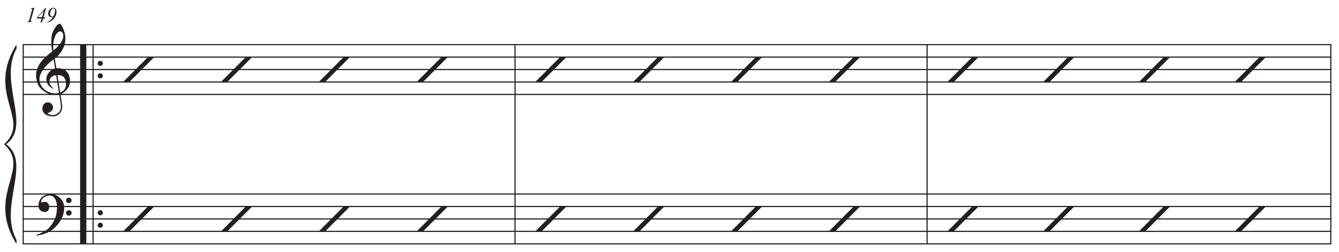

104

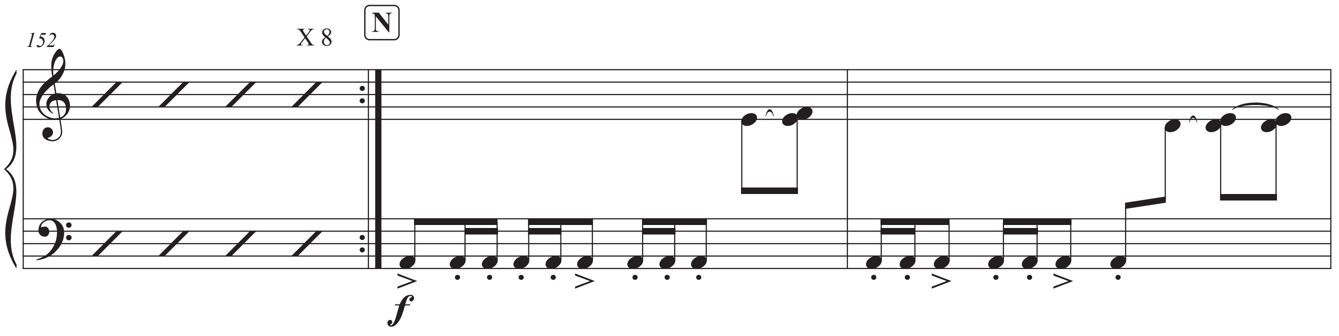

107
L
let ring

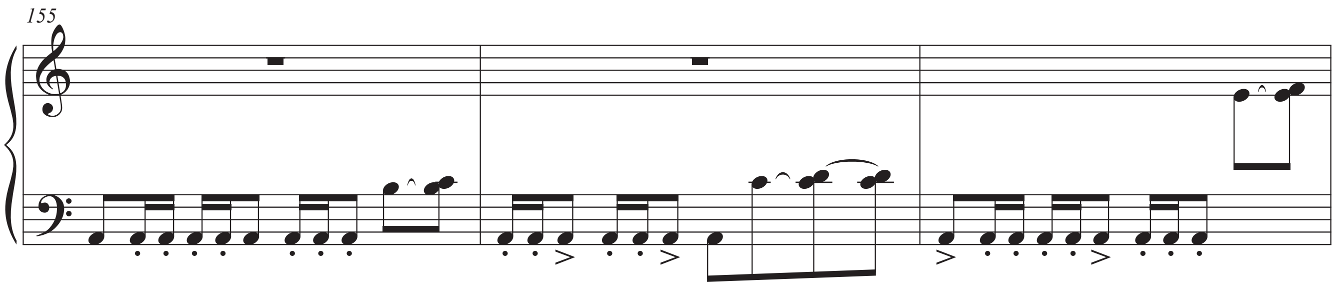

110
M
7
7

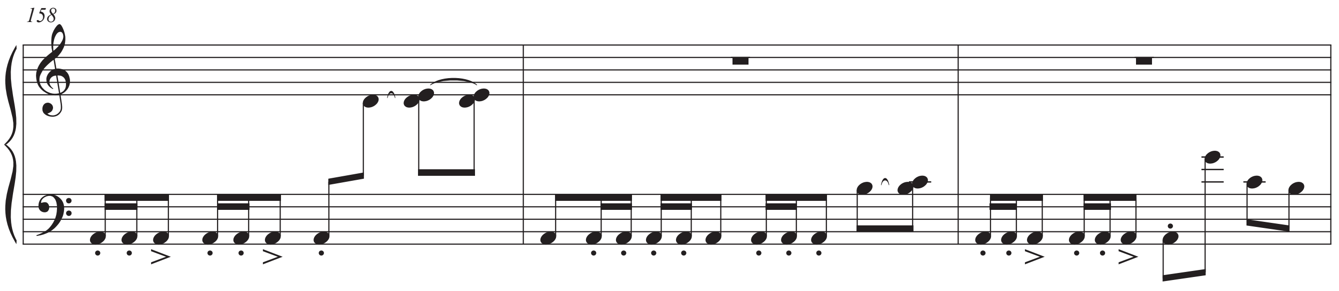


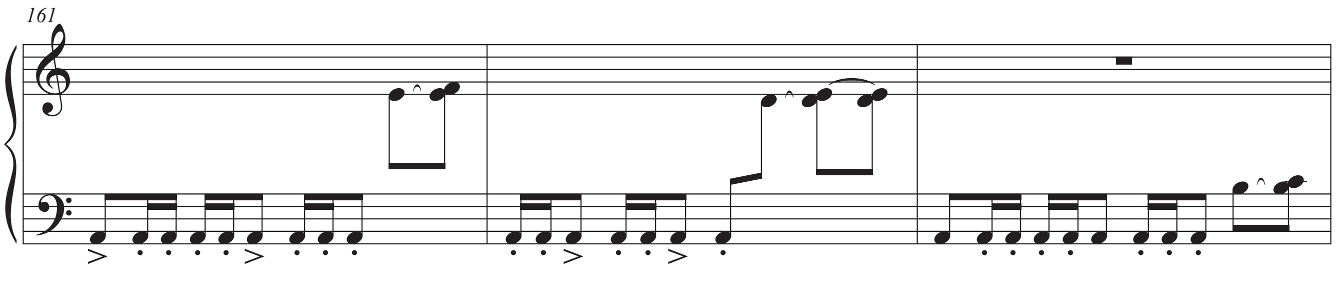






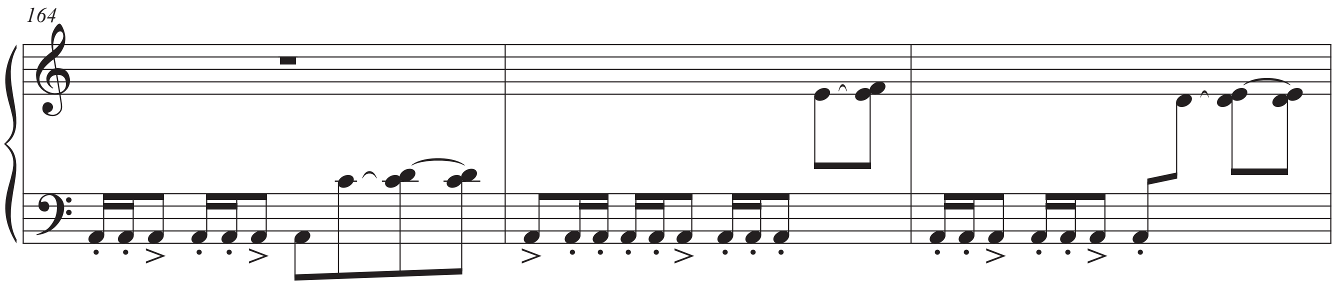

149


152
X 8
N
f

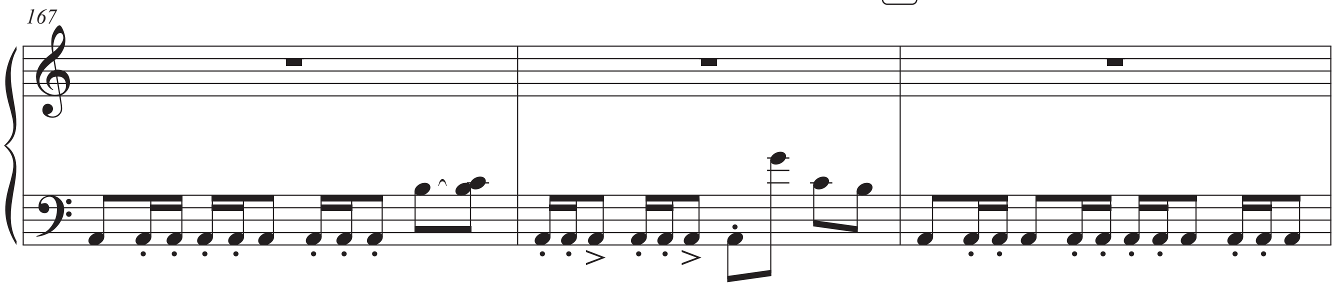

155

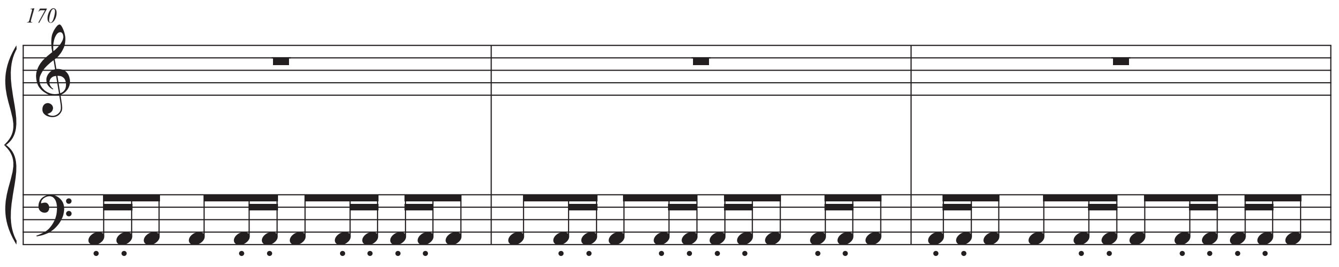

158

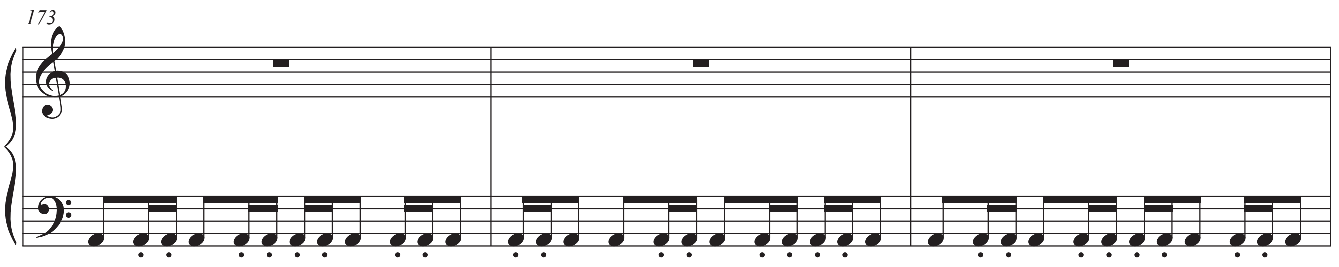

161



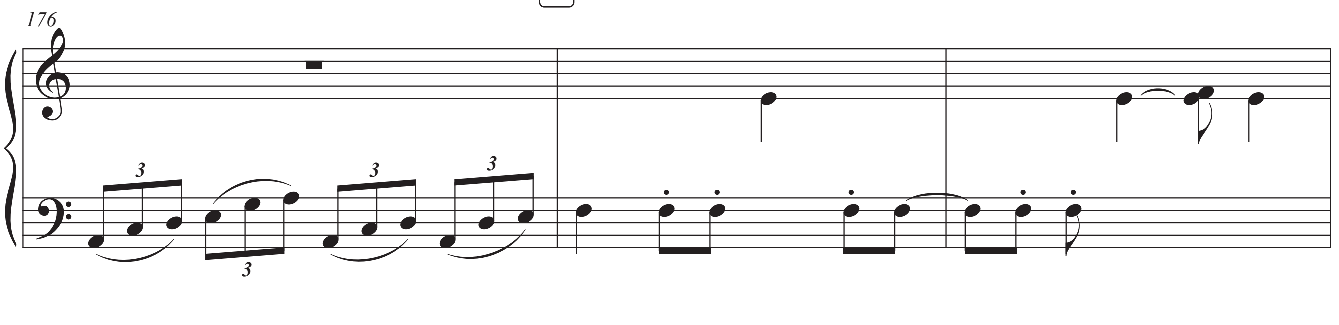
O

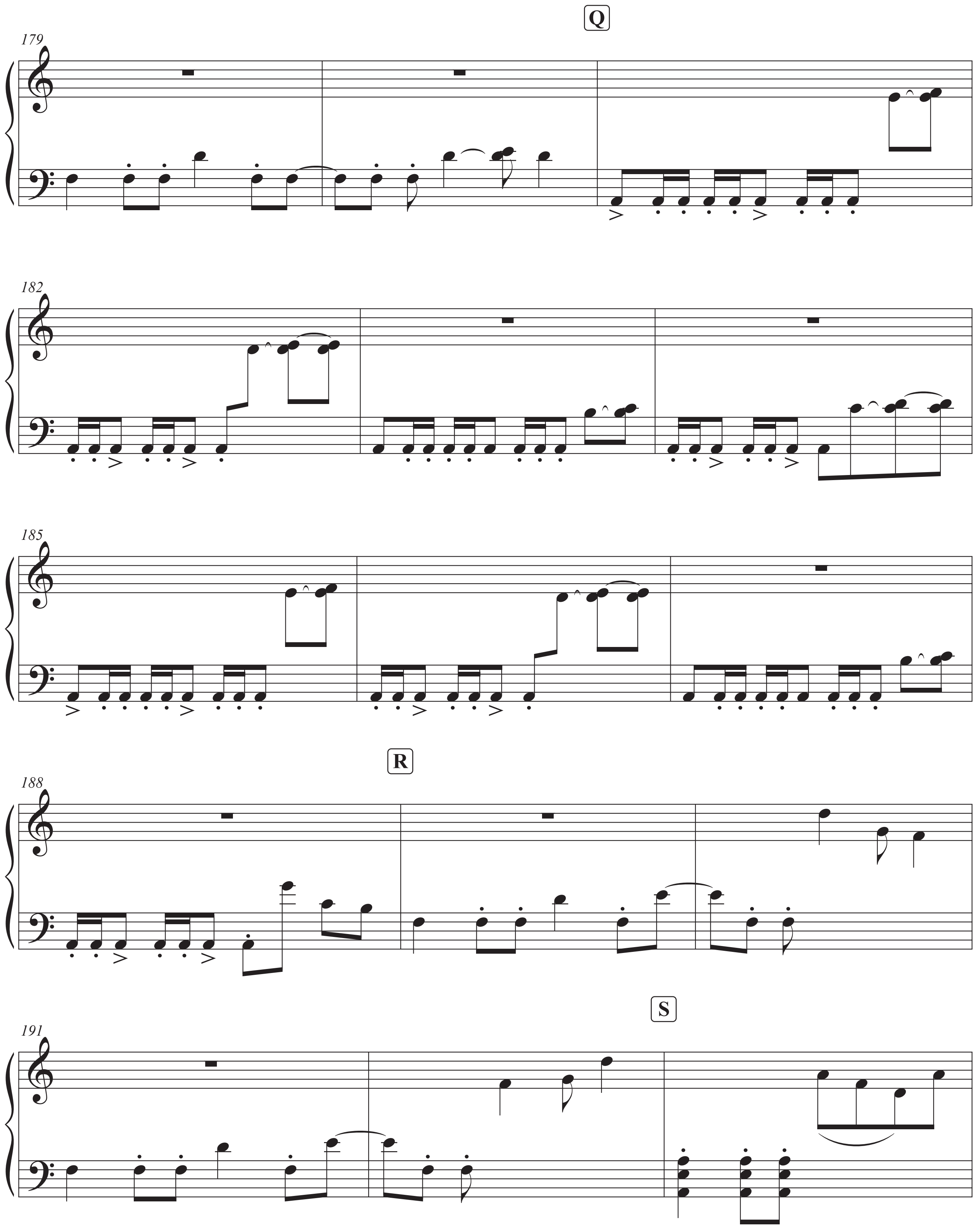


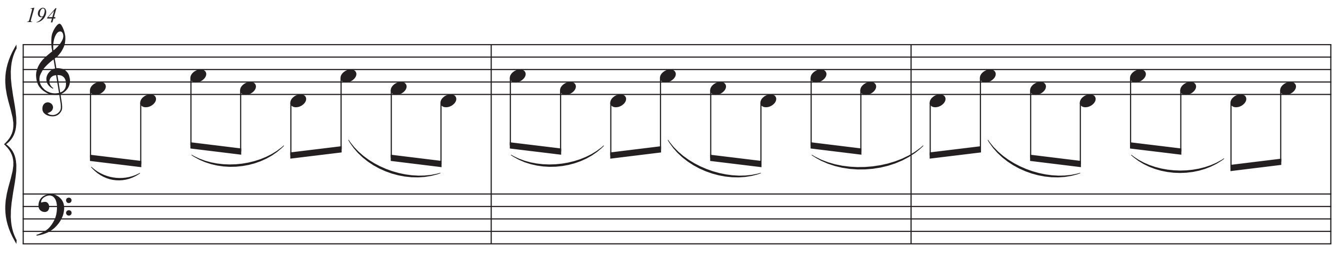
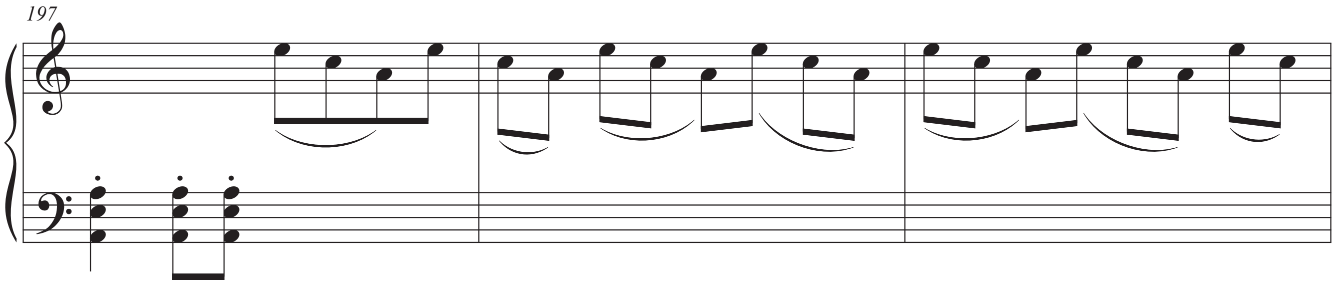
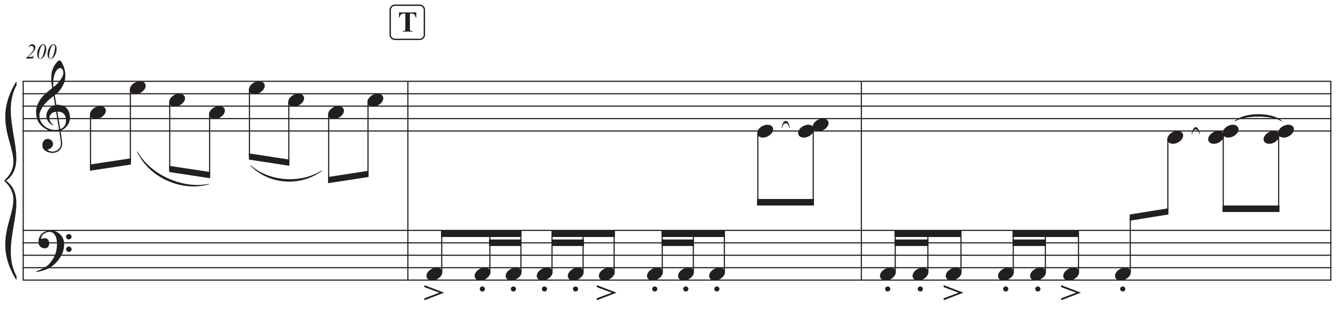
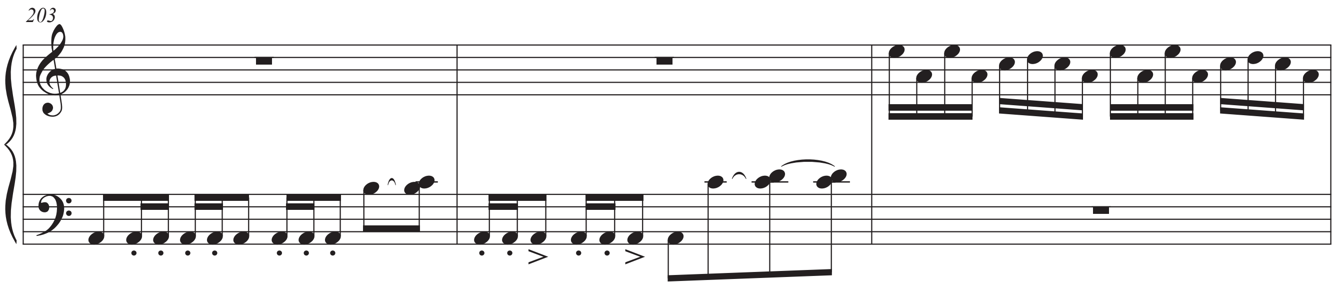
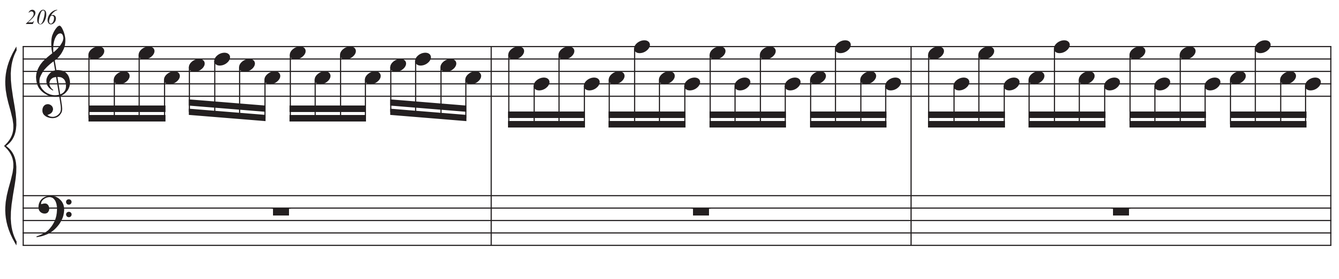


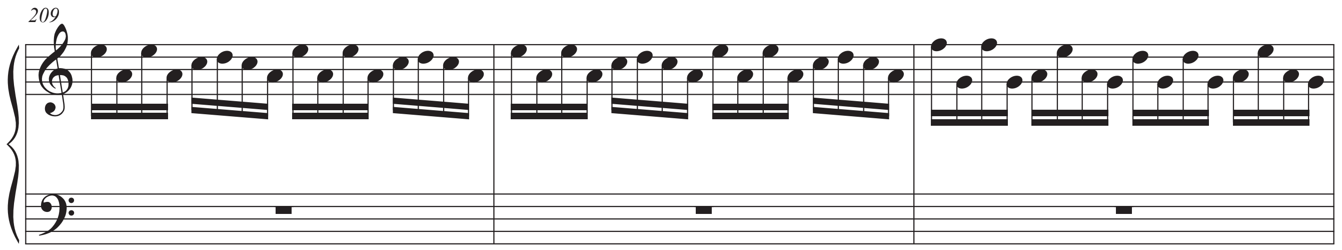


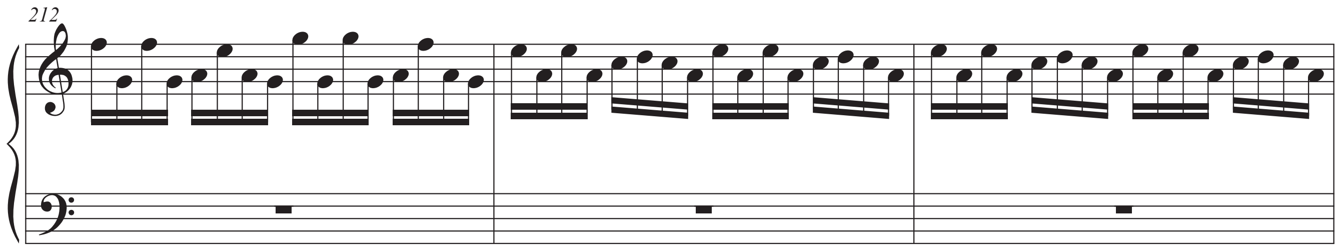
P

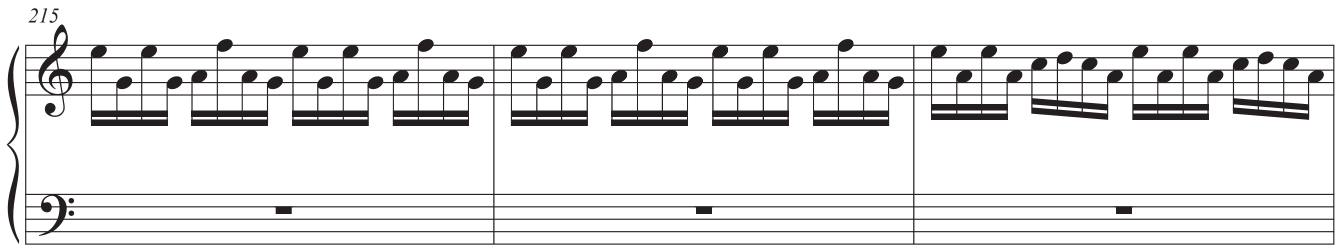


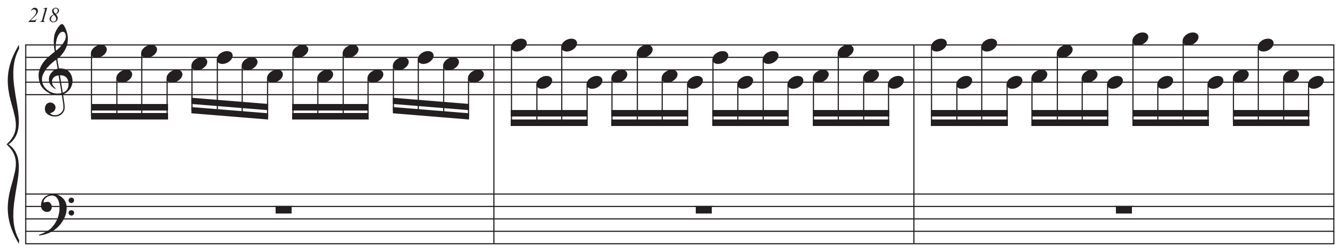

Q
179
182
185
R
188
191
S

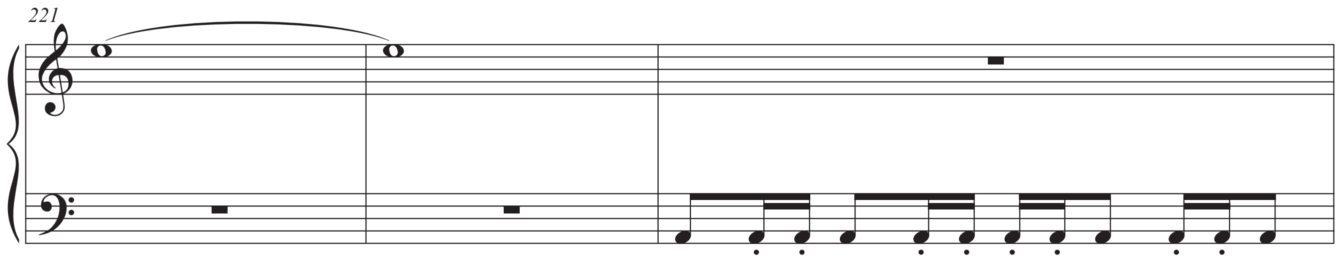





194
197
T
200
203
206

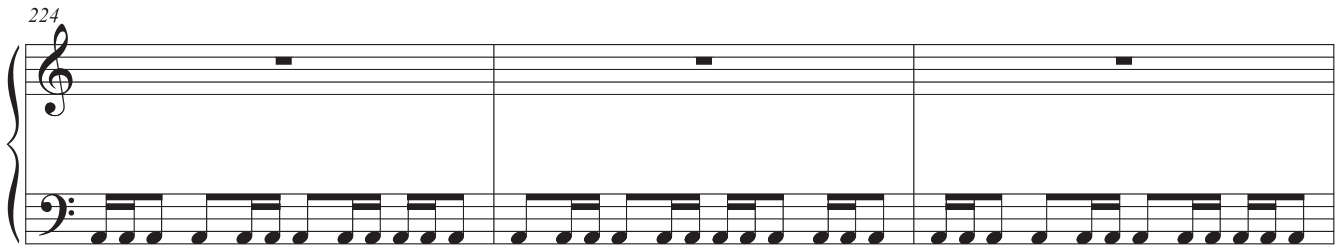


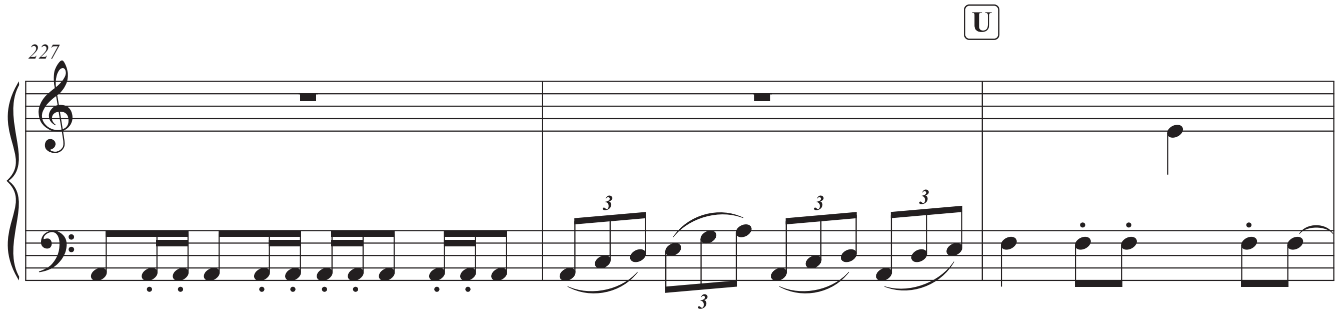


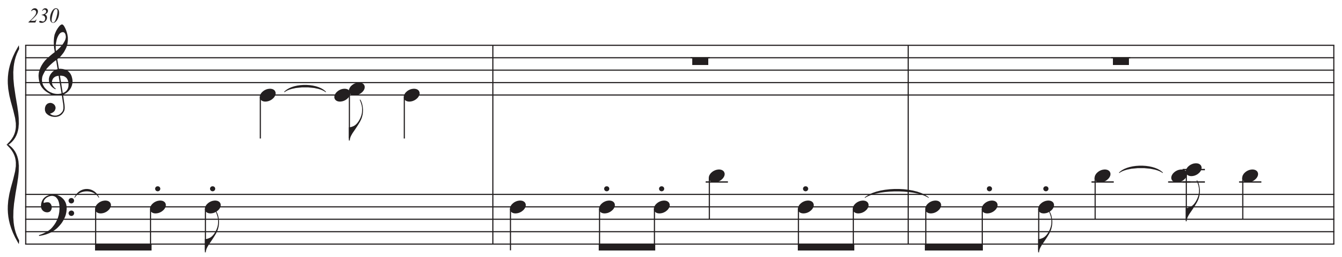


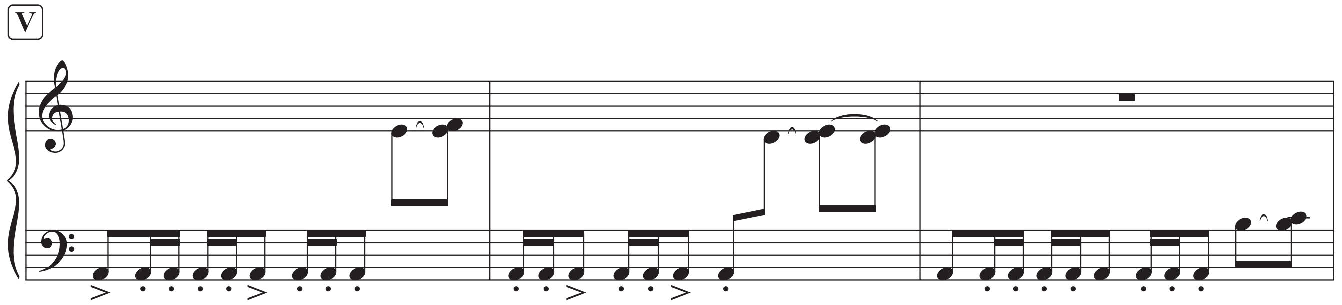


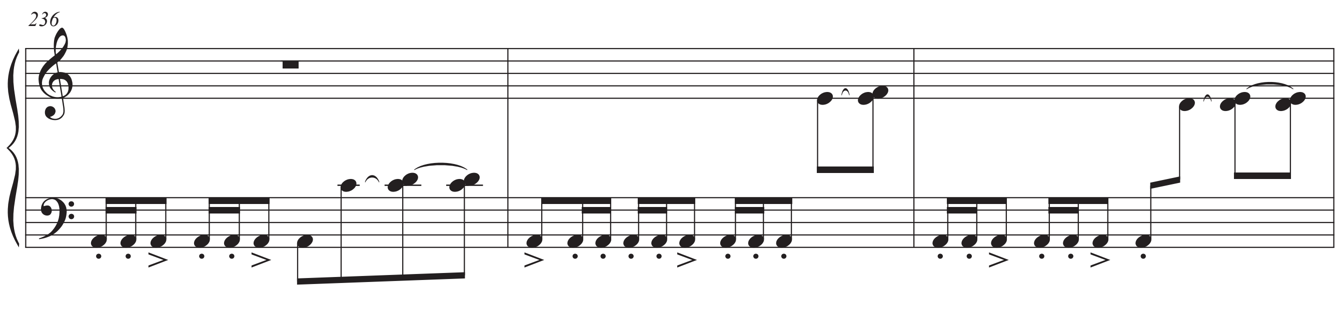




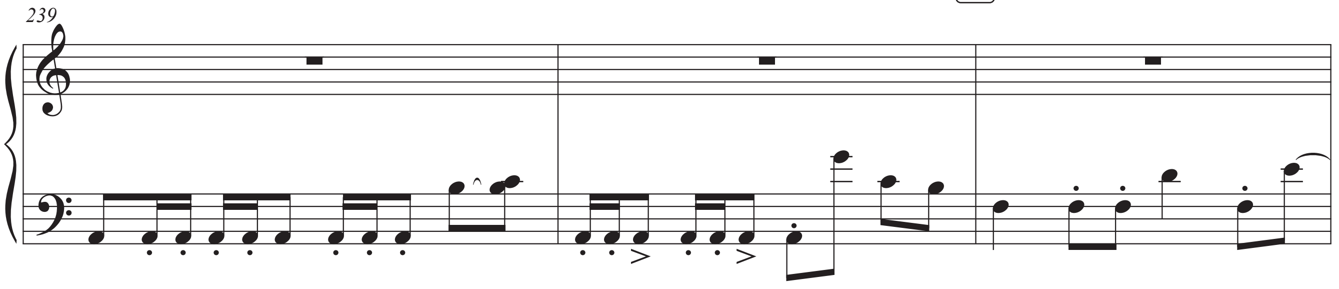


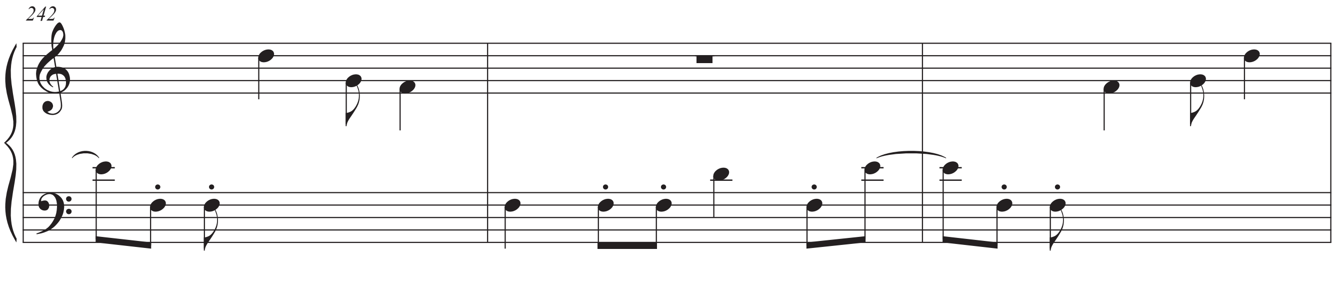




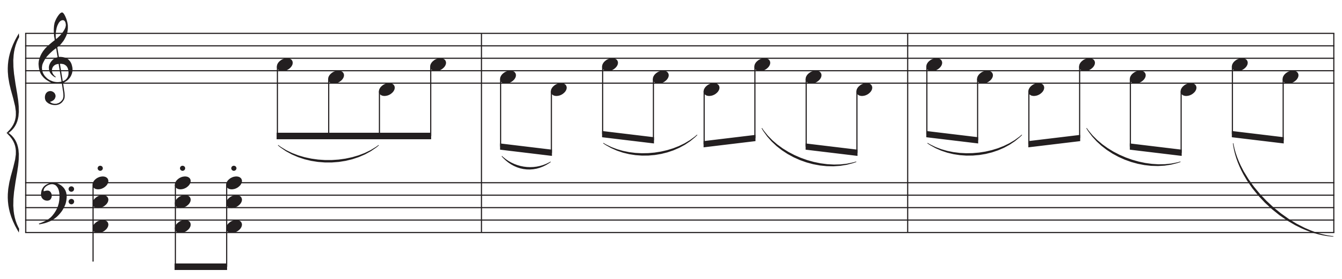


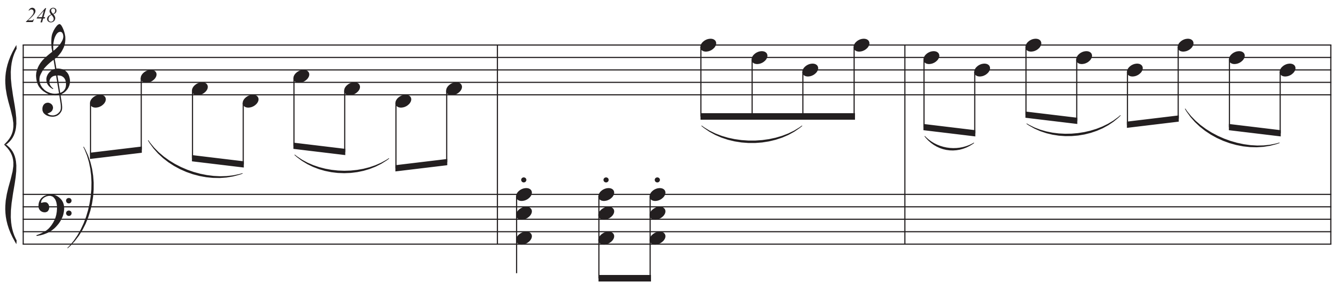
W

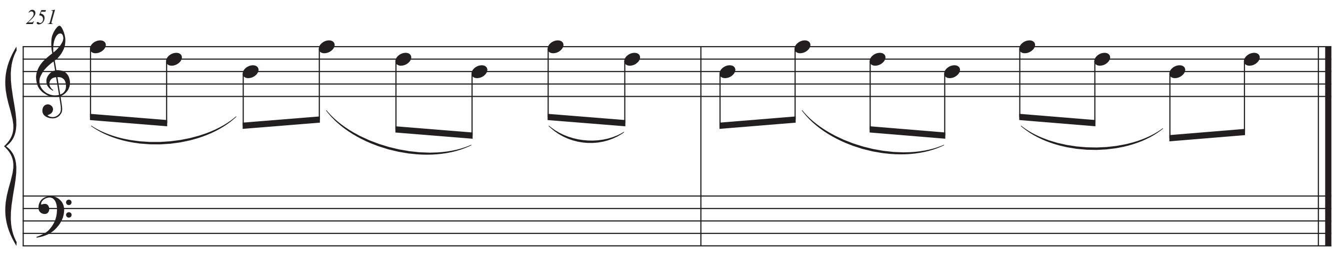




X







BIBLIOGRAPHY

Abraham, Frederick David Ed, and Albert R Gilgen. *Chaos theory in psychology.* Praeger Publishers/Greenwood Publishing Group, 1995.

Bell, Shawn, and Liane Gabora. "A music-generating system inspired by the science of complex adaptive systems." *arXiv preprint arXiv:1610.02475*, 2016.

Berdahl, Edgar, Eric Sheffield, Andrew Pfalz, and A Marasco. "Widening the Razor-Thin Edge of Chaos Into a Musical Highway: Connecting Chaotic Maps to Digital Waveguides." In *Proceedings of the International Conference on New Interfaces for Musical Expression, Blacksburg, Virginia, USA*, 390–393. 2018.

Boyland, Philip L. "Bifurcations of circle maps: Arnol'd tongues, bistability and rotation intervals." *Communications in Mathematical Physics* 106, no. 3 (1986): 353–381.

Cadoz, Claude, Annie Luciani, and Jean Loup Florens. "CORDIS-ANIMA: a Modeling and simulation system for sound and image synthesis: the general formalism." *Computer music journal* 17, no. 1 (1993): 19–29.

Cartwright, Timothy J. "Planning and chaos theory." *Journal of the American Planning Association* 57, no. 1 (1991): 44–56.

Choi, Insook. "Interactive exploration of a chaotic oscillator for generating musical signals in real-time concert performance." *Journal of the Franklin Institute* 331, no. 6 (1994): 785–818.

Cupellini, Enrico, Costantino Rizzuti, Eleonora Bilotta, Pietro Pantano, Michael Wozniewski, and Jeremy R Cooperstock. "Exploring Musical mappings and Generating Accompaniment with Chaotic Systems." In *ICMC.* 2008.

De Oliveira, Maurício C, Jacques Bernussou, and José C Geromel. "A new discrete-time robust stability condition." *Systems & control letters* 37, no. 4 (1999): 261–265.

Djoharian, Pirouz. "Generating Models for Modal Synthesis." *Computer Music Journal* 17, no. 1 (1993): 57–65. ISSN: 01489267, 15315169. http://www.jstor.org/stable/3680570.

Fels, Sidney S, and Geoffrey E Hinton. "Glove-talk: A neural network interface between a data-glove and a speech synthesizer." *IEEE transactions on Neural Networks* 4, no. 1 (1993): 2–8.

Fiebrink, Rebecca, and Perry R Cook. “The Wekinator: a system for real-time, interactive machine learning in music.” In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht).* 2010.

Garnett, Guy E, and Camille Goudeseune. “Performance Factors in Control of High-Dimensional Space.” In *ICMC.* 1999.

Hindmarsh, James L, and RM Rose. “A model of neuronal bursting using three coupled first order differential equations.” *Proceedings of the Royal society of London. Series B. Biological sciences* 221, no. 1222 (1984): 87–102.

Hunt, Andy, and Marcelo M Wanderley. “MHunt, Andy, and Marcelo M. Wanderley. ”Mapping performer parameters to synthesis engines.” Organised sound 7, no. 2 (2002): 97-108.” *Organised Sound* 7, no. 2 (2002): 97–108.

Hunt, Andy, Marcelo M Wanderley, and Matthew Paradis. “The importance of parameter mapping in electronic instrument design.” *Journal of New Music Research* 32, no. 4 (2003): 429–440.

Jensen, M Høgh, Per Bak, and Tomas Bohr. “Complete devil’s staircase, fractal dimension, and universality of mode-locking structure in the circle map.” *Physical review letters* 50, no. 21 (1983): 1637.

Kapur, Ajay, Georg Essl, Philip Davidson, and Perry R Cook. “The electronic tabla controller.” *Journal of New Music Research* 32, no. 4 (2003): 351–359.

Karplus, Kevin, and Alex Strong. “Digital synthesis of plucked-string and drum timbres.” *Computer Music Journal* 7, no. 2 (1983): 43–55.

Klatt, Dennis H. “Software for a cascade/parallel formant synthesizer.” *the Journal of the Acoustical Society of America* 67, no. 3 (1980): 971–995.

Lee, Matthew, and David Wessel. “Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms. ICMC, San Jose.” In *Proceedings of the International Computer Music Conference*, 277–277. International Computer Music Association., 1992.

Levy, David. “Chaos theory and strategy: Theory, application, and managerial implications.” *Strategic management journal* 15, no. S2 (1994): 167–178.

Lorenz, Edward N. “Deterministic nonperiodic flow.” *Journal of the atmospheric sciences* 20, no. 2 (1963): 130–141.

Majumdar, Mukul. “Chaotic Dynamical Systems: An Introduction.” *Economic Theory* 4, no. 5 (1994): 641–648. ISSN: 09382259, 14320479. http://www.jstor.org/stable/25054795.

Matsumoto, Takashi. “A chaotic attractor from Chua’s circuit.” *IEEE Transactions on Circuits and Systems* 31, no. 12 (1984): 1055–1058.

“MaxMSP.” Accessed September 1, 2019. https://cycling74.com.

Miranda, Eduardo Reck, and Marcelo M Wanderley. *New digital musical instruments: Control and interaction beyond the keyboard.* Vol. 21. Middleton, Wisconsin: AR Editions, Inc., 2006.

Morrison, Joseph Derek, and Jean-Marie Adrien. “Mosaic: A framework for modal synthesis.” *Computer Music Journal* 17, no. 1 (1993): 45–56.

Papetti, Stefano, Marco Civolani, and Federico Fontana. “Rhythm’n’Shoes: a Wearable Foot Tapping Interface with Audio-Tactile Feedback.” In *NIME*, 473–476. 2011.

Pickover, Clifford A. *The pattern book: Fractals, art, and nature.* Singapore: World Scientific, 1995.

“Pure Data.” Accessed September 1, 2019. https://puredata.info.

Rizzuti, Costantino. “Mapping chaotic dynamical systems into timbre evolution.” In *In Proceedings of Sound and Music Computinc Conference (SMC)*, 22–29. 2007.

Rodet, Xavier. “Nonlinear oscillations in sustained musical instruments: Models and control.” *Proc. Euromech, Hamburg, Germany*, 1993.

Sastry, Shankar. *Nonlinear systems: analysis, stability, and control.* Vol. 10. Springer Science & Business Media, 2013.

Sheffield, Eric. “Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments,” (2019) *LSU Doctoral Dissertations.* 4918. https://digitalcommons.lsu.edu/gradschool$_{d}issertations$/4918.

Smith, Julius O. “Physical modeling synthesis update.” *Computer Music Journal* 20, no. 2 (1996): 44–56.

———. “Physical modeling using digital waveguides.” *Computer music journal* 16 JSTOR, no. 4 (1992): 74–91.

Soria, Edmar, and Roberto Morales-Manzanares. “Multidimensional sound spatialization by means of chaotic dynamical systems.” In *NIME*, 13:79–83. 2013.

Thompson, STUART H, and STEPHEN J Smith. “Depolarizing afterpotentials and burst production in molluscan pacemaker neurons.” *Journal of Neurophysiology* 39, no. 1 (1976): 153–161.

Tindale, Adam R, Ajay Kapur, George Tzanetakis, Peter Driessen, and Andrew Schloss. “A comparison of sensor strategies for capturing percussive gestures.” In *Proceedings of the 2005 conference on New Interfaces for Musical Expression*, 200–203. National University of Singapore, 2005.

Van Den Doel, Kees, and Dinesh K Pai. “Modal synthesis for vibrating objects.” *Audio Anectodes. AK Peter, Natick, MA*, 2003, 1–8.

Verfaille, Vincent. “Effets audionumériques adaptatifs: théorie, mise en œuvre et usage en création musicale numérique.” PhD diss., Université de la Méditerranée-Aix-Marseille II, 2003.

Verfaille, Vincent, Marcelo M Wanderley, and Philippe Depalle. “Mapping strategies for gestural and adaptive control of digital audio effects.” *Journal of New Music Research* 35, no. 1 (2006): 71–93.

Yadegari, Shahrokh. “Chaotic signal synthesis with real-time control: solving differential equations in PD, MAX/MSP, and JMAX.” In *Proceedings of the 6th International Conference on Digital Audio Effects.* 2003.

Zölzer, Udo. *DAFX: Digital Audio Effects.* John Wiley & Sons, 2011.

VITA

Landon Viator is a musician, composer, and programmer from New Iberia, Louisiana planning to graduate in May 2020 with a PhD in Experimental Music and Digital Media from Louisiana State University. He received his Master of Music degree and Bachelor of Music degree from the University of Louisiana at Lafayette where he focused on music theory, composition, and music technology.

Landon grew up listening to rock and metal music and is currently inspired by progressive metal music. He plays percussion and guitar and is always striving to imbue musical characteristics from progressive metal into all of the music he composes, whether it's electronic or classical.

Landon has presented robotic percussion that plays the music of John Cage at NSEME along with Tony Morasco, Eric Sheffield, and Brian Elizondo. He has also given numerous graduate seminars on such topics as creating audio plugins, sound design in video games, mixing and mastering, and writing conference papers with Latex.