FINDING MUSIC IN CHAOS: THE PROCESS OF DESIGNING AND COMPOSING WITH VIRTUAL INSTRUMENTS INSPIRED BY PHYSICAL MODELING AND CHAOTIC SYSTEMS

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 Viator
B.M., University of Louisiana at Lafayette, 2015
M.M., University of Louisiana at Lafayette, 2017
April 2020

ACKNOWLEDGEMENTS

Acknowledgements go here.

CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT vi

INTRODUCTION 1

CHAPTER.1
A BRIEF OVERVIEW OF MAPPING 2
1.1. Introduction 2

CHAPTER.2
A BRIEF OVERVIEW OF NON-LINEAR DYNAMICAL SYSTEMS: CHAOS THEORY 7
2.1. Introduction 7
2.2. Circle Map 9
2.3. Hindmarsh-Rose Model 11
2.4. Conclusion 12

CHAPTER.3
A BRIEF OVERVIEW OF PHYSICAL MODELING 13
3.1. Introduction 13
3.2. Digital Waveguide Synthesis 14
3.3. Modal Synthesis 16
3.4. Conclusion 17

CHAPTER.4
CHAOS, PHYSICAL MODELING, AND GENETICS IN MUSIC: A LITERATURE REVIEW 18
4.1. Mapping Audio Synthesis Parameters 18
4.2. Chaotic Systems In Music 22
4.3. Physical Modeling 28
4.4. Creative Research 34

CHAPTER.5
AN OVERVIEW OF THE HEXAPAD CONTROLLER 35
5.1. Introduction 35
5.2. Construction 36
5.3. Interaction Methods 37
5.4. Instrument Type 38
5.5. Extracting Parameters 39

CHAPTER.6
X TRACTUUM 41
6.1. Introduction 41
6.2. Movement One 42
6.3. Software Realization 43

CONCLUSION 51

FUTURE WORK 52

APPENDIX.A
TITLE OF APPENDIX IN ALL CAPS 53

BIBLIOGRAPHY 55

VITA 58

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 blow up that then require re-instantiating 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 driver of the synthesis engine. To do this, one can create a physically-based synthesis model and map chaotic equations/behaviors to specific parameters of the synthesis model.

The goal of this project is to design physically-inspired virtual instruments based on conceptual instrument models that utilize chaos theory in the synthesis engine to explore novel sounds in a reliable, repeatable way for other composers and performers to use. This project presents a three-movement composition utilizing these concepts and a modular set of virtual instruments, that can be used by anyone, interacted with by an electronic music controller called the Hexapad controller.

INTRODUCTION

To understand the concepts used in this project, this document presents background information summarizing the following concepts used in creating the virtual instruments for the presented composition:

- Mapping - The process of connecting sound-producing sources with an interface for the user to interact with in a range that fits the need of the interaction.
- Chaos Theory - A field of mathematics that models and calculates certain kinds of 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, or percussion instruments.

This document then presents a literature review which summarizes compositions and projects that have preceded the current project in the computer music paradigm whose techniques and processes were instrumental in leading the way and making possible the techniques and processes utilized in the current project.

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

- The conceptual instrument model that the virtual instruments are designed around.
- An overview of the Hexapad controller and how it interfaces with the virtual instruments.
- How 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 MAPPING

1.1. Introduction

The focus of this project is to use chaotic dynamical systems, physical modeling, and genetic programming to design virtual instruments, implement them through software, analyze, and refine the models to compose avant-garde electronic music. There are many examples of music composed with or audio synthesis techniques using chaotic dynamical systems, physical modeling, and genetic programming, but this review will analyze the literature that parallels the criteria of the current dissertation project. The criteria are:

- Mapping audio synthesis parameters
- Chaotic systems in music
- Using physically-inspired modeling to create virtual instruments

When designing virtual instruments and physical models there are many parameters that affect the total audio output of the system as well as internal sections of the audio path. If this thought is applied to an acoustic instrument there are parallels that can be drawn between them. A violin has many parameters that mirror an audio synthesis model. There is volume parameter which is controlled by the pressure of the bow where as it would be a knob in a synthesis model. There is a timbre parameter controlled by bowing at different positions on the strings and by using different parts of the bow to make contact with the strings. In a synthesis model, this would be interpreted as a filter control, perhaps a set of filter controls for different types of filters and parameters for each. Then, there is all of the non-traditional prepared playing and extended techniques which can also be re-created in synthesis models in any way that achieves a similar sound to the acoustic parallel. All of these controls make up the parameters for the synthesis/physical model being created.

What is Mapping?

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.[1] The interface in which the user interacts with the virtual instrument is a separate mechanism from the source of the sound production. For example, an oscillator produces sound waves, but the user interacts with the oscillator with a separate device such as a slider or a number box. Mapping is the process of connecting the sound-producing source with the interface for the user to interact with in a range that fits the need of the interaction. An example of mapping in MaxMSP is shown below in Figure 1.1 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.

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:[2]

- Procedural processes such as neural networks
- Mapping strategies that are user-defined explicitly.

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 user retain direct control of the parameter mapping.

---

[1] 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.

[2] 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.

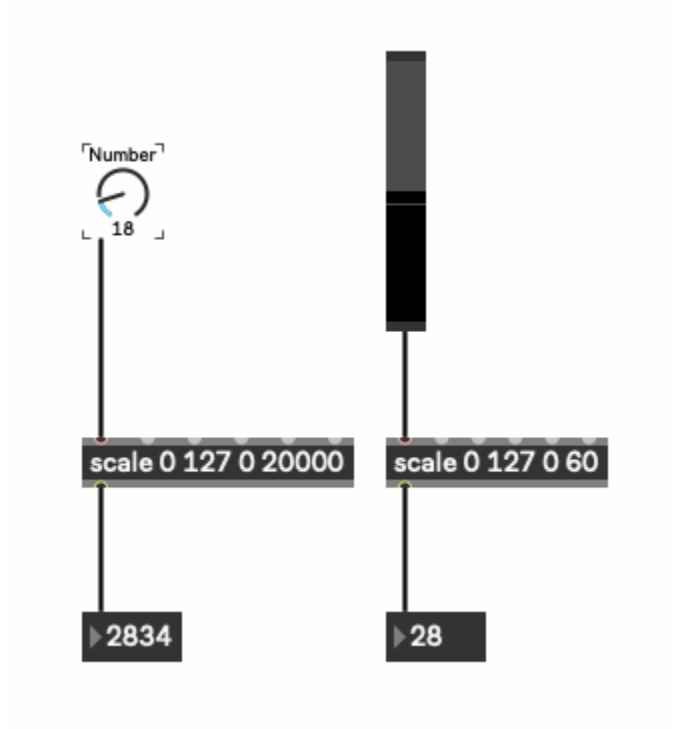

Figure 1.1. Simple example of mapping in MaxMSP.

Procedural Mapping

The procedure, in the case of the neural network, is powered by mathematical algorithms that analyze data and use that data to "learn" how to proceed forward in decision making without a human user explicitly 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, using the best fit to produce a new population of data including 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 synthesis by analyzing data from a MIDI keyboard controller and using that data to teach the neural network

to dynamically control timbre parameters for a synthesis model; this is an example of a procedural mapping process.[3]

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 user. This type of adaptive mapping in neural networks is based on the following three features:[4]

- 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 features of neural networks in mind, the authors created an interface in the shape of a glove that is worn by the user to map hand gestures to a speech synthesis model called the Glove-TalkII. The goal of the glove is to allow the user to control a granular synthesis model that plays back speech files according to the gestures made by the glove. The neural network is trained by the user's interpretation of controlling the speech synthesis model.[5]

1.1.1. Explicit Mapping

Explicit mapping is defined as the mapping method where a user retains direct control of the parameters that control a synthesis model. Hunt suggests that many user-controlled virtual instruments are explicitly mapped, specifically considered to be a few-to-many relationship.[6]

---

[3]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.

[4]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.

[5]Ibid.

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

This is a type of mapping that is defined as a method of mapping a few user-controlled parameters to control large amounts of parameters inside a given synthesis model. An example of this would be a large array of physically-modeled strings inside of a synthesis model that all pass through one or a few filters which the user has access to control. When considering this type of relationship for any two groups of parameters, any synthesis model has the three following possible parameter relationships:[7]

- One-to-one
- Many-to-one
- One-to-many

The one-to-one relationship is defined as a mapping where one user parameter controls one synthesis model parameter at any time. The many-to-one relationship is defined as a mapping where many user parameters control one synthesis model parameter at any time. Finally, the one-to-many relationship is defined as a mapping where one user parameter controls many synthesis model parameters at any time which is the most common form found in the literature on parameter mapping.

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

CHAPTER.2
A BRIEF OVERVIEW OF NON-LINEAR DYNAMICAL SYSTEMS: CHAOS THEORY

2.1. Introduction

Chaos theory has been studied extensively for over a century and applied to many fields of study such as economics,[1] mathematics,[2] and music[3] to name a few. Chaos theory is a field of mathematics that models and calculates certain kinds of nonlinear dynamical systems. A system is defined as dynamical if the systems 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 change.

Chaotic behavior in nonlinear dynamical system are more easily visualized when used in contrast with the mapping of a sinusoidal oscillator in (x, y) space because the differences are visually apparent. Figure 2.2 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 2.3 shows the mapping of the Peter DeJong chaotic map in (x, y) space calculated by Michael Hetrick.[4] The Peter DeJong chaotic map is specified by the following set of equations:

---

[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]Joseph Duemer, "Mathematics of Chaotic Systems," *New England Review (1990-)* 14, no. 1 (1991): 22–23, ISSN: 10531297, http://www.jstor.org/stable/40242401.

[3]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.

[4]Hetrick Michael, "DrawJong 2.0," 2011,

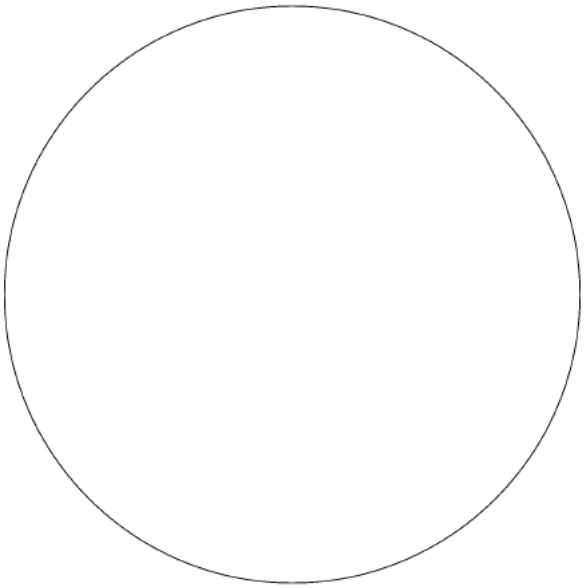

Figure 2.2. A representation of a sinusoidal oscillator state in (x, y) space.

$$x_n = sin(a * y_{n-1}) - cos(b * x_{n-1}) \tag{2.1}$$

$$y_n = sin(c * x_{n-1}) - cos(d * y_{n-1}) \tag{2.2}$$

The equations determine the dynamics of the system. In terms of (x, y) space, x and y represent points at any given point in time. As the equation iterates, each (x, 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. At certain sets of parameters it is possible for the behavior to be stable, but the edge of chaos can often be razor thin.[5] This razor thin edge can be difficult to control. The parameters can be any real numbers, which means the parameter space is infinite.

It is important to note that deterministic chaos is not random. The equations set rules and are deterministic according to the parameters passed. When applied to audio at the

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

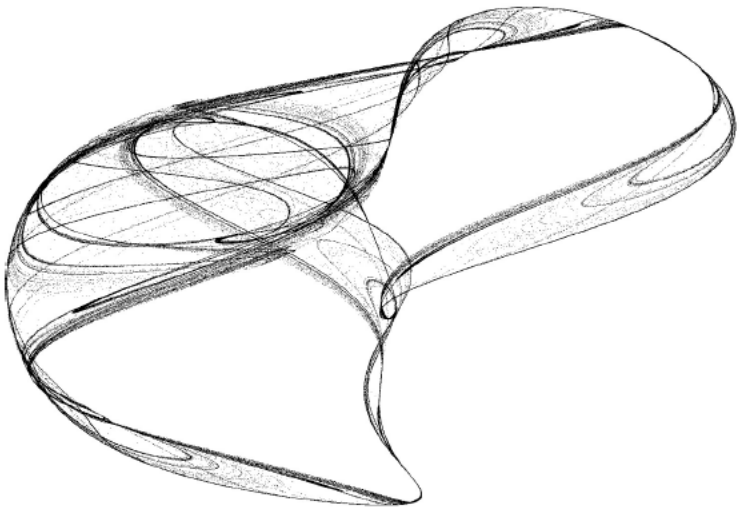

Figure 2.3. A representation of the Peter DeJong chaotic map in (x, y) space.

audio rate (44100 samples per second) or visual mapping, such as Figure 2.3, 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. Chaotic maps are also well-known to be extremely sensitive to the initial conditions of the equation. Chaotic maps do not seek out nearby points in (x, y) space and try to group themselves together, in fact the trajectories diverge over time if the initial conditions are different.

2.2. Circle Map

The Peter DeJong chaotic map is a two-dimensional chaotic map in (x, y) space. In contrast, the Circle map, which is a one-dimensional chaotic map, is specified by the following equation:[6]

$$\theta_{n+1} = (\theta_n + \Omega - (\kappa/2\pi)sin(2\pi\theta_n))\%1.0 \tag{2.3}$$

[6]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.

The Circle map contains a structure called an Arnold Tongue.[7] 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 the map is locked to its driving frequency, which is parameter Ω. 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.

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 quality of the equation. If $\kappa \neq 0.0$, Ω controls 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 which poses significant issues of using the map in practice, especially with audio. To use this map in audio, these intermediate κ values need to be avoided if the desired effect is continuous oscillation.

The Circle map is an example of a chaotic dynamical system that works on discrete time. Discrete time systems view parameters as separate, distinct moments of time.[8] An analogy that can help 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

---

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

[8]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.

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.

2.3. 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.[9] This model is an adaptation of a previously described model by Fitzhugh[10]. This model described 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; the cell does not fire indefinitely.[11] To produce the behavior of the cell, the introduction of a small electrical current is used. A slow, increasing current causes adaptations in the cell, so the Hindmarsh-Rose model adds an adaptation current to the original equation. The Hindmarsh-Rose model is described by the following differential equations[12] :

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

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

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

[9]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.

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

[11]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.

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

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 by the cell. Finally, I is a variable parameter that represents the applied current across the cell. At low levels $I < 1$, the model displays an individual burst followed by a decaying wave which slowly moves to the starting position. At higher levels of current $I < 2$ the model displays periodic bursts which lower in frequency as the model progresses.

Edge Of Chaos

There is a term associated with the exploration of chaotic systems in musical contexts called the "edge of chaos."[13] 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 multiple researchers, this edge is often incredibly small and navigating this area can be difficult.

2.4. Conclusion

Summarizing information about chaotic non-linear dynamical systems have been presented to understand the background behind about these types of systems. For the remainder of this document, chaotic non-linear dynamical systems will be referred to as chaotic systems for simplicity. Chaotic systems are used throughout the rest of this document and project as material for musical composition and audio synthesis, particularly for chaotic mapping of parameters and for chaotic oscillation.

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

CHAPTER.3
A BRIEF OVERVIEW OF PHYSICAL MODELING

3.1. Introduction

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[1]. 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.[2]

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.[3] 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, almost infinite control that the digital realm offers.

---

[1]Sheffield, Eric. "Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments." (2019), PhD Thesis, Louisiana State University, Baton Rouge, LA.

[2]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.

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

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 code to build the models or the visual patching code used in programs like MaxMSP[4] and Pure Data.[5] This document will display every discussed audio example in figures containing MaxMSP patches to allow easy implementation of the discussed audio processing.

3.2. 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.[6] 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 a simulation of this wave movement, the entire wave moves from one end to the other by one sample of space in each sample of time, hence the ability to use digital delay lines. The Karplus-Strong algorithm shown in Figure 3.5 is an example of using a digital delay line to simulate a wave traveling through a string.

The Karplus-Strong algorithm is a physical modeling technique first published by Kevin Karplus and Alex Strong in 1983 to model plucked strings and percussive timbres.[7] This method of physical modeling is an important predecessor to modern physical modeling techniques because it was a viable digital model that required low computing power, allowing it to be implemented effectively on earlier microprocessors before the advent of the powerful laptops that are so readily available now. This algorithm uses the averaging of two successive

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

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

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

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

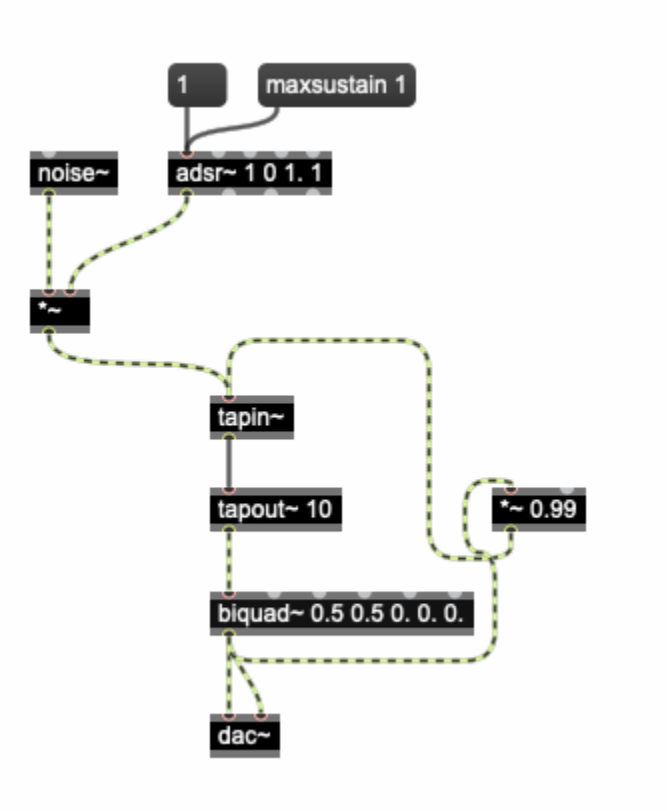

Figure 3.4. Implementation of the Karplus-Strong Algorithm in MaxMSP.

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.

The implementation in Figure 3.4 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 object 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 instrument. The now filtered signal is fed back into the delay line multiplied by 0.99 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. Waveguide techniques to model single-reed instruments have also been described by using bidirectional delay lines.[8]

3.3. Modal Synthesis

When solid objects are struck, like percussion instruments, there is a prorogation 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.[9] 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 excited an object, such as striking, scraping, and plucking. All of these types of stimulus 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.[10] Figure 3.5 shows a conceptual modal model for a resonating metal box with four resonance frequencies.

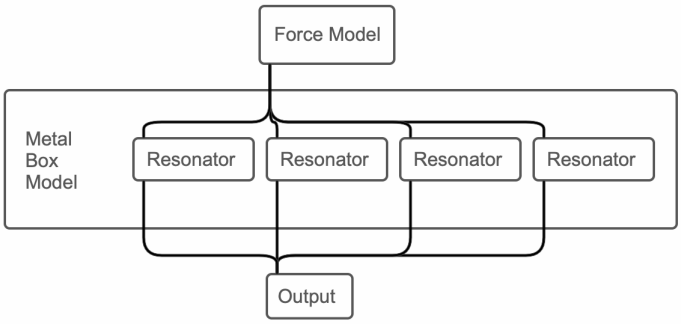

Figure 3.5. Conceptual figure of a modal model of a resonating metal box.

[8]Julius O Smith, *Efficient simulation of the reed-bore and bow-string mechanisms* (Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 1986).

[9]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.

[10]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, however a solid striking object is an appropriate example. The object strikes the metal box exciting each resonator which propagates vibrations throughout the metal box. All of the resulting waves are summed together to the output.

3.4. Conclusion

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.

CHAPTER.4
CHAOS, PHYSICAL MODELING, AND GENETICS IN MUSIC: A LITERATURE REVIEW

4.1. Mapping Audio Synthesis Parameters

4.1.1. Digital Audio Effects Parameter Mapping

Verfaille et al suggests a mapping strategy that connects gestures to audio effect parameters.[1] 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.[2] The authors focus on two control categories for parameters:

- Gestural
- Adaptive

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 idea behind this is to extract the intention of the gesture and convert it into a meaningful control for the audio effect. Adaptive controls consist of using sound features as control parameters, such as auto-tune, 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 4.6.

[1]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.

[2]Udo Zölzer, *DAFX: digital audio effects* (John Wiley & Sons, 2011).

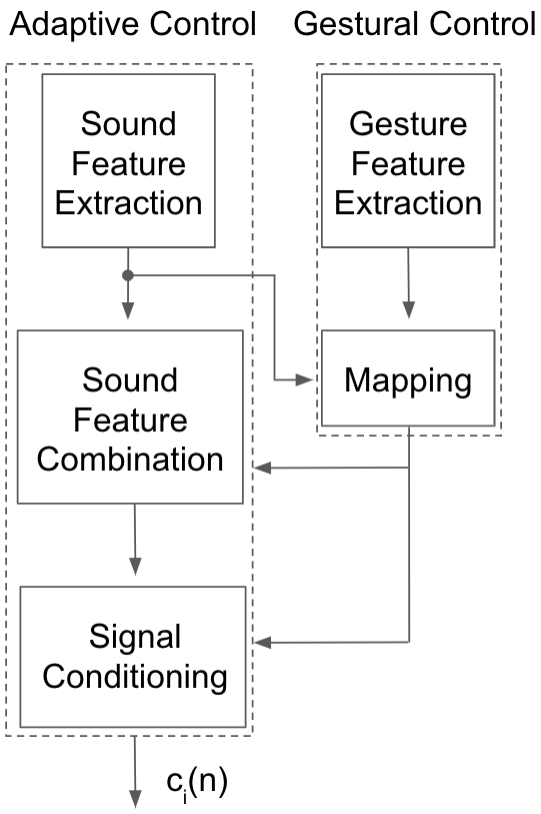

Figure 4.6. Diagram of the two-layer mapping technique connecting a sound feature to an effect control $c_i(n)$.

The authors present 6 examples of audio effect models which use this two-layer mapping strategy to control audio effects:

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 set is the fundamental frequency for the robotic voice output and the feature extraction is the RMS value of the user's voice.[3]

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

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 mean a finite number of control parameter, in this case 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.[4]

Adaptive Spatialization

In this model, a dancer controlled parameters of a 8-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 granular synthesis coupled with 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.[5]

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. To modify the intonation of a voice, the combination of pitch and loudness, one can apply pitch-shifting and gain alterations to a given fundamental frequency of the voice. Verfaille connects a mapping layer between these described sound features and a time-varying transformation. The gestural control is provided 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-

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

[5]Ibid.

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.[6]

Adaptive Equalizer

This is an equalizer with a time-varying equalization curve based on a vector as the feature. Gestural control is provided on the mean, min, and max 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 achieved with a slider. This adaptive equalizer was used as an instrument called the *Noisonic* in the piece *Flute Salad* by Verfaille.[7] This adaptive equalizer allows the user to modulate timbre, filter, and shape spectral envelopes and time amplitude envelope with the time interpolation control.[8]

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 adaptively control the depths and rates of the spectral tremoli. With this mapping, the user is able to control tremolo speed and the phasing aspect globally.[9]

---

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

[7]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).

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

[9]Ibid.

4.2. Chaotic Systems 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 even 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.

Real-Time Chua Oscillator

Chua's circuit is a thoroughly researched chaotic system.[10] Choi explored the musical qualities of the oscillation achieved by Chua's circuit.[11] Chua's circuit (which is an analog electrical circuit) is specified by the following three differential equations.

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

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

$$\frac{dv_3}{dt} = \frac{1}{L_2}[G(v_2 + R_0 i_3)$$

C_1 and C_2 are capacitors and L is an inductor, all that are passed a voltage current. v_1, v_2, and i_3 are three discrete signal paths represented by each equation. The non-linear character of Chua's circuit is found in the function of v_1, $\int(v_1)$ given by the following equation.

$$\int(v_1) = G_h v_1 + \frac{1}{2}(G_a - G_h)|v_1 + E| - |v_1 - E|$$

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

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

[11]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.

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; the piano range being 27.5hz to 4,186hz and the Chua circuit frequency range from 35hz to 15khz.[12]

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

4.2.1. 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.[13] 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 which 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 of the same name specified by the following set of differential equations.[14]

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

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

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

---

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

[13]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).

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

The variables b, r, and Pr are variable control parameters that accept any real numbers which affects the behavior of the oscillation based on the equation. Parameters X, Y, and Z are the unknown values that are solved for to generate 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.

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

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

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

Yadegari found that the implementation of these differential equations yielded oscillations that interpolated between being stable and being unstable in its phase, hence the "chaotic" keyword. 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.

4.2.2. Chaotic Mapping

Cupellini et al explored the mapping of chaotic systems to control parameters such as filter and spatial parameters for the synthesis models which utilized Chua's circuit for the chaotic model.[15] 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

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

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.

4.2.3. 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.[16] These trajectories are calculated using a chaotic system, specifically the logistic equation. The discrete version of the logistic equation is defined as:

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

The functional form is defined as:

$$\int(x) = rx(1 - x)$$

The whole panning orbit is the set:

$$O(x) = \{x_i : i = 1, n, x_i = \int^i (x)\}$$

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

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

derive multiple panning orbits from a singular evaluation of the above panning equation. To interpolate between these trajectories, by dividing the interval $[0, 1]$ in $r(i, n)$ pieces where $r(i, n)$ is the number of speakers in the ith array of the nth sound source.

4.2.4. Connecting Chaotic Systems To Digital Waveguides

Berdahl et al investigated multiple discrete-time chaotic systems to create new musical instruments.[17] The authors note the difficulty navigating the edge of chaos because it can be "razor-thin."[18] Because of this, the authors suggest connecting chaotic systems to digital waveguides to synthesis 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 authos 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:

$$V_n = \phi(V_{n-L}) \tag{4.4}$$

where V_{n-L} refers to an earlier step in time, ϕ is the chaotic system, and V_n is the final output. Figure 4.7 above demonstrates a chaotic system connected to a digital 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 system. Adjusting the parameters of the chaotic system ϕ allows the user to explore timbrel space in the chaotic system and find appropriate ways to map these parameters to user controls to create a musical instrument.

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

[18]Ibid.

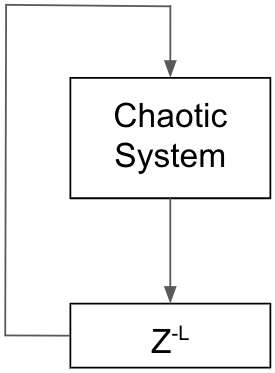

Figure 4.7. Block diagram of a chaotic system connected to a digital waveguide.

Limiting Instability

As mentioned earlier, chaotic systems tend to have a point where the system goes unstable. One way to avoid this is to bound[19] 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 tame the instability. These chaotic systems also have the possibility of getting stuck at a certain point where audio output is halted 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.

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

4.3. Physical Modeling

4.3.1. *MOSAIC*

MOSAIC is a physical modeling framework that facilitates modal modeling synthesis.[20] *MOSAIC* contains instruments based off of collections of resonant structures that interact with each other by vibration 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 such actions. To facilitate excitation and control data for these structures, *MOSAIC* contains *controller objects* which allows the user to interact with synthesis parameters. Finally, 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 to allow 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 functions written in c++ that implement the desired physical model by calling the method associated with that model. For example, Figure 4.8 above shows a snippet of code that calls a method to create a string model.

---

[20] 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 4.8

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.

4.3.2. ETabla

The *ETabla* is a electronic controller inspired by Tabla drums created by Kapur et al. 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 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 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 a traditional North Indian classical song and a ritual song, which used an atmospheric sound-scape instead of a percussive setting.

4.3.3. *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.[21] 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 analog inputs of an Arduino Duemilanove Board 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 back 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 small transients of audio caused by rebounding vibrations. Each sensor's data is unpacked so that there are four individual streams of audio signal, one for each sensor.

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

The authors included their own sound synthesis engine to accompany the *Rhythm'n'Shoes* which uses the Sound Design Toolkit, 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.

4.3.4. *Resuscitation*

Resuscitation is a composition for laptop ensemble in which four players play a shared physically-modeled virtual instrument.[22] The model concept is four strings connected to the four corners of a resonant plate connected at one point above the plate. Each player excites one of the strings by striking and plucking a piezo-based interface. Figure 4.8 below shows the mental model used to create the virtual instrument created in *Resuscitation*.

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

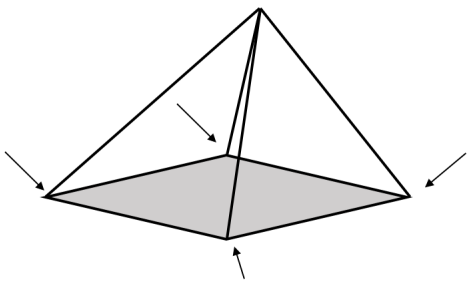

Figure 4.9. A representation of the mental model used to conceptualize the virtual instrument that is used in *Resuscitation*

[22]Eric Sheffield, "Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments," 2019,

balance of their dynamics just as a 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. Specific actions such as tapping, striking, and damping the plate and plucking the strings are notated with specific note heads and on different note spaces much like a multi-percussion piece is notated.

4.3.5. Conclusion

The concept of mapping requires close attention whether you are building custom virtual instruments or implementing parameters for standard audio effects in an audio signal path.

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, and that the edge of chaos can be explored be 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 an easy to use 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.

All of these concepts are explored in the current project and are built upon in some way in the design process of the virtual instruments.

4.4. Creative Research

CHAPTER.5
AN OVERVIEW OF THE HEXAPAD CONTROLLER

5.1. Introduction

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. The Hexapad can be connected to instrument or line-level audio inputs to send its audio signals to devices such as audio interfaces which is primarily how it is used in this project. Figure 5.11 below shows the Hexapad.

Figure 5.10. The Hexapad controller.

5.2. 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. Figure 5.12 below shows the simple circuit for each channel on the Hexapad. Each microphone is connected to an audio output jack with a resistor connected in parallel with the 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.

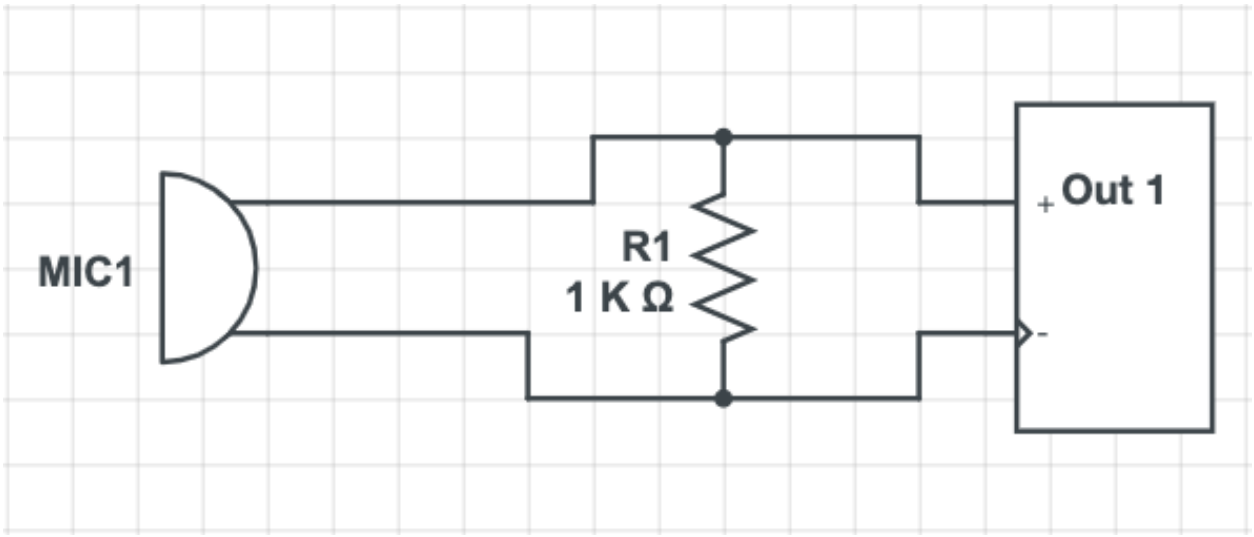

Figure 5.11. Basic audio effect configuration for the Hexapad.

5.3. Interaction Methods

5.3.1. 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 an 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, and in turn that speaker sounds the loudest, then the surrounding microphones are triggered by the propagating waves and the respective speakers sound. 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.

5.3.2. 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 laid out on the Hexapad. 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 combines 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.

5.4. 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 best example of instrument-like controllers is keyboard controllers such as MIDI controllers and keyboard synthesizers.

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. This way, unlike 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 because although inspired by percussion instruments, the Hexapad can be interacted with in any way and is not limited to what objects or gestures can excite the microphones.

---

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

5.5. 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 5.13 below shows this configuration for a single channel.

Figure 5.12. The basic circuit for each channel on the Hexapad

Another method is to convert the audio signal into a MIDI message when it crosses a certain threshold to use it as a trigger for operations that use control messages such as sample playback, envelope triggering, toggling effects on or off, etc. Figure 5.13 on the following page shows the Hexapad signal converted into a trigger for control messages. The signal is passed into a thresh object which sends a signal of 1 when the input signal passes above the threshold of the object, in this case 0.5. The number box under the thresh object then converts the signal of 1 or 0 into a control integer of 1 or 0. The change object then filters out repetitions of a number so that the triggering only happens when the number changes. The number then toggles the envelope of a sine wave (left), drum loop sample playback (middle), and a distortion effect (right).

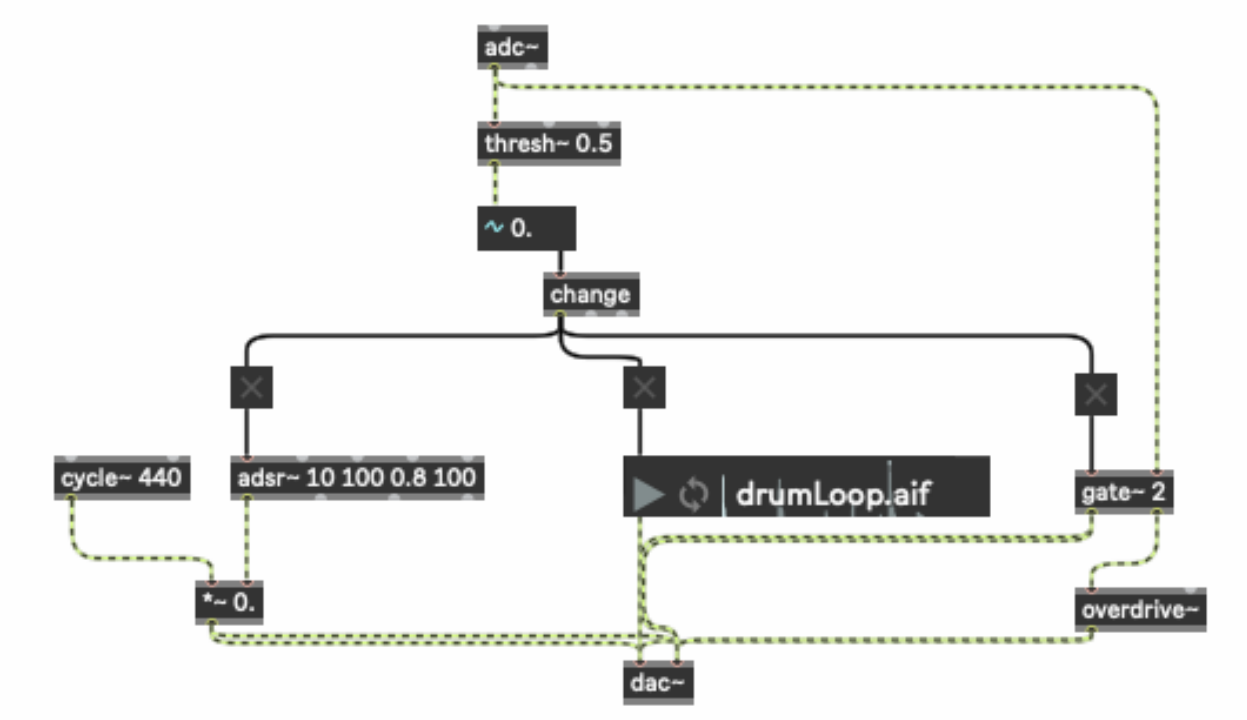

Figure 5.13. Converting the Hexapad signal into a trigger for control messages.

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, 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 5.12 to achieve interesting musical sounds.

CHAPTER.6
X TRACTUUM

6.1. Introduction

VIII Tractuum is a three-movement composition for Hexapad, guitar, and fixed-media. 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.

VIII Tractuum is Latin for *Eight Pathways* which represents the fact that the Hexapad is used in an eight-channel arrangement for the composition. Therefore, 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.

Movement 1 of *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 utilized in movement 1 focusing 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. In this way, 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.

Movement 2 is for guitar and fixed-media. This movement explores using chaotic sounds and synthesis in a more popular style of music, progressive metal, such as the artists *Meshuggah* and *Animals as Leaders*, where tonality and voice-leading practices are more traditional than in the current paradigm of experimental music.

Lastly, movement 3 focuses on the Hexapad again, but makes use of more creative accompaniment elements rather than soloistic performing such as pre-composed samples, loops, and MIDI playback to fit the Hexapad into a more collaborative and electronic ensemble performance practice.

6.2. Movement One

6.2.1. 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.

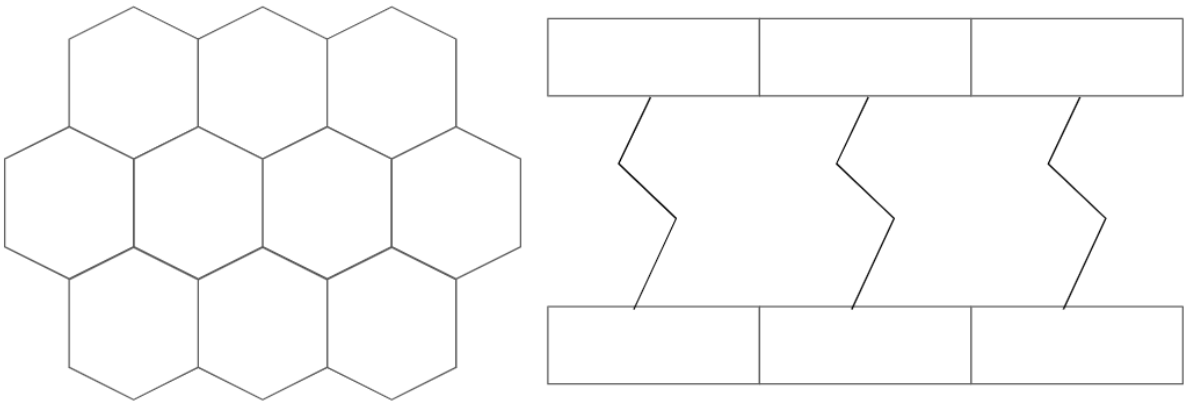

Figure 6.14. The Hexapad conceptual instrument.

The left side of Figure 6.14 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 6.14 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 string, more specifically a chaotic digital waveguide. Exciting the top plates triggers the chaotic strings which in turn trigger the bottom plates which 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]

[1]Sheffield, "Designing and Composing for Interdependent Collaborative Performance with Physics-Based Virtual Instruments."

6.3. Software Realization

Realizing the synthesis engine for the previously described virtual instrument began by realizing each individual part as a synthesis module. These modules are the resonant plate, the chaotic string, and the bottom plate. Figure 6.15 below shows the flow diagram that describes the synthesis engine for the Hexapad virtual instrument.

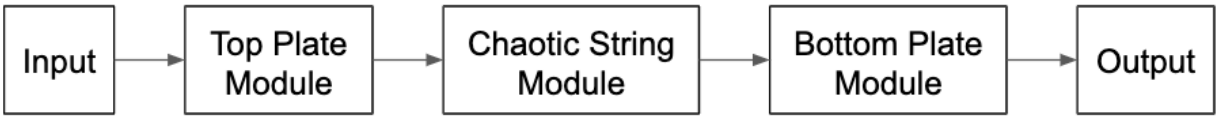

Figure 6.15. Basic flow of the synthesis engine for the Hexapad virtual instrument.

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.

6.3.1. 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]

$$y(nT) = Ax(nT) + By(nT - T) + Cy(nT - 2T) \tag{6.5}$$

$$c = -exp(-2 * \pi * bw * T) \tag{6.6}$$

$$b = 2exp(-\pi * bw * t)cos((2 * \pi) * F * T) \tag{6.7}$$

$$a = 1 - b - c \tag{6.8}$$

[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.

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 related to the frequency and bandwidth of a resonator. Listing 6.1 below shows the implementation of the resonant filter in the gen environment codebox object.

Listing 6.1. 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 = 2 * exp(-pi * 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 vale is used to initialize it since it will immediately get overwritten when audio begins processing, so it was initialized with nothing 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 which has been named x. In2 is the fundamental frequency at which the resonator resonates at which has been named f and in3 is the bandwidth of the resonator. Equation 6.5 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 needed to pass the integer manually. In this case, the sampling rate is 44,100 Hz. Equations 6.6 - 6.8 are then implemented directly using the previously named variables. The next equation in Listing 6.1 is the implementation of equation 6.5. Remembering that y(nT) is the output sample and x(nT) is the input sample, this equation can be understood more intuitively as:

$$output = input * A + B * previousSampleOne + C * previousSampleTwo$$

Then, 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 spiced into any part of the synthesis engine. Figure 6.16 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.

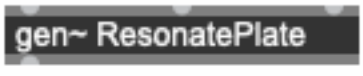

Figure 6.16. Resonant plate gen object.

6.3.2. 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 6.17.

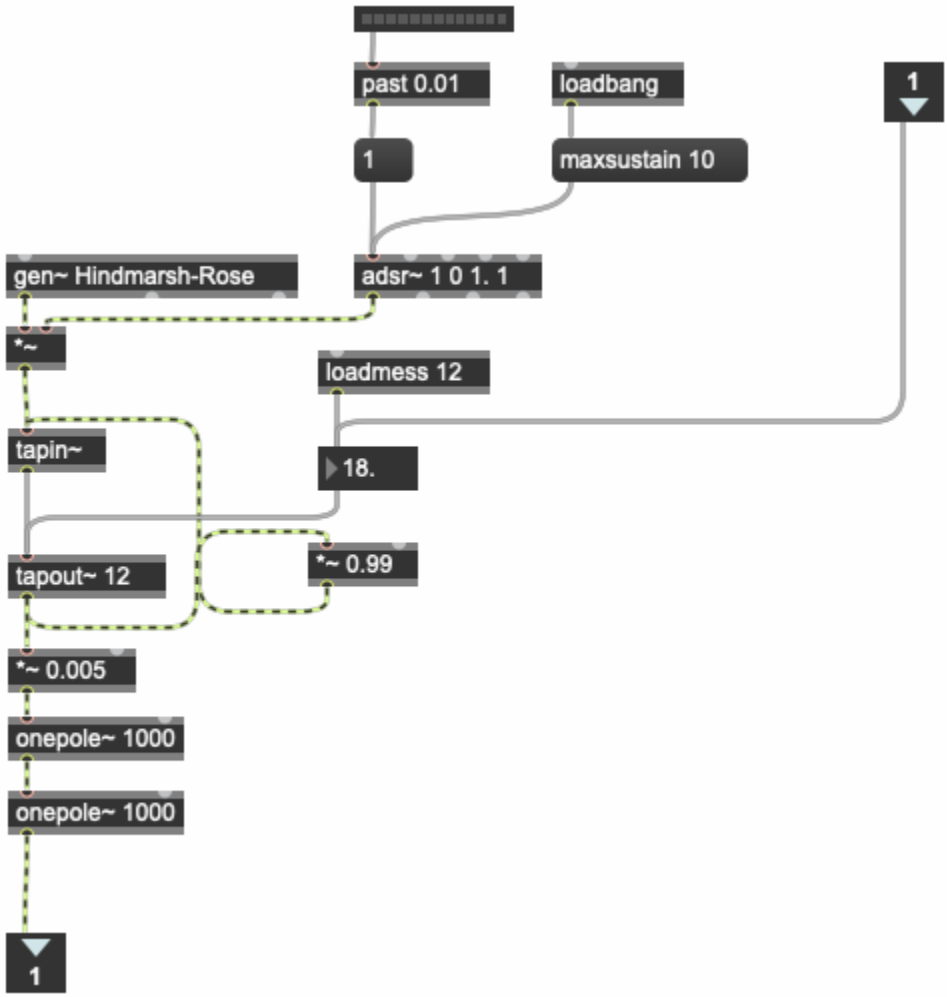

Figure 6.17. Chaotic string module.

At the top of Figure 6.17, above the 'past' object, there is an audio meter that measures the peak level of the audio signal. It outputs the value of the audio in the form of floating point numbers and the past object sends out a bang when the audio signal is above 0.01. This is the trigger that excites the digital waveguide. In this digital waveguide, the usual noise object that was previously discussed in Chapter 3 has been replaced with a chaotic oscillator

as the noise source for the digital waveguide to insert a chaotic system in a stable way. Because it's being used as a short burst of audio with the envelope object, the oscillator does not go unstable or break. This chaotic oscillator implements the Hindmarsh-Rose chaotic equations previously discussed in Chapter 2. The Hindmarsh-Rose model is described by the following differential equations:

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

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

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

Just like the top plate algorithms in the previous section, the Hindmarsh-Rose chaotic equations were implemented in the gen environment in MaxMSP. Listing 6.2 on the following page shows the code for the gen codebox object.

Listing 6.2. Gen implementation of the Hindmarsh-Rose chaotic equations.

```
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 * (x * x * x) + b * (x * x) + i;
dy = c - d * (x * x) - y;
dz = r * (s * (x - x) - 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 oscillator has three equations to compute, for 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 and interesting and stable oscillation was discovered. The newValue variable allows the oscillator to self oscillate, other wise audio would be required to be sent into it to trigger oscillation. The variable itself is not necessary, just the value of one would be sufficient.

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 has 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.

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 6.17 above shows the gen object for the Hindmarsh-Rose oscillator which has one inlet that can receive audio signals or Param values to change the coefficients in the equations. It has three outlets, one for x, y, and z.

6.3.3. 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, ethereal quality and completes the mental model discussed above in Figure 6.15.

6.3.4. Chaotic Drone

Movement One uses one more chaotic sound engine which utilizes the Circle Map map discussed in Chapter 2. The Circle Map is defined by the following equation:

$$\theta_{n+1} = (\theta_n + \Omega - (\kappa/2\pi)sin(2\pi\theta_n))\%1.0 \tag{6.9}$$

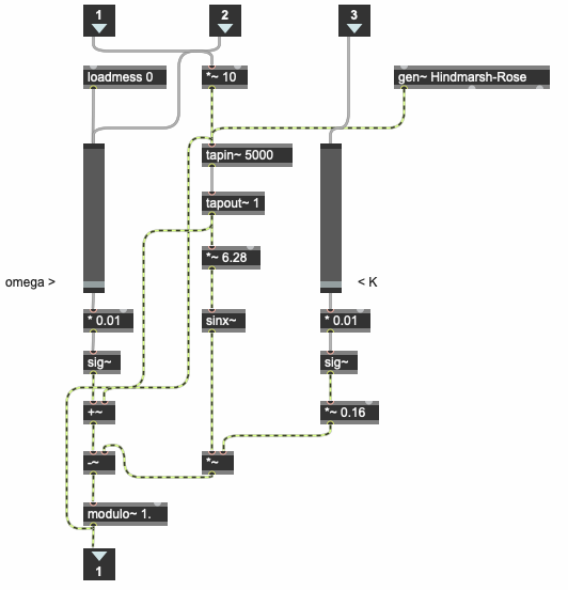

Figure 6.18. MaxMSP implementation of the Circle Map.

CONCLUSION

Conclusion goes here.

FUTURE WORK

APPENDIX.A
TITLE OF APPENDIX IN ALL CAPS

Appendix A goes here. An example PDF follows as appendices are usually useful for including supplemental documents.

EMDM Dissertation Latex Template
Example PDF

Replace as needed with your own PDFs (e.g. musical scores, IRB forms, etc.)

BIBLIOGRAPHY

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.

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.

Duemer, Joseph. "Mathematics of Chaotic Systems." *New England Review (1990-)* 14, no. 1 (1991): 22–23. ISSN: 10531297. http://www.jstor.org/stable/40242401.

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.

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.

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.

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.

Michael, Hetrick. "DrawJong 2.0," 2011.

Miranda, Eduardo Reck, and Marcelo M Wanderley. *New digital musical instruments: control and interaction beyond the keyboard*. Vol. 21. 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.

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

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.

Smith, Julius O. *Efficient simulation of the reed-bore and bow-string mechanisms*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 1986.

———. "Physical modeling synthesis update." *Computer Music Journal* 20, no. 2 (1996): 44–56.

———. "Physical modeling using digital waveguides." *Computer music journal* 16, 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

Vita goes here.