IMPROVISING IN THE LOOP: SEVEN IMPROVISATORY, LOOP-BASED PIECES FOR CUSTOMIZED DATA-DRIVEN INSTRUMENTS

by

NATHAN MORRIS ASMAN

A DISSERTATION

Presented to the School of Music and Dance
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Musical Arts in Music Performance: in Data-driven Instruments

June 2020

DISSERTATION APPROVAL PAGE

Student: Nathan Morris Asman

Title: Improvising in the Loop: Seven Improvisatory, Loop-Based Pieces for Customized Data-driven Instruments.

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Musical Arts in Data-driven Instruments degree in the School of Music and Dance by:

Jeffrey Stolet	Advisor and Chair
Akiko Hatakeyama	Core Member
Jon P. Bellona	Core Member
Colin Ives	Core Member

and

Sara D. Hodges	Interim Vice Provost and Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded June 2020.

DISSERTATION ABSTRACT

Nathan Morris Asman

Doctor of Musical Arts in Data-driven Instruments

School of Music and Dance

June 2020

Title: Improvising in the Loop: Seven Improvisatory, Loop-Based Pieces for Customized Data-driven Instruments.

Improvising in the Loop serves as both a compendium of and comprehensive guide to the music I have written and performed as well as the instruments I have created and designed during my tenure as a doctoral student at the University of Oregon's School of Music and Dance. Loop-based music has long played a fundamental and influential role in the genesis of the music that I write and perform, as has the idea of musical improvisation. Every piece that I have included in my Digital Portfolio Dissertation contains improvised loop-based music in some way. The first of seven pieces that I will be discussing is called *Étude No.1, for Curve.* It is the piece I wrote for my second custom-built interface, called Curve. The étude was written for Ableton Live and Max/MSP. *T(Re)es (Trees: Remixed)* was written for the Monome. The software I used to write *T(Re)es* was Ableton Live and Max/MSP. *Trio 1-465* is next and is the first ensemble-based piece in my repertoire. *Trio* was written for cello, turntables, and my own custom-designed interface utilizing Contour Design's RollerMouse Red Plus. *Trio* employs Ableton Live, Max/MSP, and the Ms. Pinky software environments. *Impromptu No. 1, for ShrutiBox+* was written for a custom-built hybrid instrument/interface of my own design called Shrutibox+. Ableton Live and Max/

MSP served as the compositional and performative software environments for the impromptu. *Multios* is a piece I wrote using only my iPad and iPhone, utilizing a variety of iOS apps for the sound design, data mapping, composition, and ultimate performance of the piece. *ArcMorph* was written for my Sensel Morph and Arc, and was realized and performed with Ableton Live and Max/MSP. The final piece I will discuss is called *Kaurios*, which is also the name of the custom-built interface. Composed and performed with Ableton Live and Max/MSP, *Kaurios* signifies the culmination of my studies as a doctoral student at the University of Oregon.

SUPPLEMENTAL FILES

These supplemental materials consist of digital videos of each piece discussed in this Digital Portfolio Dissertation and the custom-made software (Max/MSP patches, Ableton Live sets, and iOS files) associated with each piece.

VIDEO FILE I: Performance of *Étude No.1, for Curve*

VIDEO FILE II: Performance of *T(Re)es (Trees: Remixed)*

VIDEO FILE III: Performance of *Trio 1-465*

VIDEO FILE IV: Performance of *Impromptu No. 1, for ShrutiBox+*

VIDEO FILE V: Performance of *Multios*

VIDEO FILE VI: Performance of *ArcMorph*

VIDEO FILE VII: Performance of *Kaurios*

SOFTWARE FILES I: *Étude No.1, for Curve*

SOFTWARE FILES II: *T(Re)es (Trees: Remixed)*

SOFTWARE FILES III: *Trio 1-465*

SOFTWARE FILES IV: *Impromptu No. 1, for ShrutiBox+*

SOFTWARE FILES V: *Multios*

SOFTWARE FILES VI: *ArcMorph*

SOFTWARE FILES VII: *Kaurios*

CURRICULUM VITAE

NAME OF AUTHOR: Nathan Morris Asman

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, Oregon
Denison University, Granville, Ohio

DEGREES AWARDED:

Doctor of Musical Arts, Data-driven Instruments, 2020, University of Oregon
Master of Music, Intermedia Music Technology 2013, University of Oregon
Bachelor of Arts, Music History, 2008, Denison University

AREAS OF SPECIAL INTEREST:

Electronic Instrument Building and Design
Electroacoustic Composition and Performance
Digital Arts
Film Scoring

PROFESSIONAL EXPERIENCE:

Instructor, Continuing Education, Lane Community College, Eugene, OR, 2018
Assistant Instructor, SPARK Academy, Lane Community College, Eugene, OR, 2017-2018
Graduate Teaching Fellow, University of Oregon, Eugene, OR, 2014-2016
Instructor, CASCADE Initiative, University of Oregon, Eugene, OR, 2015-2016

GRANTS, AWARDS, AND HONORS:

Academic Travel Award, University of Oregon, 2015, 2017, 2018, 2019
Graduate Teaching Fellowship, Music Technology, University of Oregon 2014-2016
One of ’25 Ducks Who Will Change The World,’ University of Oregon, 2013
Presidential Scholar, Denison University, 2004-2008

CONFERENCE PRESENTATIONS:

Guthman Musical Instrument Competition, 2017, 2018, 2019
New Interfaces for Musical Expression (NIME), 2018
Society for Electro-Acoustic Music in the United States (SEAMUS), 2015, 2018
VU Symposium, 2016
International Computer Music Conference (ICMC), 2015

ACKNOWLEDGMENTS

It would be impossible for me to list each and every person that has helped me throughout my academic career, but of special significance to me is the help and guidance of Dr. Jeffrey Stolet, Dr. Akiko Hatakeyama, Dr. Jon P. Bellona, Dr. Chet Udell, Instructor John Park, Professor Colin Ives, Dr. Jeremy Schropp, Dr. Loren Kajikawa, and Dr. Jeremy Grimshaw. Additionally, I would never have gotten this far without the help, friendship, and insight from Zachary Boyt and Connor Sullivan. Finally, family has and will always be the most important thing in my life, and without my parents, David and Anne Asman, my little brother, Aaron Asman, and my wife, Lindsey Asman, I simply would not be the musician, artist, or person that I am today.

I — quite literally — cannot thank all of you enough. That being said, I suppose I'll start here. So, thank you.

This dissertation, and the work discussed therein, is dedicated to my Grandmother, Carol Asman. Her unwavering support in any and all of my artistic endeavors will always be with me, and has helped shape the musician and artist that I have become. I will forever miss her terribly, but she will always live on within my music and creative work.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Overview

This document, *Improvising in the Loop: Seven Improvisatory, Loop-Based Pieces for Customized Data-driven Instruments*, is part of my Digital Portfolio Dissertation (DPD). It is a guide, compendium, and explanation of the music that I have written and the data-driven instruments that I have designed and built as a doctoral student at the University of Oregon. The purpose of this text is to illuminate the ideas, themes, influences, and concepts that suffuse my work on both a micro and macro level. In addition to the explanations and walkthroughs of each of my pieces, understanding what happens behind the scenes is equally necessary to the overall comprehension of my work. So before I focus on my body of work, I will discuss in Chapter II the conceptual background and theoretical frameworks of not just my seven pieces, but of data-driven instruments as a whole. After elucidating those concepts and frameworks, I will focus on my body of work on a piece-by-piece basis in Chapter III.

The Seven Compositions

The first piece I discuss in Chapter III, *Étude No.1, for Curve*, was written for my custom-built interface called Curve, and demonstrates my use of loops and musical improvisation. I used loops to form the structure of the *Étude No.1*, creating the entire musical foundation of the piece. Improvisation also played a key role in a how I structured the piece in real time. Each performance of the piece yielded different results

based upon how I had played and recorded each loop in the first section. Ableton Live served as the sound synthesis engine and Max/MSP as the data processor for the étude.

The second composition in my DPD is called *T(Re)es (Trees: Remixed)*, and was written for an interface called the Monome.[1] The Monome is the interface with which I am most familiar, and was my choice for *T(Re)es* because it was also my main performance interface in my band. The musical material that I used in this piece is comprised of very small, short, looped samples that were taken from my original composition, *Trees* (2009). I used Ableton Live as the sound synthesis engine and Max/MSP as the data processor for *T(Re)es*, strengthening the connection between the original version of *Trees* and the new version due to the fact that I utilized and then built upon the same software that I used for the original composition. It was important to me to maintain a connection to the original piece because I wanted the new version to reflect and convey my musical activities outside of the academy, bringing my academic and non-academic music together during performance.

Trio 1-465 is my first and only ensemble-based work discussed in my this text. I wrote *Trio* for cello, turntables, a custom vinyl control software for the turntables called Ms. Pinky,[2] and a data-driven instrument formed by Contour Design's RollerMouse Red Plus,[3] Ableton Live, and Max/MSP. *Trio 1-465* showcases improvised loops by using them as sectional bookends that form the beginning and end of multiple sections

---

[1] "Monome," accessed February 22, 2020, https://monome.org/.

[2] "The Interdimensional Wrecked System (IWS) — Ms Pinky's Playhouse," accessed February 10, 2020, https://mspinky.com/about/.

[3] "RollerMouse Red Series," *Contour Design Inc.*, n.d., accessed April 11, 2020, https://www.contourdesign.com/product/rollermouse-red/.

throughout the piece. In addition to acoustic sound that is made by the cello, Ableton Live served as the sound synthesis engine for the piece.

Impromptu No. 1, for ShrutiBox+ is the fourth piece in my portfolio. Written for an electronically-enhanced instrument of my own devising called ShrutiBox+, the impromptu explored the original, natural sound of an Indian reed- and bellows-based instrument called a shruti box through the use of improvised loops that are recorded in real time and then modulated through the use of custom-built electronics that I designed to augment and enhance the shruti box. The impromptu was composed and performed with Ableton Live and Max/MSP for sound synthesis and data mapping, respectively. Additionally, my shruti box became the original sound source for the last three pieces of my DPD, as that was the overarching theme to my final doctoral recital, given in October of 2018.

The fifth piece in my DPD is called *Multios*, and was written exclusively for iOS on my iPad Pro and iPhone X+. I did not use a computer at any point in the composition or performance of the piece. *Multios* instead utilized a variety of iOS applications that I chose and chained together to form the compositional and performative software environments of the piece. Sonically, the sounds I used for *Multios* derived originally from my shruti box, which I then modified and shaped into the final soundscape for the piece using an iOS app called iDensity.[4] In performance, I improvise loops of the sounds coming from iDensity in real time with the help of another iOS app called Loopy,[5] which

[4] "IDensity," *Electronic Music Software*, accessed February 10, 2020, https://www.apesoft.it/idensity/.

[5] "Loopy, the Live Looper App for IPhone and IPad," accessed April 4, 2020, https://loopyapp.com/.

compositionally forms the backbone of *Multios*. Additionally, iDensity served as the source of sound generation during the performance of the piece, with various data streams generated in the app TC-Data[6] mapped to various musical parameters within iDensity. When paired and routed through the other iOS apps Audiobus 3[7] (Audiobus Remote on my iPhone), and AUM,[8] I was able to tap into the potential of my iPad Pro as not only a performance interface, but a full-fledged data-driven instrument and compositional environment.

The sixth piece of my DPD discussed in Chapter III is called *ArcMorph*, and was written for the Sensel Morph[9] and the Arc.[10] Both the Morph and the Arc are highly adaptable interfaces which granted me the freedom to customize and adapt the interfaces to my specific needs for *ArcMorph*. The Arc consists of two endless encoders, each with its own set of sixty-four LED lights that can be used to visualize the position of each dial. The Morph completed my performance interface by affording me the button-type control that I needed for the realization of the piece. Ableton Live was the sound synthesis engine for this piece, and Max/MSP was my data mapping software environment. The original sounds for *ArcMorph* came from my shruti box. Again utilizing improvised loops created in real time, I was able to delve into the sonic world hidden in the shruti box samples and

---

[6] "TC-Data - Bit Shape," accessed February 10, 2020, http://www.bitshapesoftware.com/instruments/tc-data/.

[7] "Audiobus: Live, App-to-App Audio.," accessed April 4, 2020, https://audiob.us/.

[8] "AUM - Audio Mixer," *Kymatica.Com*, last modified February 4, 2020, accessed February 10, 2020, http://kymatica.com/apps/aum.html.

[9] "The Sensel Morph," *Sensel*, accessed February 22, 2020, https://sensel.com/pages/the-sensel-morph.

[10] "Arc," *Docs*, accessed February 22, 2020, https://monome.org/docs/arc/.

explore their nuances and minutiae in great detail with the Arc and Morph through the use of granular synthesis.

The seventh and final piece that I discuss in Chapter III is called *Kaurios*. Sharing its name with the interface itself, *Kaurios* signifies the culmination of my research and work in the realm of data-driven instrument design and building at the University of Oregon. Kaurios (the interface) is the first of my custom-built interfaces to feature wireless technology, utilizing the Bluetooth Low Energy (BLE) MIDI[11] communications protocol. Receiving the wireless MIDI data is Max/MSP, which then passed it along into Ableton Live for the sound generation and musical performance of the piece. The original sounds for the piece itself derived from my shruti box, and laid the foundation for my use of improvised loops and sonic manipulation throughout the piece. The functional versatility of Kaurios, from its buttons, distance sensor, and 9-degrees-of-freedom motion sensor to its portability and wireless capabilities, offered me a unique set of interactive techniques to explore during performance.

[11] "Bluetooth LE MIDI Specification," accessed April 11, 2020, https://www.midi.org/specifications/item/bluetooth-le-midi.

CHAPTER II

CONCEPTUAL BACKGROUND AND THEORETICAL FRAMEWORKS

Introduction

Comprehension of the overarching ideas, theories, components, and systems operating in my data-driven instruments (DDIs) and their respective compositions is crucial to the understanding of my work. DDIs can sometimes differ heavily from traditional acoustic instruments in terms of design, interaction, implementation, composition, and performance. Therefore, this chapter will explain the key concepts and theoretical frameworks about my DDIs and their use in my compositional processes and real time performances. Especially relevant and highlighted in this chapter will be: the definition of a data-driven instrument (data as a replacement for energy within the performance model, modularity, and mutability), hardware (observability in performance, components, design, and implementation), software design and use (data mapping and sound synthesis software, data mapping strategies), data types and communication protocols, sound synthesis, and composition (looping, improvisation, and musical influences).

What is a Data-driven Instrument?

To understand a DDI, we must first understand how a traditional acoustic instrument operates. Acoustic instruments all share the need for physical energy to produce their sound. “Whereas traditional [acoustic] instruments are driven by the energy

exerted into the instrument's physical systems, new instruments — data-driven instruments — replace energy's function with data streams. And the way we play these instruments is by generating data through performative actions" (Stolet 2013). In other words, where the physical energy exerted into a traditional acoustic instrument is what drives and ultimately produces sound, in a DDI the physical interaction with a performance interface generates data streams that are mapped to specific musical parameters to create or modulate an instrument's sound. For example,

> [r]ather than depressing keys of a piano or plucking strings of a violin to perform music, music is produced and controlled in real time by performative actions that create data streams that actuate musical events and expressively control musical parameters. Just as the amount of energy exerted into traditional acoustic instruments affect musical parameters like volume and timbre, different numerical values affect musical parameters contained in a synthesis algorithm - so a performer's task is to create data sequences that will produce aesthetically interesting or pleasing results when routed to control desired musical parameters (Stolet 2013).

All of my DDIs operate within this paradigm. As I physically interact and perform with the hardware interfaces of each DDI, I am dynamically generating a multitude of data streams that are then mapped to specifically chosen musical parameters within my sound synthesis engine, resulting in a real time musical performance with my chosen DDI.

There are three fundamental components that typically make up any given DDI: 1) a hardware interface that produces data, 2) a software layer dedicated to the mapping and routing of the data being generated by the hardware interface, and 3) a sound synthesis environment that responds to the data and creates the resultant sound (Bongers 2007, 11) (See Figure 1). This modular design is an essential feature of DDIs. The concept of modularity centers around the idea of building more complex things out of

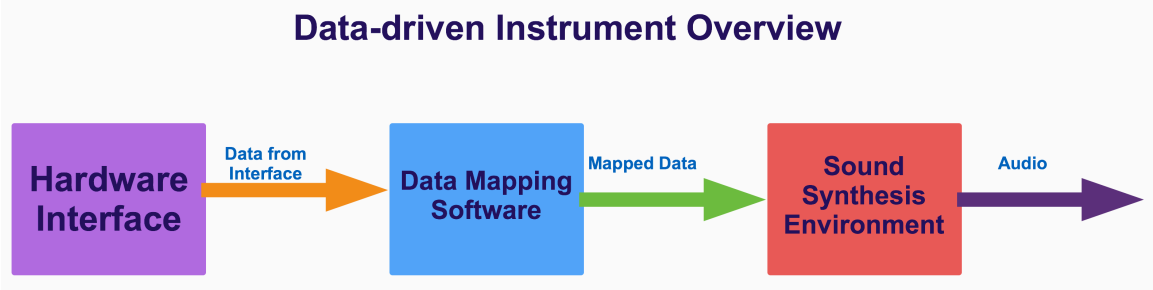


Figure 1: Basic overview of my data-driven instruments

existing simpler components that are connected, combined, and placed in association with one another. The modularity granted by the interface, data mapping, and sound-producing components and the myriad ways that they can each be realized lies at the core of what a DDI represents. This conceptual model grants me the freedom to design, develop, and create new DDIs, compositions, and performances. Furthermore, thanks to the variety of data mapping strategies and sound-synthesis algorithms available, I am able to employ the concept of *mutability* into my individual DDIs and real time performances. When something is mutable, it is "capable of change or of being changed" (Merriam-Webster, n.d.). For example, using a mute with a brass instrument is a simple demonstration of mutability in the acoustic domain. However, the concept of mutability is greatly expanded within DDIs. For instance, in my piece *Étude No. 1, for Curve*, I am able to change the sound that is produced by the sensors in real time, from one section to the next. During the first section of the piece, the sensors produce a particular sound, but in the fourth section those same sensors produce an entirely different sound. The exciting thing about DDIs is that an instrument can easily change many times, in real time during a performance, leading to a wealth of compositional and performative opportunities.

Hardware Performance Interfaces

One of the most important influences on the design and implementation of my performance interfaces is that of observability in performance.

> Some interfaces can be better observed in a performance context as compared to other interfaces. The problem is not with the interface itself, but rather with the audience not being able to observe what the performer is doing. […] Obscuring of the performative act creates a negative impact, because an audience has fewer cues to guide the listening experience and can develop little appreciation about the performance since **controlling actions** can't be seen. How to make the performative operation of an interface in a musical performance ***observable*** can be one of the most important considerations when structuring a musical performance (Stolet 2013).

The audience's observability of my interfaces has always played a crucial role in the design and implementation of each interface, and has guided both how I create my custom-designed interfaces and how I employ pre-built devices in my compositional processes and performances. Because the interfaces I use in each of my pieces have rarely, if ever (in the case of my custom-built DDIs), been observed in a musical context or performance before, the audience may require far more visual cues to fully understand and comprehend how the interface operates and ultimately generates music. My use of lights, lighted buttons, and video projections highlight the interactions that I execute during my performances, allowing the audience to better observe my performative actions so that my music can be more easily accessed and understood.

With regards to the hardware interfaces that I use in my DDIs, there are several components that come together to create the performance interfaces that I use for each of my pieces. First and foremost is the *sensor*. Each DDI that I have created or have used in my pieces features an array of different sensors that act as the focal point for my

performative interactions. A sensor is a device that measures and reports its detection of a physical property such as heat, light, sound, pressure, or acceleration (Merriam-Webster, n.d.). The sensors within my hardware interfaces measure certain physical properties that report their corresponding data streams that reflect those changes in real time based on my performative actions. Broadly speaking, there are two basic sensor categories that I use in the design and implementation of my DDIs: sensors that output single packets of data when instructed to do so — buttons — and sensors that output a continuum of data over time — faders. Buttons consist of two distinct states: pressed or not pressed. When pressed, the change of state sends out a single packet of values that corresponds to and controls some sort of musical result or outcome. In some cases, the button needs to be pressed and released *each time* a change of state is desired, resulting in a specific type of button called a toggle. In other cases, the button needs to be pressed only a single time, reporting a change of state while it is being pressed and/or actuated, and then reporting the opposite state as the button is released. These types of buttons are called momentary buttons or switches (see Figure 2). Buttons differ from faders in that they generally offer no type of ongoing real time control over a musical parameter. That is to say, since

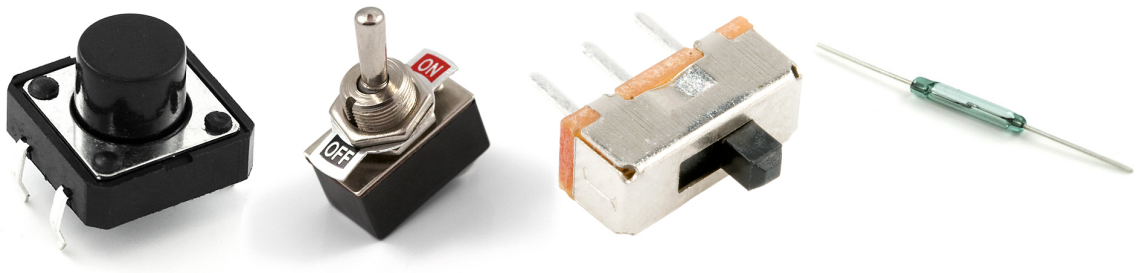


Figure 2: Examples of different types of buttons and switches (Sparkfun, n.d.)

buttons only consist of ‘pressed’ or ‘not pressed,’ there is no ongoing continuum of values being produced from a simple momentary button or toggle without continuous physical interaction by the performer.

Faders, on the other hand, “output what is best characterized as a ‘data stream’ that unfolds over time and whose values are defined by the fader’s […] position” (Stolet 2013). Additionally, faders can exist in more than one dimension. For instance, I can have a single-dimensional fader such as a potentiometer (a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider that can be read as an analog value (Electrical4u, n.d.)) that reports it’s data on a single axis only, i.e. as you push the knob of a fader up or down (see Figure 3). There are also multi-dimensional faders that are commonly used in my hardware interfaces, such as a two-dimensional roller that reports data about it’s position within an X/Y plane (see Figure 3), or a three-

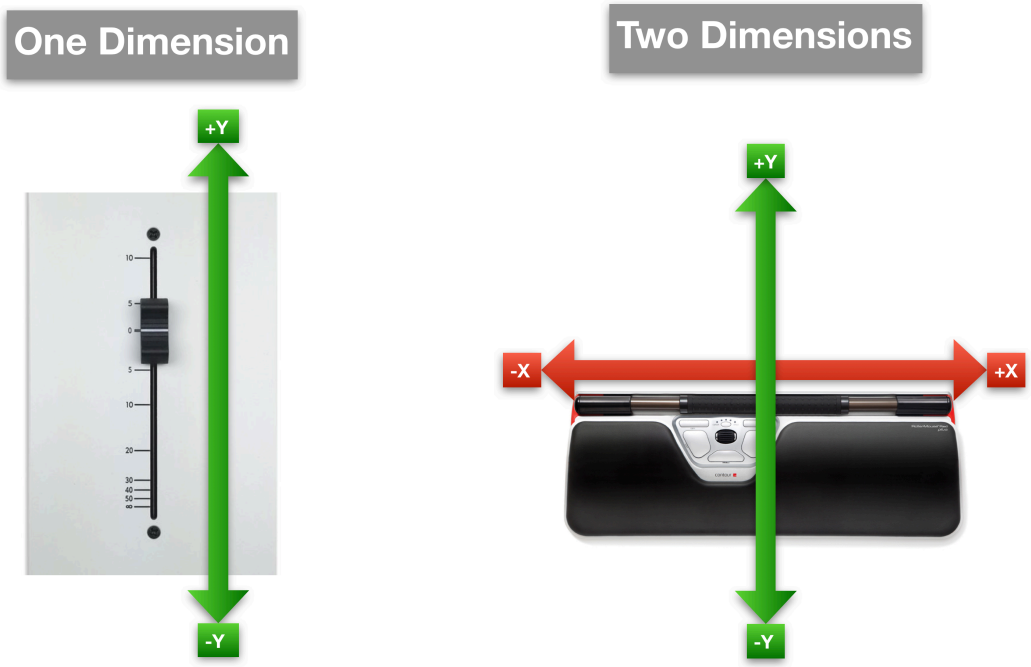


Figure 3: A single-dimensional fader (left) and a two-dimensional fader (right)

dimensional fader like an accelerometer that reports data on its position and acceleration through space on X-, Y-, and Z-axes simultaneously (see Figure 4). In order to maximize the level of control that I have over my music, each interface I use consists of both buttons and faders.

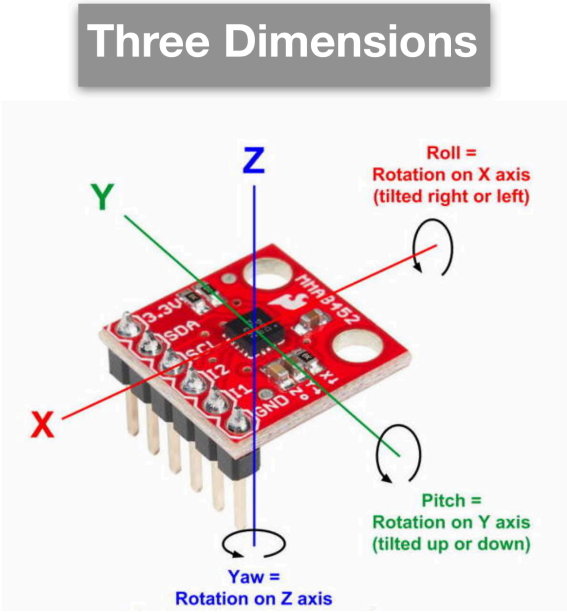


Figure 4: A three-dimensional fader (Code: Internet of Things, n.d.)

Several of the pieces I discuss in this document feature interfaces that I did not build and configure from unassembled sensors and components. For these pieces (as discussed in Chapter III.2: *T(Re)es (Trees: Remixed)*, III.3: *Trio 1-465*, III.5: *Multios*, and III.6: *ArcMorph*), I used a variety of pre-built electronic devices that I either repurposed to function as interfaces or customized to fit the specific needs of the piece that I was writing. These devices are the Monome (*T(Re)es*), the RollerMouse Red Plus (*Trio 1-465*), the iPad Pro and iPhone X+ (*Multios*), and the Sensel Morph and Arc (*ArcMorph*).

On the other hand, I also custom-built certain interfaces. In each of these cases, the sensors embedded within their respective interfaces were connected to microcontrollers through which data was sent to the computer. These interfaces include Curve (Chapter III.1: *Étude No.1, for Curve)*, the ShrutiBox+ (Chapter III.4: *Impromptu No. 1, for ShrutiBox+)*, and Kaurios (Chapter III.7: *Kaurios*). Within my custom-built interfaces, a microcontroller acts as the central processing unit (CPU). The Arduino Mega and Adafruit Bluefruit Feather M0 boards are the microcontrollers that I selected to use for my work. Arduino is an open-source electronics platform that consists of specific hardware components called boards and a corresponding software environment called the Arduino IDE (Integrated Development Environment) where the actual coding and programming of each Arduino or Adafruit board takes place (Arduino, n.d.). The board acts as the point of transduction, where typically analog inputs from the various sensors are converted into digital messages. Each sensor is connected to the board via wires and soldering, and the board then plugs into my computer via a Universal Serial Bus (USB) connection or transmits the data wirelessly via Bluetooth. The board then reads the input from each sensor and sends that data into my computer as serial data for most of my interfaces, which is then mapped, shaped, routed, and sent as MIDI (Musical Instrument Digital Interface) data into my sound synthesis environment within the same computer. The data then controls, generates, and modulates various musical parameters and allows me to finally perform my music in a real time performance environment.

Software

Two software environments, Ableton Live (Ableton) and Max/MSP (Max), form the foundation of my creative software workspace. Ableton serves as the sound synthesis environment, and Max functions as the data mapping/routing environment.[12] Ableton is a digital audio workstation (DAW) that I use to compose and perform music. All of the sounds that are heard in my pieces (excluding *Multios*) are created and/or processed within this software environment and output during each performance. Ableton features two main views; Arrangement View and Session View (see Figures 5 and 6 respectively). Arrangement View lets Ableton function as a timeline by horizontally displaying each track, enabling me to temporally structure my sounds and music on a horizontal plane, similar to most traditional DAWs. Session View rotates each track vertically, letting me instead group and work with each individual sound by the effects chain I applied to it.

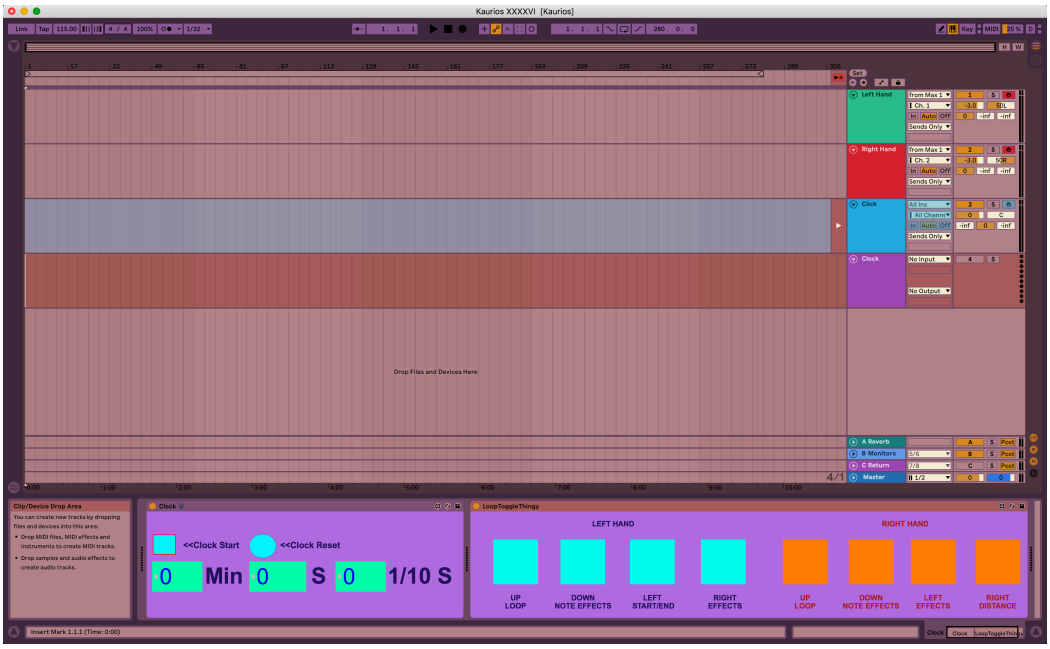


Figure 5: Ableton Live's Arrangement View

[12] Save for *Multios* and my use of iOS-based applications for that piece, which I will discuss in-depth in Chapter III.5.

Figure 6: Ableton Live's Session View

Both Views have been especially conducive to my particular work flow and creative style, allowing me to seamlessly switch between sound processing, compositional arrangement, and performance without ever leaving the software. Additionally, Ableton's MIDI mapping capabilities allowed me to compose and simultaneously shape the performance of each piece because I could assign my MIDI mappings as I think of them during the compositional process. This MIDI mapping practice has proven to be an extremely efficient tool in my creative arsenal.

Max is a visual programming language developed for musicians and multimedia artists that facilitates the mapping, shaping, and routing of data to and from various input and output devices (Cycling '74, n.d.). In my DDIs, Max operates as the bridge between my hardware interfaces and Ableton by taking the data streams generated by my hardware interfaces and shaping, mapping, and routing that data to the sound-producing

algorithms I have devised in Ableton. Each DDI features a specific program — called a patch — that I created within Max for that particular DDI. For instance, in *Impromptu No. 1, for ShrutiBox+,* my physical interactions with ShrutiBox+ generate data, which is sent via USB into my computer as serial data into Max. When that serial data is received by Max, it is transformed via a variety of different 'objects' that are virtually connected, turning the original data into the particular data streams needed for the performance of the piece. Those data streams are then routed from Max as MIDI data and sent into Ableton (See Figure 7).

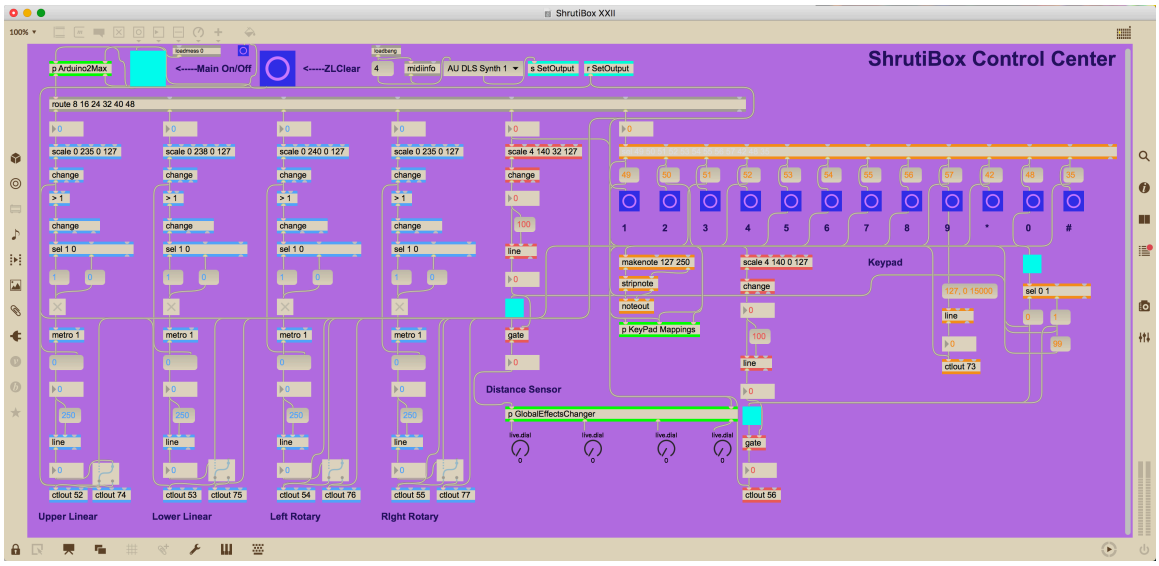


Figure 7: Example of a Max/MSP Patch

Data Mapping

Each Max object has a very specific function within the overall patch. As I chose and virtually connected those objects together, the data mapping system for that particular DDI was developed. Within the overall scheme of any of my given DDIs, data mapping functions as a kind of 'data orchestration.' In a traditional composition, orchestration

plays the role of mapping which melody is assigned to which instruments or voices. In one of my DDIs, data mapping functions as a technological orchestration, mapping and routing which data streams are assigned to which musical parameters. Whether mapping one data stream to one parameter, one data stream to many parameters, or many data streams to one parameter (Hunt and Wanderley 2002, 99), data mapping parallels the role and function of orchestration within a traditional composition.

Using these three broad data mapping paradigms as a starting point, the Max objects I utilized in each patch dictated how my data was mapped for that particular piece, which in turn dictated how the hardware interface for that piece controlled the sounds and music during performance. Max originates with around 1,000 discrete objects, not including the multitude of external objects that users create and disseminate themselves (some of which I use myself in my own patches). Given the sheer number of different combinations and data mapping schemes that can be derived from these objects, I have selected and will discuss some of my most-used operations to illustrate how data mapping works within my DDIs.

Data mapping refers to the set of operations that take place in the second stage of the data-driven instrument model. Data mapping in the real time performance of electronic music is primarily focused on the transformation of one set of data values into another set of data values. Data mapping allows me to exert my creative will and musical intent over the hardware interfaces I employ. I use data mapping to determine how and where the data being generated will control the musical performance. For instance, one of the data streams I generate from an accelerometer I use spans the numerical range of 0 -

1023. In order for that data stream to be usable in a particular context, I *scale* this data so that it fits within the native range of a MIDI data stream, 0 - 127 (see Figure 8). Now that the new range of the data stream fits within the bounds of a MIDI data stream, I can utilize the full range of the data during performance.

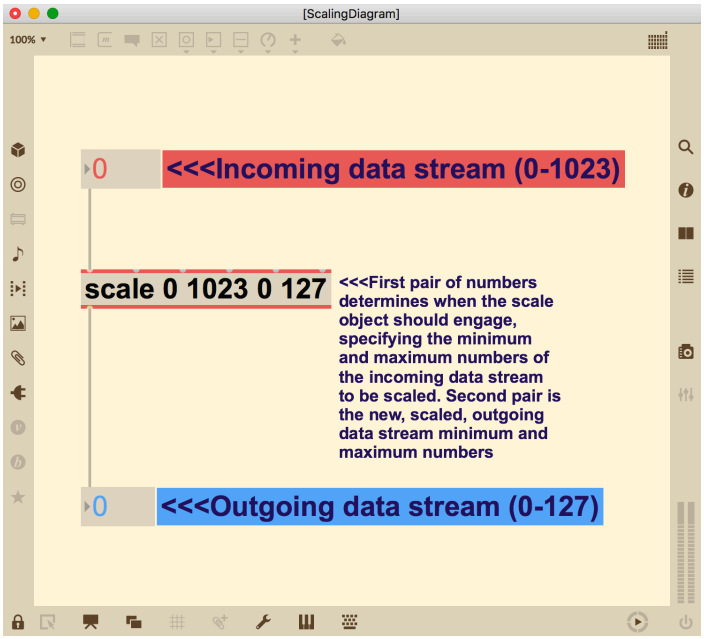


Figure 8: Basic scaling example in Max/MSP

Similar to scaling a data stream so that it spans a different range, *offsetting* a data stream is equally important to data mapping. Offsetting values of a data stream changes the minimum and maximum values of a range to different minimum and maximum values that are an equal distance apart. Typically the offsetting of data is accomplished through addition and subtraction. Sometimes an interface generates a data stream that is not entirely usable because of its range. For instance, a data stream I use occupies a range of 42 - 84, but I need the data to occupy the range of 0 - 42 instead. By applying simple subtraction to the original data stream (- 42 in this example), I can offset the data stream's

range, changing the minimum and maximum values of the data stream and yielding a range of data that is usable in performance.

Another common data mapping technique that I employ is *smoothing*. Data smoothing reduces the differences between contiguous values of data and can be accomplished through data interpolation or through simple averaging algorithms. Many times, when raw data is generated from a sensor and sent into Max, there is a certain amount of jitter that is inherent within the data stream, requiring smoothing of the data. I use the term 'jitter' to describe the unintended random variation in the data stream produced by performance interfaces. The random variation can be seen in the random fluctuations and jumps in the numbers themselves. I use data smoothing to help eliminate these numerical gaps and jumps caused by the jitter by using the 'line' object, which linearly interpolates data between one value and another over a specified amount of time (see Figure 9). Interpolation is "the computation of points or values between ones that are known or tabulated using the surrounding points or values" (Wolfram Mathworld, n.d.). I use data scaling, offsetting, and smoothing in each of my DDIs, and when combined with other Max objects and data mapping techniques, I am able to design, control, compose, and perform each of my pieces.

Data Types and Communications

The main data type and communications protocol that I utilized in my DDIs was MIDI data. MIDI is a standard that describes a communications protocol that facilitates communication between and among electronic musical instruments and computing devices who's essential purpose is to play and control music. In my work, at one stage or

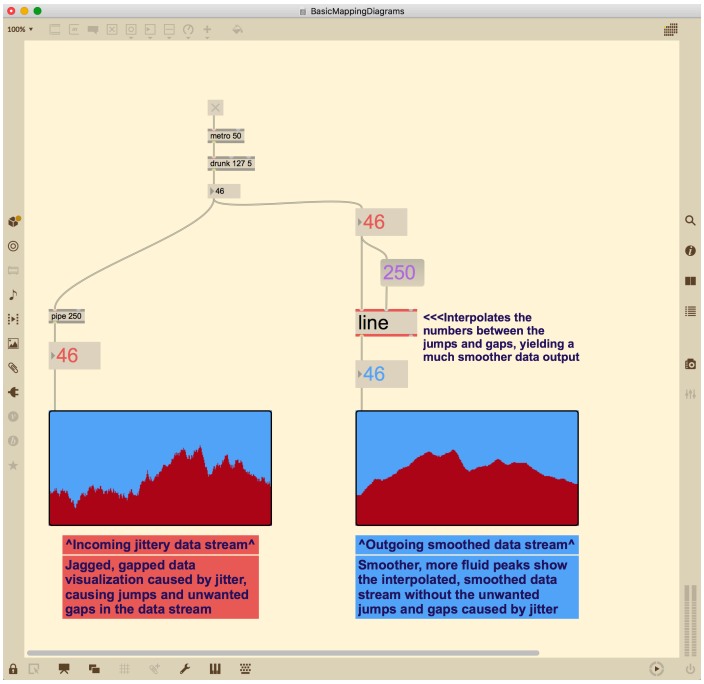


Figure 9: Basic smoothing example in Max/MSP

another, I convert data that is generated from my interfaces into MIDI data because of the many conveniences the protocol provides. Because MIDI is fully implemented in both Ableton and Max and is ubiquitous across most electronic music software, my workflow was streamlined and made efficient.

A second digital music communications protocol that I use is OSC (Open Sound Control). OSC was developed at the Center for New Music and Audio Technology at the University of California, Berkeley and is widely used by the electronic music community. While OSC is similar to MIDI, OSC is more flexible and sophisticated. Whereas MIDI's native resolution is 0 - 127,[13] OSC's resolution is much higher. "OSC allows one to transmit multiple data types commonly used on modern computers including 32-bit integers, floating point numbers, strings, and more" (OpenSoundControl 2004).

[13] While generic MIDI has a resolution of 0 - 127, high-resolution MIDI was later developed and has a much higher resolution than generic MIDI: 0 - 16,383.

Additionally, OSC's flexibility and sophistication can be seen through its non-fixed nomenclature and organizational scheme. MIDI data is typically sent on a specific MIDI channel between 1 and 16. OSC messages are not restricted to such a limited, set number of channels. Rather, OSC data streams are labeled and partitioned with their ranges determined by the user. For instance, if I was using three data streams that were sent from a single accelerometer yielding X-, Y-, and Z-axis data, I could organize and route each data stream by labeling the overall set of three data streams as '/accelerometer1,' and then labeling the individual X, Y, and Z data as '/X', '/Y', and '/Z'.

Sound Design and Synthesis Techniques

All of the sound synthesis within my seven pieces was done in Ableton — with the exception of *Multios*, Chapter III.5. Ableton features an extensive library of 'devices,' comprised of digital synthesizers, oscillators, sound generators, and effects, as well as third-party plugins and Max for Live devices that I used to create the sounds for my pieces. Additionally, Ableton's MIDI mapping capabilities help streamline my compositional process, allowing me to create my sounds and simultaneously organize them compositionally and performatively within the same software environment.

Ableton began integrating Max into its software in 2013, with full integration achieved in 2017. Because Max and Ableton are my most-used software environments, this integration has been especially helpful to me with respect to my sound design. In particular, I was able to leverage the number of granular synthesizers and effects for use during my sound processing. "Granular synthesis involves generating thousands of very short *sonic grains* to form larger acoustic events. The technique can be classified as a

form of *additive synthesis,* since sounds result from the additive combination of thousands of grains" (Roads 1988, 11). Granular synthesis opened up an entire world of sonic possibilities for me. I am able to create new sounds and sonic textures through granulating my initial sound materials, allowing me to develop, refine, and produce the music contained in my DPD. Exploring the nuances of individual sounds at the microscopic level through the use of granular synthesis has become one of my primary sound design, synthesis, and compositional techniques. Finally, granular synthesis mirrors my love and implementation of loops within my music, as each individual grain functions like a micro-loop, layering together and repeating to form an underlying sonic structure.

Another sound synthesis technique that I employ in each of my pieces is sample-based synthesis, or sampling. Sampling is a synthesis technique that employs the use of either hardware or software samplers, creating a digital recording of a particular instrument or sound (Roads 1996, 117). Whether particular sounds were individual pitches or longer melodic samples, each recorded audio file functioned as a building block of the overall soundscape that I created for each of my pieces.

Composition

There are two overarching ideas that permeate the music of my DPD: loops and improvisation. These two concepts play a vital role in both the compositional and performative processes of my music. The idea of a loop hinges upon the concept of repetition. While the precedent for the concept of loops or looping can be found in pre-electronic music in the forms of ostinatos and chaconnes, looping in electronic music

arises from the early practice of tape loops. In this practice, a short segment of recorded sound was played back and repeated to create rhythmic patterns. An extension of this idea is the repetition of a short recorded motive or melodic phrase to create a similar effect. These short sections of recorded sound or music can be repeated continuously and layered together to create rich, complex rhythmic and melodic patterns and variations. Loops can be created using a wide array of techniques including, but not necessarily limited to, turntables, synthesizers, sequencers, loop pedals, delays, samplers, and computer music software such as Ableton and Max. From utilizing pre-made loops that I have constructed beforehand to building loops in real time during performance, loops can be heard in every piece within my DPD. In these seven works, I use loops in three ways: first, loops lend form and structure to my pieces. Second, loops yield fantastic rhythmic and melodic complexities because they can be layered, shaped, and amalgamated together. Third and finally, because loops and repetition are used so prolifically throughout much of the traditional popular music that influences me, the use of loops in my pieces accentuates my love and tendency towards a more rhythmically traditional style of musical composition, performance, and expression.

The second concept that is central to all of my work is improvisation. In this context, improvisation can be defined as "the simultaneous invention and sonic realization of music" (Nettl and Russell 1998, 10). Each piece in my DPD incorporates improvisation in its performance. From improvising the actual creation of each loop as I perform in real time (highlighted in *ShrutiBox+, Multios, ArcMorph, Kaurios,* and *Trio 1-465*), to improvising the structure that my pre-constructed loops coalesce into

(highlighted in *T(Re)es* and *Étude No.1, for Curve*), improvisation helped me maintain a sense of freshness and originality within my music and performances at every iteration. The added level of musical instinct and creativity that becomes necessary when I have to be completely present in the moment of performance is what makes my music enjoyable, interesting, and engaging for me to write and perform.

One of the biggest musical influences of my work is the music of Steve Reich. Specifically, his ideas and work regarding rhythm and melody have directly influenced my own compositions in both of those regards.

> Reich was one of a number of composers moving in a broadly similar direction - one characterized at the time (often negatively) as a reaction against the prevailing complexities of total serialism. In place of atonality, constant variation, and rhythmic asymmetry, these composers proposed a steady-state tonality, a fixed rhythmic pulse, and unremitting focus on a single, slowly unfolding pattern that anyone could follow who had a mind to do so (Hillier 2002, 5).

Reich's music and work was shaped by his reaction to the music that surrounded him at the time of his development as a composer, leading him to the evolution of his ideas and theories regarding both rhythm and melodic material. These musical ideas have been influential in my development as a composer. Additionally, my reasons for pursuing and writing the music that I have composed during my time as a student parallel Reich in that each piece I have composed has been broadly shaped and influenced by the music that surrounded me. I have noticed a distinct difference in the way that experimental art music written within the academy is reacted to and consumed by listeners versus popular or commercial music written outside of the academy. While I do not have the space within this document to compare and contrast academic and commercial electronic music, one

of the central pillars of my work is to musically explore these differences with my compositions and performances, attempting to elucidate and bridge the differences that exist. Put another way, academic art music often explores and offers new ways of creating and conveying musical languages or syntaxes, whereas popular or commercial music seems generally more concerned with using well-established harmonic and formal languages that do not impede the consumption, comprehension, appreciation, and support of the music. One of my sincerest hopes is to create music, instruments, and performances that appeal to both sides of this musical spectrum, and can be appreciated by both analytical and casual listeners.

Studying and adopting some of Reich's compositional techniques in regards to rhythm and melody and incorporating them into my body of compositional techniques has allowed me to craft and refine my personal creative style and musical fingerprint. For example, Reich's 'rhythmic construction/reduction' plays the largest role in terms of my compositional influences. Rhythmic construction may be understood as "the [gradual] substitution of beats [or notes] for rests until a complete pattern is revealed, and then the superimposition of the same process beginning at a different point in the pattern, or at the same point but on a different beat" (Hillier 2002, 5-6). The opposite of this construction is simply the reverse, where rests are gradually substituted for beats and notes. Not only has rhythmic construction guided my use of rhythm within my pieces, but I have also extrapolated the basic idea behind it and applied it to my use of loops and melodic material. For instance, instead of gradually substituting single beats or notes for rests until a particular pattern emerges, I replace that single beat or note with a single looped

piece of sonic material and slowly replace each rest or silence with a new loop, which can be heard in some way in each of my pieces discussed in this document. This technique not only yields new polyrhythms from the rhythms contained within my loops, but it also simultaneously reveals new melodic patterns and material from within a composition as well. As Reich notes in his description of *Violin Phase* (1967):

> As one listens to the repetition of the several violins, one may hear first the lower tones forming one or several patterns, then the higher notes are noticed forming another, then the notes in the middle may attach themselves to the lower tones to form still another. All these patterns are really there; they are created by the interlocking of two, three, or four violins all playing the same repeating pattern out of phase with each other. Since it is the attention of the listener that will largely determine which particular resulting pattern he or she will hear at any one moment, these patterns can be understood as psychoacoustic by-products of the repetition and phase-shifting. When I say there is more in my music than what I put there, I primarily mean these resulting patterns. Some of these resulting patterns are more noticeable than others, or become noticeable once they are pointed out…The listener thus becomes aware of one pattern in the music that may open his [or her] ear to another, and another, all sounding simultaneously in the ongoing overall texture (Reich 2002, 26).

These 'resulting patterns' are what intrigue me the most about Reich's constructive/ reductive technique. Using relatively simple blocks of sound or melody or rhythm as individual loops, and then gradually layering them on top of each other in slightly different ways (or in reverse) for each performance yields subtly different musical outcomes and patterns with each performance and for each particular audience member. This technique is what lies at the heart of my music and draws me to the improvisatory, loop-based music of my Digital Portfolio Dissertation.

CHAPTER III

INDIVIDUAL PORTFOLIO COMPOSITIONS

III.1 *ÉTUDE NO. 1, FOR CURVE*

Introduction

Étude No. 1, for Curve is the first piece I wrote for Curve — my first interface built entirely from the ground up without incorporating a previously-built interface of any kind. It was also the first interface I built that had individually addressable RGB LEDs that reflect and highlight my physical interactions with the interface. Curve is comprised of sixty LEDs, twenty FSRs, two soft potentiometers (touch faders), a keypad, and a 9-degrees-of-freedom (9DoF) sensor package, all connected to an Arduino Mega. Although the interface contains many different controls, I designed Curve to be as ergonomic and efficient as possible in terms of physical interaction. Connecting to my computer via USB, the Curve interface becomes the primary component in a complete DDI with the addition of Ableton as the sound synthesis environment and Max as the data mapping environment.

Being the first ever composition using this new interface, a unique set of qualities presented themselves into the compositional process. To study and explore the control, performative possibilities, and affordances that this new interface offered me, I examined the musical options that Curve provided. I discovered and refined four unique modes of interaction with the interface, which inspired me to break up the piece into four discrete sections. Each section highlights one of the four performative techniques that I developed

for the interface and is denoted by a different method of physical interaction with the interface, as well as a unique lighting mode designed to correspond to and emphasize each performative technique.

Instrumental Forces

Evolving from an assigned project for a 3D modeling and design class in which I enrolled, the basic design and layout of Curve stemmed from the way that one interacts with a traditional piano keyboard. I considered the way that my hand rested on a piano keyboard, where each finger naturally fell, and what the most comfortable way to perform with my fingers might be. These considerations led me to the general layout of the interface itself. After roughly determining the size of the interface, I began to experiment with and test different shapes and layouts that were comfortable and functional for me to utilize in performance. I settled on the final shape and rendered it in a 3D modeling software called Blender, which allowed me to produce an approximation of how the interface would look when realized (see Figure 10). After finalizing the

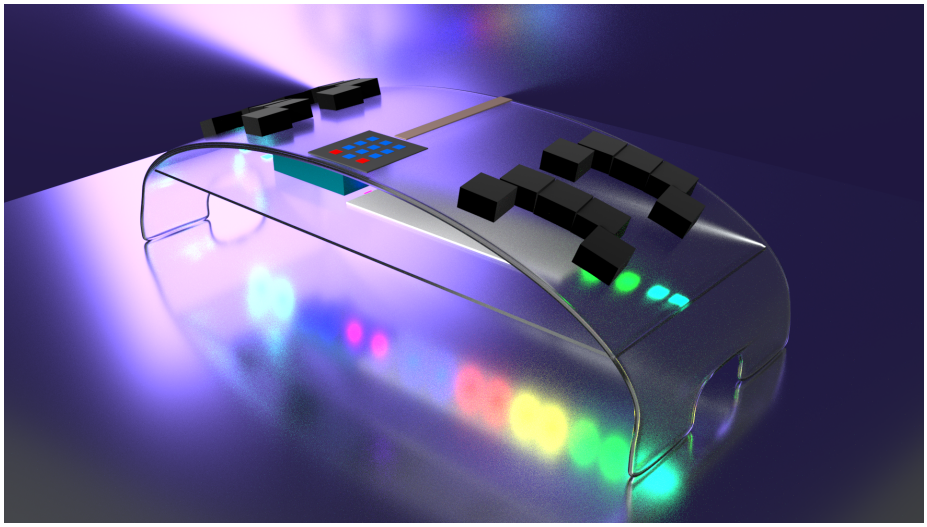


Figure 10: Final 3D rendering of Curve from Blender software

outline and layout of the interface in Adobe Illustrator, I laser-cut a piece of 1/4" clear acrylic. However, in keeping with building the most ergonomic interface possible, I then 'slumped' the acrylic over a wooden form by placing it into the oven on low heat, thereby slightly melting the acrylic and molding it into a *curved*, three-dimensional shape that could stand on its own (see Figure 11). Not only did slumping the acrylic yield a more comfortable playing surface for my hands and wrists, it also heightened the observability of the interface from an audience perspective due to the added height and expanded viewing angle that the new shape provided.

Figure 11: Final laser-cut acrylic piece (top left), wooden 'slumping' form (top right), final 'slumped' piece (bottom)

Attached to the upper and lower sides of the interface are two sets of thirty individually addressable RGB LEDs. I wanted to create an additional layer of observability for the audience and myself as the performer. By assigning each LED to a

specific data stream from one of the available sensors, I was able to create an extra layer of visual feedback that directly reflects and highlights my performative interactions with Curve. Additionally, in order to further enhance the observability of Curve in performance, I adhered small, 1.5” x 1.5” foam cubes to each of the twenty FSRs (force sensitive resistors) (see Figure 12). The added height of each cube extended the performative range of its corresponding FSR, allowing my performance of the instrument to be better observed by an audience and more musically expressive for me as a performer.

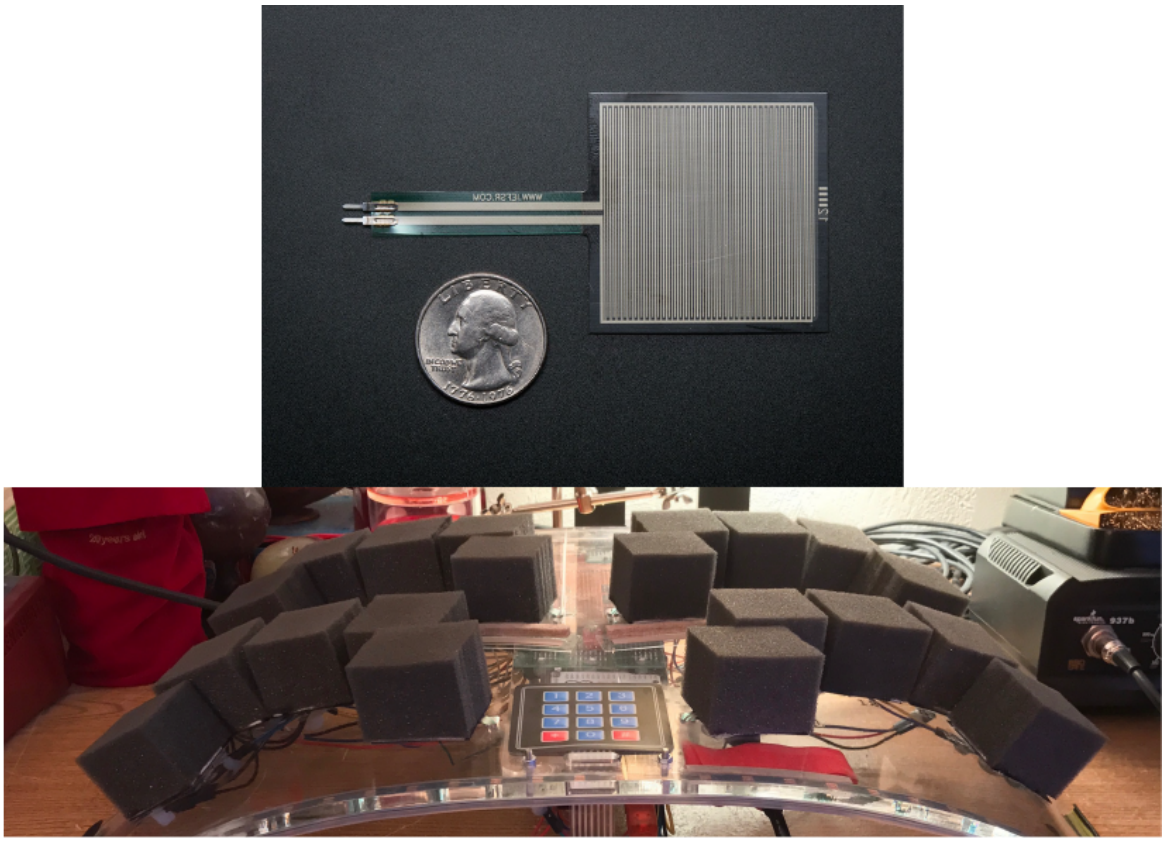


Figure 12: A single FSR (Adafruit, n.d.) (top), 1.5” foam cubes applied to Curve (bottom)

I arranged the twenty FSRs so that I could play them each with a single finger, similar to the individual keys of a piano keyboard, with each FSR available to be

assigned to a discrete pitch, sound, or loop. However, rather than have one continuous line of twenty FSRs, I divided them up into four discrete quadrants, each made up of five FSRs, with two quadrants on the upper half of Curve, and two on the bottom half. The upper and lower halves of Curve are divided by two touch faders that run across the length of the interface, acting as both a visual divider and alternate point of interaction during performance (see Figure 13). I also affixed a 12-button keypad onto Curve, providing me with the button-type control that I needed to successfully navigate my composition during performance. Lastly, I added a 9 degrees-of-freedom (DOF) motion sensor package to the underside, completing Curve's suite of sensors that I can utilize in performance (see Figure 14).

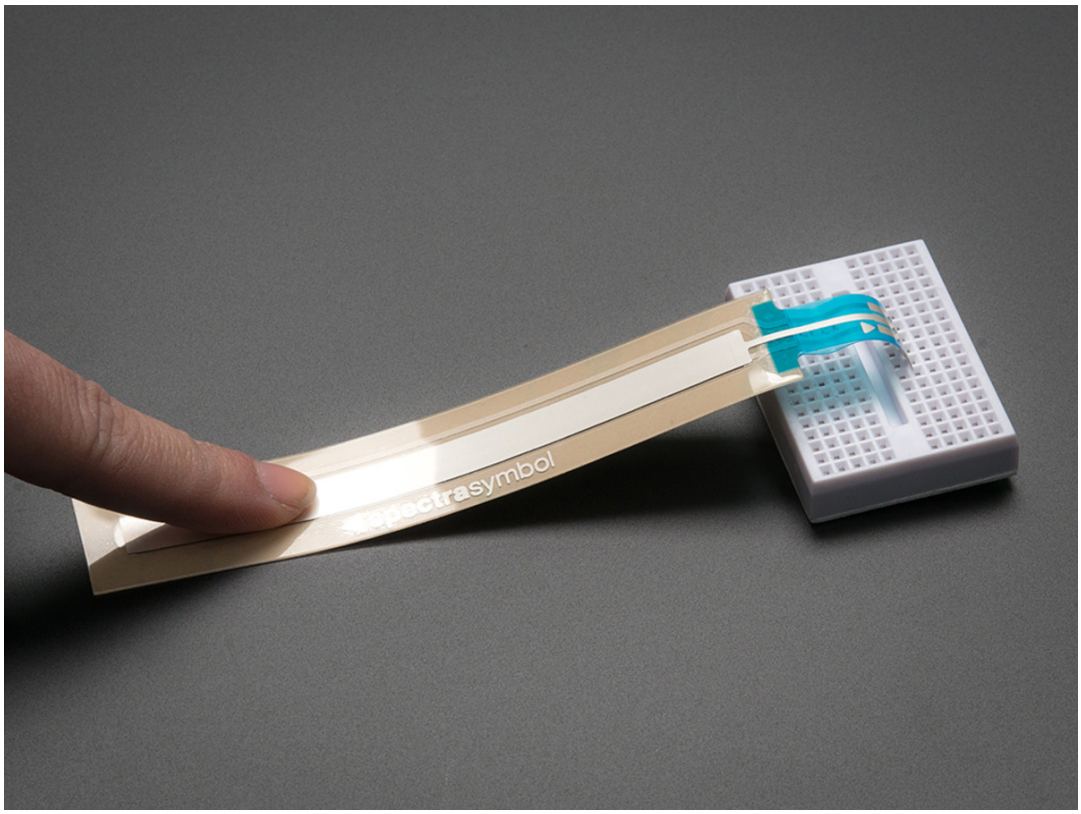


Figure 13: A touch fader (Adafruit, n.d.)

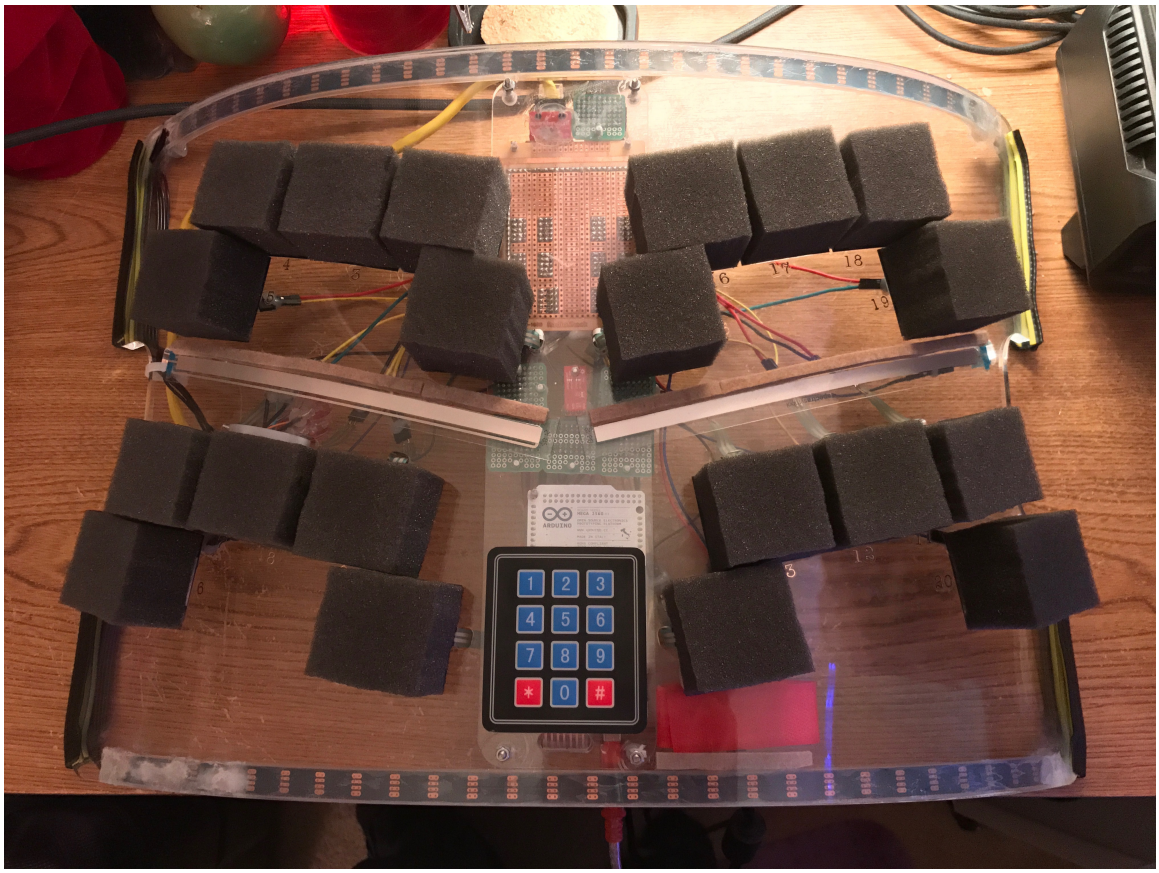


Figure 14: Final layout and arrangement of Curve (overhead view)

Each electronic component is connected to a single Arduino Mega microcontroller, which acted as the main hub and ‘brain' of Curve (see Figure 15). I chose to use the Arduino Mega because of its expanded ability to connect the large quantity of components used in the construction of Curve. The Arduino connects to my computer via a USB connection, where the serial data from each sensor is received in Max. The resolution of the Arduino Mega’s analog-to-digital converter (ADC) is 10-bit, generating sensor value ranges between 0 - 1023. After scaling the data streams in the Arduino code itself from 0 - 1023 to 0 - 255 to make the data more easily receivable in Max, I then scaled each data stream down to 0 - 127, the standard range for MIDI data. Additionally, as the added pressure of the foam cubes created unwanted data generation, I filtered out

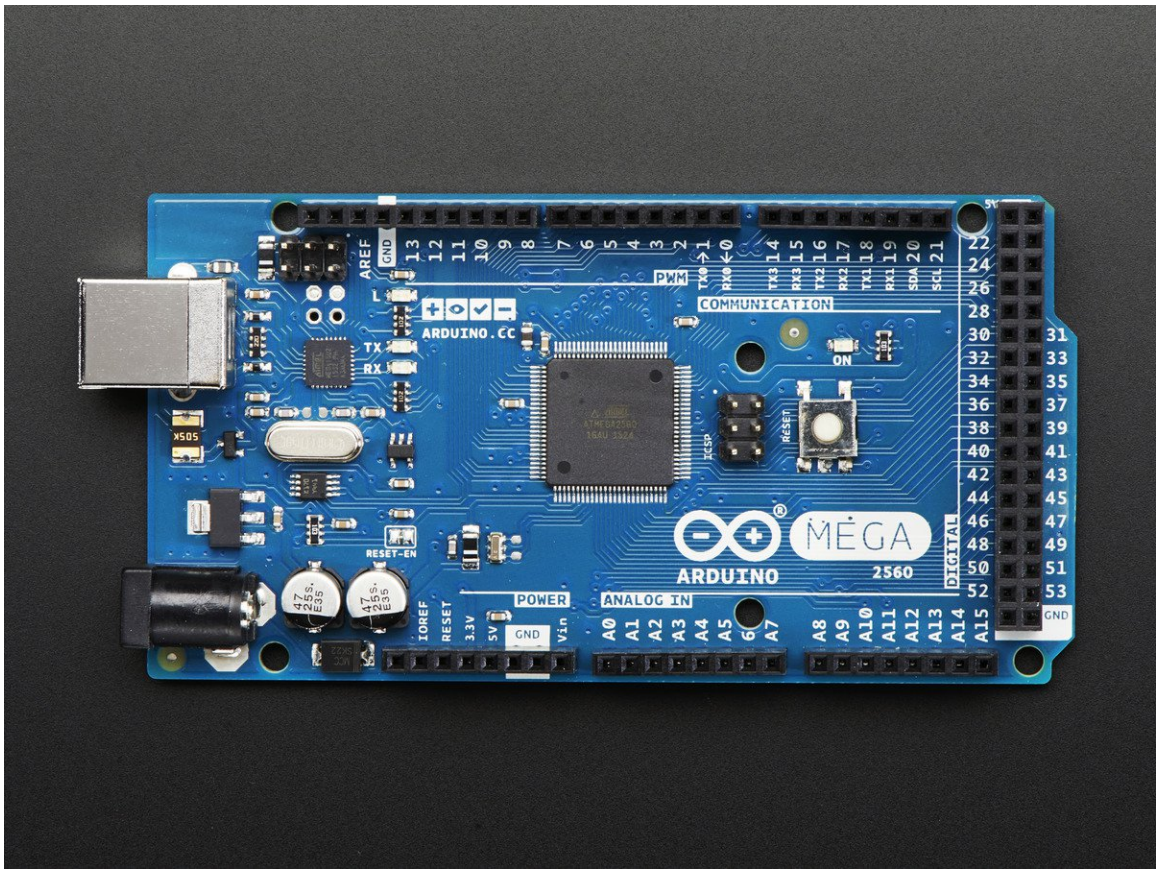


Figure 15: The Arduino Mega (Adafruit, n.d.)

the bottom 5% of each FSR's data stream to avoid any false-positives in the triggering and control of my sounds in Ableton. Meanwhile, the data streams generated by the 9DoF sensor produce values between -1.0 and 1.0. To reserve processing usage in Max I instead scaled the data using the onboard processor of the Arduino Mega, ultimately sending data streams to Max that were already between 0 and 127. Finally, the keypad generated ASCII data based upon which button is being pushed. That data was received by the Arduino Mega and sent into Max, without the need for scaling.

In addition to scaling sensor data into usable MIDI data, I also smoothed the data in Max so that it was more consistent and reliable to use in a performance environment. The raw data generated by each of the sensors aboard Curve, while highly responsive,

could end up being *too* sensitive to yield a reliable and repeatable performance. It became necessary to filter and interpolate the data within Max, removing seemingly random spikes or jumps in the continuous data streams — jitter — that could cause unwanted musical results. Using the 'line' object in Max, I interpolated between non-sequential values that occurred in each data stream to make my performance smoother and more consistent. After each data stream was transformed through mapping, they were assigned and routed to specific musical parameters in Ableton (see Figure 16).

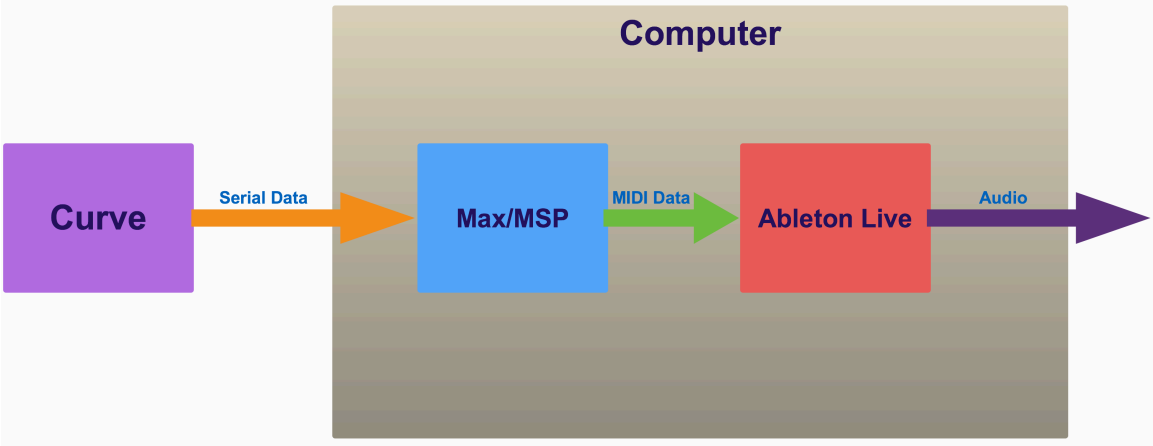


Figure 16: Overview of my DDI for *Étude No. 1, for Curve*

The lower left FSR quadrant became the bass quadrant, the lower right became a high-mid synth pad, the upper left became a rougher, more abrasive lead synth in the mid-range, and the upper right quadrant became a high, bell-like sine wave that sits atop the frequency spectrum and fills out the soundscape. Furthermore, instead of assigning each individual FSR to a particular pitch in the spirit of a traditional piano-like instrument, I sampled each of my sounds and made a collection of small, short loops that were triggered by each of the FSRs.

Sound Design

All of my sound design and synthesis was done in Ableton. My goal was to design sounds that occupied a specific niche within the frequency spectrum, but that collectively remained cohesive in their overall timbral quality. Each of the four main sounds — a bass sound, a high-mid pad, an abrasive synth lead, and a high bell-like sound — were designed to be controlled by Curve and were routed to a discrete audio channel, as the original version of *Étude* was written for a quadraphonic performance environment.

The bass sound (occurring at 3:16 in the first supplemental video) was created using a software instrument in Ableton called Morpheus, "a modular metallic instrument consisting of a series of metal bars, each attached to its own resonator" (Ableton, n.d.). Without any external effects, Morpheus sounds similar to a vibraphone or marimba-type mallet instrument, with a very pure, simple sine wave-like quality to its sound. To add more sonic character and nuance to the original Morpheus samples, I layered two different instances of the Morpheus samples, each mirroring the other and yielding the effect of a harmonic interval when each pitch was triggered individually. Each instance of the samples is slightly effected with reverb, chorus, and delay. As I wanted these sounds to fill out the bass end of the frequency spectrum, I transposed the pitches down to the F#0 - F#1 range. After sampling the sound and creating the small, short loops that I ended up assigning to each FSR, I brought the loops into my Ableton performance set. From there, I further treated the loops by boosting the bass frequencies through a combination of EQ and a harmonic maximizer plug-in, creating a boomier, stronger bass line that stands out and adds extra *oomph* to the overall soundscape.

I used an Ableton software instrument made up of samples from a Fender Rhodes electronic keyboard for the lower right FSR quadrant (occurring at 0:43 in the first supplemental video). By applying a combination of reverb, delays, and EQ effects, this sound functionally operated as a synth pad with a legato, sustained sound. The pad added an essential layer to the overarching sound field built during performance. The pitches sit in the F#3 - F#4 range, higher than the bass notes but still in the lower-mid range of the sound field. I also utilized a Max For Live granular device that reinforced the sonic intricacy of the loops in the form of subtle granulation during the sustained portion of each pitch.

For the upper left FSR quadrant, I used a Max For Live software instrument that is modeled after a semi-modular four-operator FM synthesizer (occurring at 2:05 in the first supplemental video). I chose a relatively simple configuration built around two different square waves, giving the sound a distinct, sharp quality that cuts through the other quadrants' voices, sitting atop the timbral spectrum and acting as a lead synth in the range of F#3 - F#4.

Lastly, I used one of Ableton's software instruments called Operator for the upper right FSR quadrant (occurring at 4:20 in the first supplemental video). This instrument generates waveforms with FM synthesis and then filters them with a variety of different filters and effects (Ableton, n.d.). I designed this sound to be the highest pitched and simplest timbre of the four sounds. Built from a sine wave that was processed through a Max For Live convolution reverb device, the sound resembles a bell-like tone with a short attack and longer decay. The staccato quality of each note coupled with the higher

range of F#4 - F#5 yields a collection of loops that enhance the existing sounds, completing the overall soundscape.

Composition and Performance

I structured *Étude No. 1, for Curve* so that each of the four musical sections highlighted a different mode of performative interaction with Curve. The first section focused on my interaction with each of the twenty FSRs (occurring at 0:43 in the first supplemental video). In the case of the data from the FSRs, I mapped the pressure being exerted upon an FSR to a toggle-like function that sent a MIDI note-on message and a subsequent note-off message as soon as the pressure was released. Simultaneous to each note-on message, the continuous data streams generated by each FSR were mapped to the amplitude of each sound that was being triggered by that specific FSR. The continuous data from each FSR was also assigned to a set of three of the RGB LEDs, based upon the FSR's physical location on Curve. For example, if I played the bottom right-most FSR, the three LEDs in the right-hand corner of Curve would light up accordingly, changing color based upon how much pressure I was exerting onto that FSR at that point in time.

Beginning with the lower right quadrant, I played through each quadrant one at a time, recording and constructing four 'macro' loops from each FSR's smaller, individual, 'micro' loop. Because each quadrant produced its own distinct sound, the four macro loops laid the foundation for the remainder of the piece and filled out the sonic world one element at a time. This first section was strongly influenced by Steve Reich's concept of 'rhythmic construction,' where I treated each micro FSR loop as a single 'beat,' procedurally layering them together and substituting them for silence (or 'rests') which

then made up each of the four foundational macro loops. Given the improvisatory nature in the way that I created the macro loops, never playing the micro loops of each quadrant in exactly the same order, timing, amplitude, or number, each performance of *Étude No. 1, for Curve* yielded a different set of macro loops, thereby creating a unique musical structure for each performance of the piece. This improvised foundation created a chain reaction throughout the rest of the piece, making each subsequent section of *Étude* unique to that particular performance as well. Once the final macro loop from the upper right FSR quadrant was recorded and looped, I moved on to the second section of the piece.

As the second section began (occurring at 5:30 in the first supplemental video), I moved from playing the FSRs to playing the touch-faders, highlighting the section change both aurally and visually. Given the horizontal orientation of the touch-faders on Curve, controlling the panning of the soundscape felt natural to me, since my interactions with each of them reflected the left-to-right (and vice versa) movement of a stereo audio field. To that end, and since there are two of the touch-faders on the interface, I combined the data from both of the touch-faders into a single data stream in Max. That data stream was then mapped in Ableton to control where the audio from three (of the four) FSR quadrants was sent (the fourth quadrant is the bass, which I chose not to actively pan during performance). I set up four discrete channels in Ableton, each with its own unique audio effect and panning position within the stereo field. As I moved my fingers along the touch-faders, the single data stream from Max determined to which of the four channels the audio was sent. While each of the three quadrants' audio was being moved through the stereo field, they were also moving through the different audio effects that were

assigned to each specific point within that stereo field. This scheme created the auditory illusion that there were four static points within the stereo field, each with its own unique audio effect through which the audio travelled through. Finally, to highlight the second section of the piece and the new mode of interaction with Curve, each of the touch-faders was assigned to half of the LEDs, with the left touch-fader controlling the left half of the LEDs and the right touch-fader controlling the right half. As I moved my fingers along each touch-fader, the color of the corresponding LEDs changed in accordance with the position of my finger along the touch-fader.

Utilizing the sonic material generated from the first section, the second section explored this material and developed it through the use of specific audio effects and real time spatialization. As I moved my fingers along each of the touch-faders, the audio from three of the four macro loops was sent between four different return tracks in Ableton. Three of the return tracks had a unique audio effect on them, affecting whichever audio signal was being sent to that particular track at any given point in time. Each of the return tracks was also assigned a single specific audio channel that placed the sounds in a particular spatial location within the sound field of each performance. Therefore, moving my fingers back and forth across the touch-faders was central to the performance — the controls modulated and spatialized the sounds in real time, which allowed me to explore and reveal the aural intricacies and nuances of my sounds through my performative actions. Additionally, because the audio effects were statically assigned to specific audio channels, the musical impression was that the audio was moving through audio effects

placed within the physical performance space rather than the audio effects being attached to the audio itself.

The third section of *Étude* featured the 9DoF motion sensor and was distinct from each of the other three sections in that it was the only section where I physically moved Curve itself, as opposed to playing it from a stationary position (occurring at 6:52 in the first supplemental video). For this piece, I mainly utilized the X- and Y-axes of the onboard accelerometer. Each axis was mapped to the parameter of a granulator; specifically the density and length of each grain, as well as the panning of the granulator and the file position of the buffer from which the granulator's audio was playing back. As I moved Curve through the air with respect to the X- and Y-planes of the 9DoF sensor, the audio was modulated and panned according to my movements. I also tied the data from the X- and Y-axes to the LEDs. However, instead of only controlling some of the lights based upon the location of the sensors on Curve, the data streams controlled the color of all the LEDs since the motion sensor worked by moving the entire interface.

In order to actuate the motion sensor, I physically picked up Curve and moved it through the air during performance. Sonically, this section transformed the four discrete, individual macro loops into one underlying sonic texture, based on the movement of Curve through the X- and Y-axes of the motion sensor. The Y-axis controlled the length and density of a Max For Live granulator device called Density FX, which had been pre-loaded with a 10-second recording of the four macro loops. The X-axis controlled the location of the recording's waveform that was being played through the granulator, as well as the panning of the track. Compositionally, I wanted the music of this section to

reflect the change in performative action, hence the complete transformation of the musical material during this third section.

The fourth and final section of *Étude No. 1, for Curve* functioned compositionally as a restatement that returns the material of the four macro loops from the first two sections of the piece (occurring at 9:10 in the first supplemental video). Not only did I return to the FSRs as my means of musical interaction, but the original sound material also returned to the forefront. However, instead of controlling each individual sound that was assigned to each FSR, I controlled the overall soundscape with each FSR quadrant. To do this, I first took the data being generated from each FSR, and averaged it together in Max to form a single data stream for each quadrant, based on how much pressure I exerted upon the entire quadrant of FSRs. I mapped each quadrant's data stream to a unique audio effect in Ableton, so that the average pressure from each FSR quadrant directly controlled that particular effect. Furthermore, the first time I exerted pressure on each quadrant during the final section, it triggered that quadrant's audio, allowing me to control the timing and pacing of the reintroduction of the original soundscape created in the first section. Lastly, each quadrant's average pressure also controlled the brightness of a specific color of the LEDs, creating a connection between each of the four audio effects mapped to each quadrant and the visual feedback of the LEDs on Curve. In addition to triggering the reintroduction of each quadrant's sound, the average pressure generated by each quadrant was mapped to a unique audio effect that had not been previously heard, creating a fresh yet familiar take on each of the original four macro loops.

III.2 ***T(Re)es (Trees: Remixed)***

Introduction

T(Re)es (Trees: Remixed) is an octophonic remix that derived from an original composition I wrote in 2009 for my band, Hamilton Beach.[14] To create the new composition, I broke down each individual part from the original piece into short, simple audio recordings that could be re-articulated or looped in order to shape my musical palette. This looping technique allowed me to maintain a strong connection to the original piece while still being able to craft and shape an entirely new sonic journey within a performance. Additionally, I employed finger-drumming using the same percussion sounds contained in the original version of *Trees* to strengthen the compositional and performative connection to the original piece. The Monome was my main performance interface for ten-plus years during my performances with my band and for the original version of *Trees*, so by utilizing the Monome as the performance interface for this piece, I was also able to stay true to the soul of the original music.

Instrumental Forces

For *T(Re)es*, I used a performance interface with which I was more familiar than any other due to my use of the interface with my band — the Monome. Designed, invented, and initially released in 2006 by Brian Crabtree and Kelli Cain, the Monome began as a simple, elegant, and minimalist hardware device built as an 8 X 8 grid of small, individually backlit buttons housed in a square, wooden housing (see Figure 17).

---

[14] "Dear Earth, Love Moon, by Hamilton Beach," *Hamilton Beach*, accessed April 10, 2020, https://hamiltonbeach.bandcamp.com.

The elegance of the Monome lies within its simplicity. Instead of having any sort of predefined software layer or script that dictates how, when, or why it should be used, the Monome is a completely blank slate that simply sends and receives OSC data via a USB connection.

Figure 17: The 2008 Monome grid 64 model (Monome, n.d.)

The Monome I utilized in *T(Re)es* is the 2008 grid 64 model that is comprised of 64 backlit buttons and an internal accelerometer. Because the Monome does not send specific musical directives, a software environment that can map generic OSC messages to musical commands was necessary. I used Max to accomplish this task (see Figure 18). When coupled with Max, the Monome became a fully-functional interface that I could customize. In *T(Re)es*, I developed two performance mechanisms that I utilized in the piece, with each mechanism dictating what kind of button (momentary or toggle) was engaged. First, each time a button was pressed a momentary trigger was created. Second, each time a button was pressed it was interpreted as the 'on' or 'off' of a toggle. I divided the Monome's 64 buttons into two distinct sections, one that utilized the buttons as the

momentary triggers, and the other that used the buttons as toggles. This arrangement allowed me to develop the structure of the piece by assigning a specific sound to a particular toggle, which permitted me to leave each sound playing and looping until I physically chose to turn it off. Then, the momentary triggers allowed me to utilize the finger-drumming technique that I had been mastering over the last decade. The juxtaposition created by using the 64 identical buttons of the Monome, but in two different, discrete modes allowed me to highlight both modes of interaction with the Monome during the performance of the piece.

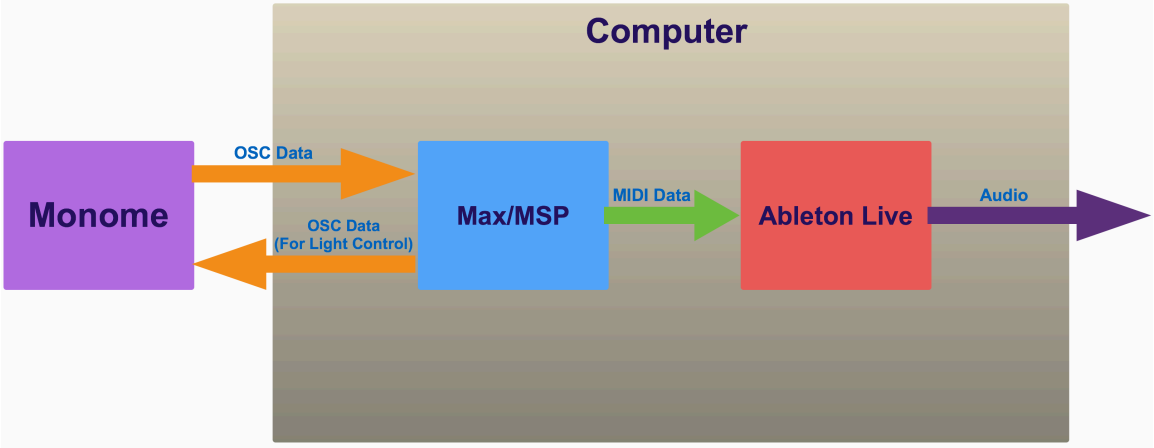


Figure 18: Overview of my DDI for *T(Re)es (Trees: Remixed)*

T(Re)es stands apart from each of the other pieces in my DPD in that it is the only piece in which I did not use any type of fader, dial, knob, slider, or other type of sensor that generates continuous, ongoing control over musical parameters. I created two automated fadeouts in Max that were triggered by buttons on my Monome to fade out the bass and then fade out the piece upon its end, but those were the only two instances of MIDI control change (CC) data being generated in the piece.

Sound Design

Every sound heard in *T(Re)es* was derived from the original version of *Trees*. I started with each of the five original main sounds that made up *Trees*: a 'prepared' piano made from Ableton's Tension software instrument, a small piano made from Ableton's Collision software instrument, a grand piano sampled by Ableton, a synth pad made from Ableton's Analog software instrument, and a synth bass made from Ableton's Operator software instrument. After bouncing each individual instrument track into separate sound files, I then chopped up the prepared piano, small piano, grand piano, and synth pad into eight very small, short loops that were each assigned to a single specific toggle and audio channel, thereby creating an octophonic performance environment by placing each loop in its own spatial location. Instead of making small loops out of the bass instrument, I copied the software instrument and corresponding MIDI file into my new *T(Re)es* Ableton set, where I chose six chords (from the original version) to play. Finally, I copied the software drum kit and latin percussion samples from the original version of *Trees* into the new performance set for me to utilize during the performance of *T(Re)es*. Each sound was crafted to be triggered by the individual buttons offered by the Monome. This led me to create sounds that were short in duration that were meant to be played momentarily (for percussion) or looped (for original *Trees* loops) to form a musical structure.

Composition and Performance

Writing music for my academic endeavors has generally remained separate from the music that I wrote for my band because I enjoyed having separate musical outlets for my work. Because both sides of my compositional interests represent the entirety of

myself as a composer, I wanted to combine the two styles into one cohesive piece. A performance of *T(Re)es* was driven by the rapid and virtuosic depressing of the buttons of the Monome to trigger the prepared and sampled sounds from *Trees*. I mapped the thirty-two buttons in the top four rows to each of the thirty-two loops I made out of the original *Trees* software instruments (the prepared piano, small piano, synth pad, and grand piano, respectively), and used six more of the buttons from the fifth row to act as toggles for each of the bass chords (see Figure 19). Each of the buttons from the bottom four rows, excluding the bass toggles, functioned as momentary buttons and control the finger-

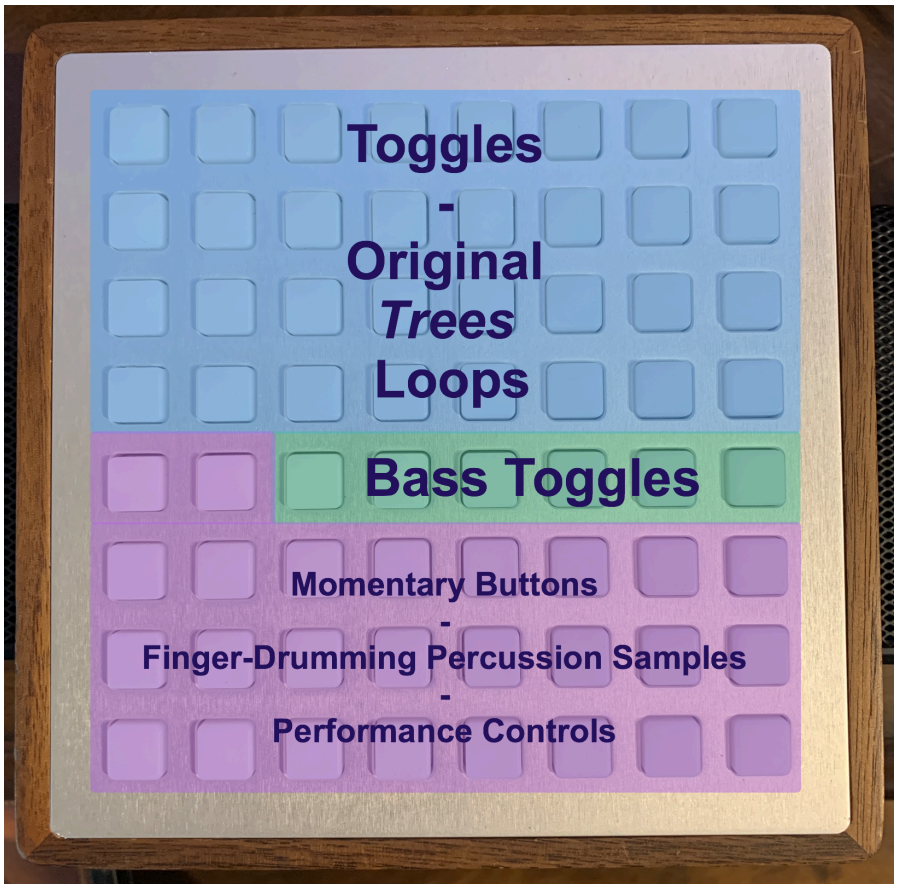


Figure 19: Diagram of Monome button mappings

drumming samples, the triggers for the beginning of the piece (triggering my performance clock and the Ableton global transport), and the fadeout of the bass and composition (at its conclusion). Compositionally, *T(Re)es* was heavily influenced by Reich's rhythmic construction and reduction technique.

I began the piece by toggling on a single loop from the second row (the small piano). Each subsequent loop that I introduced into the sound field acts similarly to how the beats and rests work in Reich's pieces (i.e. *Drumming)* based on rhythmic construction and reduction (occurring at 0:36 in the second supplemental video). Additionally, I utilized a performance technique similar to my finger-drumming technique, where I rapidly turn on and off the loop toggles that I have been layering together. When coupled with the improvisatory nature of the composition and performance of the piece, each instance of the performance generated many unique polyrhythms, melodic patterns, and aural results that together create an idiosyncratic version of the piece each time it is performed.

I moved through each row of loop toggles one at a time, from the small piano loops to the synth pad loops, to the prepared piano loops, to the grand piano loops, building the musical structure one loop at a time. As I explored the different musical combinations of each toggled loop, I eventually formed a static combination of loops for the remainder of the piece. Once I had a sonic texture in place, I began the second section by shifting my performance from the toggles to the momentary buttons, introducing the percussion samples and a new mode of performance in the form of finger-drumming (occurring at 5:16 in the second supplemental video).

A technique that I adopted and started using consistently in performance around 2008, finger-drumming has been an important part of my performative repertoire for over a decade. Finger drumming is a term used to describe a performance technique that derives from traditional African and Latin percussion techniques, but is also now often employed to describe a method used by button-centric or percussion-modeled electronic interfaces that involves rapid hand and finger movements. I start with a collection of Latin percussion samples consisting of quintos, bongos, congas, and tumbas, and then added a hi-hat sound as well. As I locked in on a rhythmic pattern to play with my fingers, I finally added in bass chords (occurring at 6:05 in the second supplemental video), toggling them on and off to create a melody that filled out the sound field and articulated the apex of the piece. As the climatic moment ends, I triggered the final fade out of the bass and one by one turned off each of the underlying loops while continuing to trigger percussion samples with my finger-drumming technique. The piece closed as I turned off the final toggle loop, and slowly allowed the rhythm of my finger-drumming to fade out.

III.3 ***Trio 1-465***

Introduction

Trio 1-465 is a composition for cello, turntables, and a data-driven instrument comprised of Contour Design's RollerMouse Red Plus, Max, and Ableton. Formed around the notion of a traditional jazz trio, *Trio 1*-465 was developed in collaboration with Zachary Boyt (cello) and Connor Sullivan (turntables). One of the central ideas of

the piece was to sample sounds from the cello in real time on my computer and then use the turntables to manipulate those same sounds, also in real time. We were able to realize this idea by employing Ms. Pinky's custom control vinyl and software environment. By transforming the turntables into data generators using the Ms. Pinky control vinyl, I was able to map the corresponding data streams from each turntable to control the sounds of the cello that are captured in real time, maintaining the same affordances that a traditional turntable and vinyl record provides.

Instrumental Forces

The RollerMouse is a device that emulates the function of a traditional computer mouse. Originally designed as a more ergonomic alternative to a generic computer mouse, the RollerMouse is made up of a rolling pin-like bar that the user can move upwards and downwards, or side-to-side. In its default orientation, the RollerMouse lies flat on a table and is positioned horizontally in front of the user. When using the RollerMouse, the bar is rotated (literally spun around), where the rolling motion corresponds to the Y-axis (up and down) and the side-to-side motion corresponds to the X-axis (left and right). Additional buttons are located in the center of the RollerMouse, including left- and right-click buttons, a double-click button, and copy and paste buttons. Also located in this central area is a clickable scroll-wheel, though not all of these elements were used in my performance of *Trio 1-465*. The RollerMouse is connected to my computer via a standard USB connection.

The data from the RollerMouse enters Max as generic mouse data. Max contains an object called 'mousestate' that receives, routes, and outputs that mouse data, and after

scaling it, I was able to utilize the data from the RollerMouse as usable data streams to control musical parameters in Ableton. In order to maximize my control over those data streams, I chose to rotate the RollerMouse by 90 degrees (see Figure 20), turning the

Figure 20: The RollerMouse Red Plus (Contour Design, n.d.)

side-to-side motion into the up and down motion, and the rolling motion into side-to-side motion. Because I rotated the orientation of the RollerMouse, I was able to assign the new up and down motion via Max to the note control of my software instrument, and the side-to-side motion to the amplitude of each note, yielding a more observable field of motion to an audience that also felt more natural to me as a performer. Additionally, the new side-to-side motion allowed me to exert more nuance and fine control over the data stream that controls each note's amplitude. I then assigned the buttons of the RollerMouse to trigger each note, control the start and stop of the cello recording buffers, and to control which part of the mapping and audio processing software should be used.

I was able to transform Connor's turntables into a fully-functioning DDI through the use of the Ms. Pinky 'Interdimensional Wrecked System' (IWS). In brief, the IWS is

an environment that permits skilled turntablists to send control data as audio streams that reside in external control vinyl records. In the case of the *Trio 1-465*, control messages were sent from the turntable mechanism into Max to control the audio that is generated throughout the performance of the piece. Using the custom Ms. Pinky control vinyls (see Figure 21) in conjunction with Max and the Ms. Pinky Max For Live devices, I was able

Figure 21: A Ms. Pinky control vinyl record (Ms. Pinky's Playhouse, n.d.)

to harness the data that is generated by Connor's interactions with his turntables and map it to specific musical parameters. For *Trio 1-465*, I shaped and mapped Connor's data so that it could control the real time samples that I record of Zachary's cello, much like a traditional turntable controls the sound that is recorded onto a traditional record.

Sound Design

Sonically, *Trio 1-465* is a complex network of audio routing, manipulation, and synthesis (see Figure 22). Ableton served as the main audio matrix, hub, and DDI sound synthesis engine, where I was able to generate the sounds for my DDI, route and connect the audio from the cello (via Max where it is initially received) to the turntables, and compose the overarching structure of the piece. Each time I triggered a pitch with the RollerMouse, two sounds were played simultaneously (occurring at 0:37 in the third supplemental video). One sound was made with a software instrument based on the MiniMoog Voyager analog synthesizer. After adding reverb, I used the sound as the main bass timbre for the piece. The second sound was created with a variant of the vibraphone-like software instrument that I utilized in *Étude No. 1, for Curve*. This sound was used as a high-pitched bell accompaniment to the more prominent bass timbre. The two sounds complimented each other and provided a balanced timbral space when accompanied by the turntables and cello. Zachary's cello was unaltered in terms of sensors or data-generating electronics, but instead was mic'd with a contact microphone that was connected to my computer through an audio interface. A small amount of reverb was added to the cello to add depth and nuance to the sound (occurring at 0:57 in the third supplemental video). However, the cello's sound was also routed into Max, where its sound was recorded into a 20-second buffer and sent via the Ms. Pinky Max for Live device to Connor's turntables. Connor was then able to scratch and manipulate the sound of the cello in real time, to complete the Trio's soundscape (occurring at 1:24 in the third supplemental video).

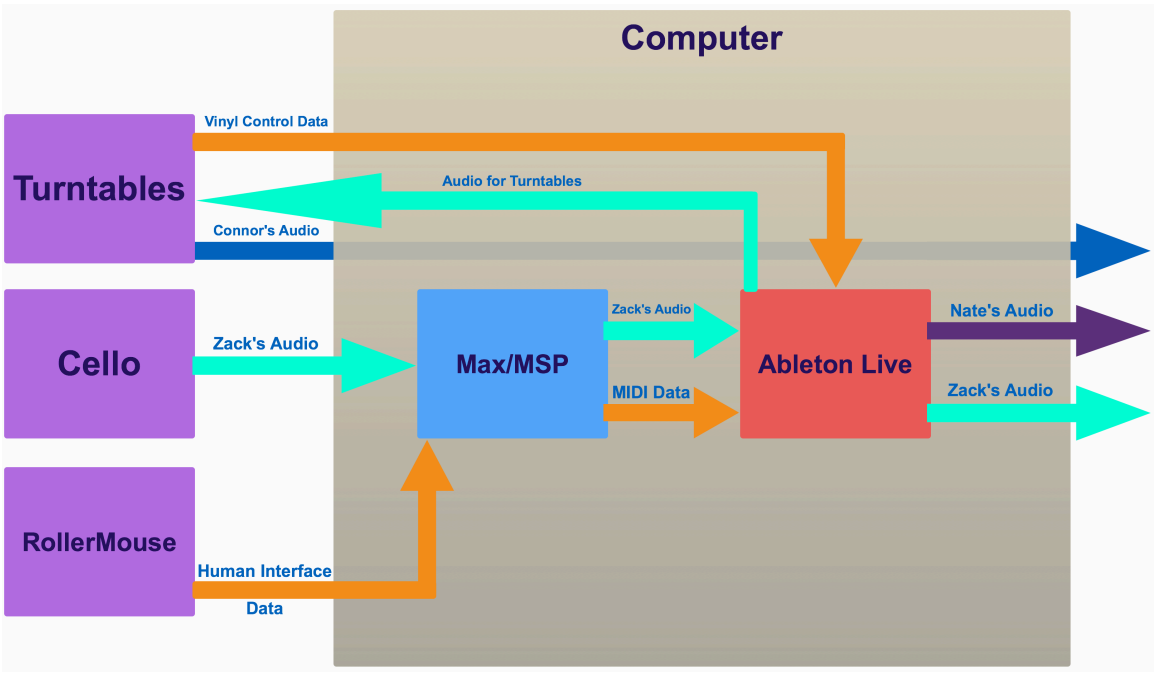


Figure 22: Overview of my DDI and audio routing for *Trio 1-465*

Composition and Performance

The main muse behind my instrumentation and performance of *Trio 1-465* was the idea of a traditional jazz trio. I wanted to emulate the way that so minimal an ensemble can create such a dynamic, virtuosic, and *complete* sound and performance with only three musicians. Each of our instruments resided in a particular niche within the overall sound field, which, when heard simultaneously together in performance, yielded a balanced and innovative sonic journey that we navigated together as an ensemble. My DDI for *Trio 1-465* functioned as the rhythm section would in a traditional jazz trio, laying the groundwork for each section by creating the initial loop (and repeating rhythm), and then playing the bass line in the corresponding mode to denote each of the seven sections of the piece and create a cohesive musical structure. Zachary's cello was the main source of melodic material outside of my bass lines, and Connor's turntables

enhanced and complimented the sound field by reinforcing the rhythmic material from my DDI and the melodic material from the cello.

Compositionally, I was inspired by the concept of reinterpreting memories of past musical styles — Gregorian Chant and jazz, in this case — in a more modern way. To that end, the musical form of the piece is divided into seven individual sections. Each section takes on the form of a church mode that was used in Gregorian Chant, which I chose because of chant's historical and musical significance within the history of Western tonal music. For *Trio 1-465*, the modes I employed for each section were (in order of performance) Lydian, Locrian, Phrygian, Aeolian, Dorian, Mixolydian, and Ionian. Each section of the piece is improvised, achieving a musical structure that is globally determined but locally improvised. Each church mode is performed using the pitches of an F# major scale, but by changing the finalis (or the mode's 'tonic') and order of how those pitches are played, we were able to suggest each of the seven different church modes that I chose to play through during the course of the piece. For example, to highlight Dorian mode using the pitches of the F# major scale, we would play them in this order: G#, A#, B, C#, D#, E#, F#, with the most weight being on the G# (as the finalis), suggesting Dorian as the current mode. This compositional method allowed me to achieve a sense of cohesiveness, while also maintaining a sense of movement and journey throughout the performance of the piece.

Trio 1-465 is the only piece of my DPD in which I notated a score of any kind (see the Appendix for full score). Given that this was an ensemble-based work and not a solo piece for only myself, I used a score to make it easier for the ensemble to play

together and follow each other during performance. I chose to visually represent each of our parts by a different color, demarcated by a separate lane. The score is further divided by mode, and visually depicts the sonic elements heard within each section of the piece.

The form of the piece generally follows the same pattern through each of the seven sections, where I build loops against which the cello and turntables can sound. Concurrently, I record fresh 20-second loops of the cello into a buffer for the turntables, giving Connor new audio at the outset of each section to scratch and manipulate. The piece comes to a close at the conclusion of the seventh section, played in the mode of Ionian. I chose Ionian as the final mode and section of the piece because it is the equivalent of playing the F# scale in its original major mode, with the F# being the finalis and 'tonic' of the piece as a whole.

III.4 ***Impromptu No. 1, for ShrutiBox+***

Introduction

Impromptu No. 1, for ShrutiBox+ is a composition I wrote for an instrument I designed called ShrutiBox+. This piece encapsulates and showcases the idea of enhancement both musically and technologically through a performance on an "augmented" shruti box (Miranda, Wanderley, and Kirk 2006). By creating improvised loops in real time using the natural sound and pitches of the shruti box, I developed the structure for the electronic side of the piece, which allowed the DDI facet of the instrument to be brought to the forefront during a performance.

Instrumental Forces

The performance interface for *Impromptu No. 1, for ShrutiBox+* was predicated on an existing musical instrument called a shruti box (see Figure 23). A shruti box is an

Figure 23: My shruti box, before electronic enhancement

instrument of Indian decent that works on a system of reeds and bellows, similar to that of an accordion or harmonium. I wanted to augment and enhance the capabilities of this traditional instrument using digital technology. To execute these instrumental

enhancements I used four touch-faders (two linear, two rotary), a keypad, and a time-of-flight distance sensor. Measurements made by these components were sent to an Adafruit Feather M0 microcontroller board that served as the hub and point of connection for each sensor. The Adafruit Feather converted voltages sent to it from each sensor into data streams that were sent to my computer via a USB connection.

One of the most important restrictions I set for myself was to not harm or alter the original instrument in any permanent way. I wanted everything I added to the shruti box to be completely removable and impermanent. Working together with my Digital Arts mentor, John Park, we were able to devise a way to utilize the screws and holes that were already present on the instrument to affix three custom-designed, laser-cut pieces of acrylic to the three main edges of the shruti box. The inclusion of these acrylic layers permitted the attachment of my technological components without any damage to the original instrument (see Figure 24).

On the top overlay I placed one rotary and two linear touch-faders. On the right side I placed the second rotary touch-fader and the distance sensor. On the left overlay I placed the keypad and Feather M0 board. Configuring the components in this manner allowed me to perform the piece by easily switching between playing the original shruti box and operating the added electronics and attached sensors. The Feather M0 board sent the data into Max, where I then converted the serial data into MIDI data. The audio produced by the acoustic mechanism of the shruti box was sent into Ableton for sound processing (see Figure 25).

Figure 24: The electronics and acrylic layers attached to my shruti box

Each of the ShrutiBox+'s sensors were assigned to a particular musical parameter within Ableton. After initially scaling the data streams generated by the touch-faders in Max, I converted the data streams into MIDI data and assigned each of them to a specific pair of effects parameters in Ableton. I also scaled the data stream generated by the distance sensor. In this instance, I multiplied the distance sensor's data stream by four and then split it into four individual streams, allowing me to assign multiple data streams from a single sensor to four different effects parameters in Ableton. Finally, I used the

buttons on the keypad to control the loops, the beginning and end of the piece, and the sectional changes.

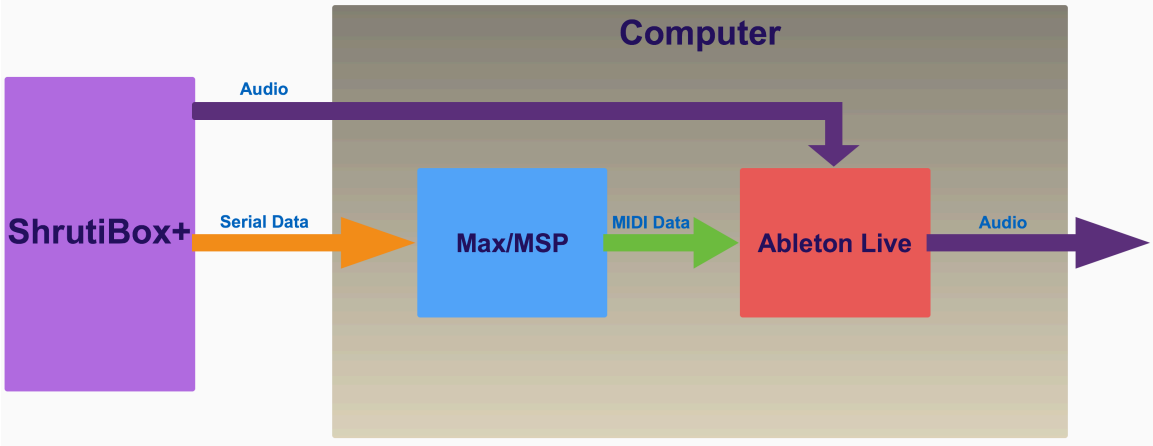


Figure 25: Overview of my DDI for *Impromptu No. 1, for ShrutiBox+*

Sound Design

All of the sounds heard in *Impromptu No. 1* originated from the natural sound of the shruti box, as I wanted this piece to center around the exploration of the sounds of the shruti box through the use of my added electronics and sound design. I began each of the first four sections of the piece by playing an unaltered, unaffected note or chord. As each section progressed, I introduced audio effects that were controlled by a specific touch-fader (one touch-fader per section), thereby modulating and changing the original sound as I performed.

The first sound heard in the piece was a single note: C5. To sonically transform this unaltered sound, I used an audio effect made from a combination of a Max for Live granular device and an Ableton reverb device. The combined effect was introduced gradually over time, achieving a subtle change in the overall soundscape by granulating and adding reverb to the sound (occurring at 1:31 in the fourth supplemental video). I

used a Max for Live reverb device to effect the next note I introduced: F5 (occurring at 2:47 in the fourth supplemental video). When I modulated the parameters of the reverb to extreme positions, the effect completely transformed the original audio signal into a new timbre.

I used a Max for Live spectral delay device created by the Institute for Research and Coordination in Acoustics/Music (IRCAM) (occurring at 4:46 in the fourth supplemental video) to modulate the next notes I introduced: C4 and G4. Using custom parameter settings, the device acts as a kind of resonator. Later I applied a combination of devices consisting of a Max for Live delay device and another IRCAM-developed Max for Live transposition device for the final notes I introduce: C3, F3, and A3. The delay device is a six-tap delay line with independent bandpass filters on each tap, and, after employing custom parameter settings, functioned as a kind of resonator for the final chord. To enhance the lower frequencies of the sound field (occurring at 6:13 in the fourth supplemental video), I used the transposition device to transpose these final three notes down one octave.

While much audio processing was applied to the live audio streams, some effects were instead applied to the looped audio of the shruti box. Initially, I added a rhythmic layer to the sound field through the use of a Max for Live granular delay device (developed by IRCAM) and an Ableton delay device (occurring at 6:53 in the fourth supplemental video). I utilized a complex effects chain for the final section that was comprised of three different Ableton devices chained in series. Together, the devices created the overall effect of bit reduction, taking the original soundscape and ‘reducing’

its sonic quality (occurring at 8:11 in the fourth supplemental video). This signal 'reduction' was very important because I wanted the final sound to depart from the established soundscape while still maintaining an aural connection to the original material.

Composition and Performance

Because I wanted *Impromptu No. 1, for ShrutiBox+* to reflect the nature of the instrument itself — droning, slow, gradual, elegant, rich — the piece is relatively minimal in terms of density and prevalence of individual notes. The piece is comprised of five individual sections, with each of the first four adding to and layering on top of the previous section's sound, creating the fully established soundscape by the conclusion of the fourth section. The soundscape was built up slowly and gradually through those first four sections, beginning with the single note of C5. After the first note, I then allowed the sound to die back down as the bellows of the shruti box deflated, repeating this process multiple times until the note was at full volume, creating a kind of 'breathing' effect. Once the note was at full volume, I introduced the granular and reverb effect that I had assigned to the first section's touch-fader. After adequately affecting the sound, I then recorded a loop of the audio in real time, setting the first layer of the piece. The second section mimics the first in regards to form and structure, but was based on the note F5, creating an interval of a perfect fourth with the loop that was already playing and adding a layer of aural complexity and richness to the soundscape.

While the third section followed the same form as the previous two sections, in this section I introduced the first instance of polyphony. Beginning at an octave below the

first note of the piece on C4, I left that note sounding while alternating between D4 and E4, finally settling on the interval of a perfect fifth with G4. After then transforming the sound through my interaction with the corresponding touch-fader, I recorded the loop and added it to the existing soundscape.

Polyphony also played a key role in the composition and performance of the fourth section. Featuring the lowest octave on the shruti box, the fourth section began on C3. I slowly brought in F3 followed by A3, creating the final chord performed in the piece. As I applied the corresponding touch-fader's audio effect to the chord and recorded the loop, I was able to add bass to the soundscape, completing the overall sound field and laying the foundation for the final section.

Because the fifth section did not utilize any live audio input, nor did it utilize any of the touch-faders affixed to the instrument, it stood apart from the previous four sections of the piece. I wanted the final section of *Impromptu* to reflect a new area of sonic and performative exploration and to signify a satisfying conclusion to the musical journey of the piece. In the fifth section my performative focus shifted entirely to the distance sensor. Interacting with the distance sensor highlighted the departure from the previous sections because I introduced a new performative action. Instead of interacting with the surface of the instrument itself, the distance sensor necessitated that my hand physically leave the instrument's surface. When combined with this shift in performative interaction, the new sonic qualities of this section created a sense of conclusion. To create a musical symmetry and to maintain a strong connection to the beginning of the piece, the piece ended with a short restatement of material from the beginning of the composition. I

slowly removed the effects from the sound controlled by the distance sensor and the rhythmic element of the delay in the effects chain to complete the piece with the sonically rich, droning quality of the shruti box itself, mirroring the sounds of the unaffected audio present at the beginning of the piece.

III.5 ***Multios***

Introduction

Multios is the fifth piece in my DPD, and is the only composition where I did not incorporate a desktop or laptop computer in any aspect of its composition or performance. Taking place entirely on an Apple iPad Pro and iPhone, *Multios* exists wholly within an iOS environment, utilizing an ensemble of applications that are chained together to form the compositional and performative software environments. Sonically, the sounds I used for *Multios* derived originally from the shruti box, which were modified and shaped using an app called iDensity.[15] By creating improvised loops of the sounds' output from iDensity with the help of another app called Loopy,[16] I was able to form the compositional and performative infrastructure for *Multios*. The piece was performed in four sections, with each section highlighting a different audio sample from the shruti box. To create data streams that controlled the sound generation and modification executed within iDensity, I used an app called TC-Data.[17] When iDensity and TC-data were then

---

[15] by apeSoft - "IDensity," *Electronic Music Software*, accessed February 10, 2020, https://www.apesoft.it/idensity/.

[16] by A Tasty Pixel - "Loopy, the Live Looper App for IPhone and IPad," accessed April 4, 2020, https://loopyapp.com/.

[17] by Bit Shape - "TC-Data - Bit Shape," accessed February 10, 2020, http://www.bitshapesoftware.com/instruments/tc-data/.

connected within my software system to Audiobus 3, Audiobus Remote,[18] and AUM,[19] I was able to create my performance control data, map that data appropriately, and synthesize the audio in real time. This combination of iOS hardware and software created a complete DDI and compositional environment.

Instrumental Forces

My DDI for *Multios* is not as easily parsed as the other DDIs in my DPD because the hardware interface, data mapping software layer, and sound synthesis environment all reside within a single device: an iPad Pro. An iPhone played a supplemental role in the performance of the piece, running a single app that synchronizes with the iPad and allows me to control Loopy. But because the main hardware performance interface was comprised of an iPad Pro, the majority of the design and implementation of the DDI for this piece is software-based and took place entirely within iOS. I began my software design and data mapping with TC-Data (see Figure 26). TC-Data is an app that generates MIDI data based on a person's physical interactions with the iPad's screen (TC-Data, n.d.). I interacted with the iPad using my fingers, and TC-Data allowed for my interactions with the iPad's screen to be customized and transformed into expressive and interesting musical outcomes. TC-Data also allowed me to split the iPad's screen into up to four discrete sections, each having its own set of selected data streams that were separate from the other screen sections. For *Multios*, I split the screen into two sections, the left-hand side producing five data streams that controlled the real time sound

[18] by Audiobus Pty - "Audiobus: Live, App-to-App Audio.," accessed April 4, 2020, https://audiob.us/.

[19] by Kymatica - "AUM - Audio Mixer," *Kymatica.Com*, last modified February 4, 2020, accessed February 10, 2020, http://kymatica.com/apps/aum.html.

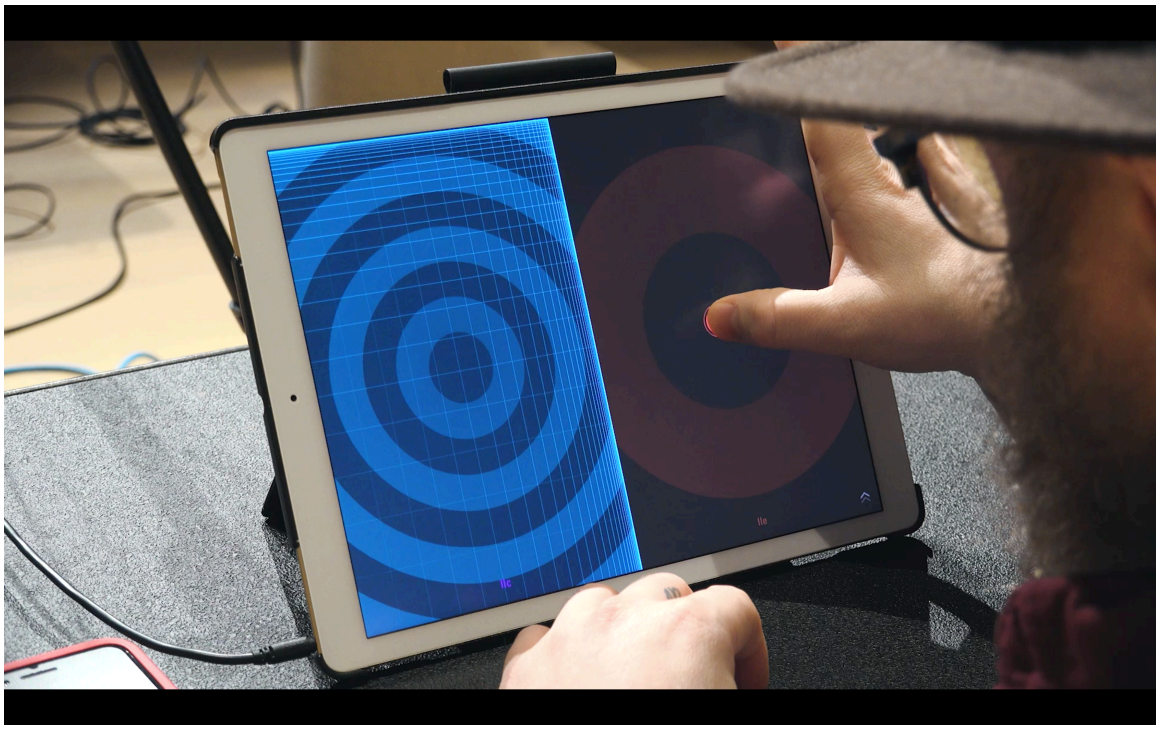


Figure 26: Screenshot of the iPad Pro running TC-Data during performance

generation during the piece, and the right-hand side producing two data streams that controlled the audio effects I used after the sounds had been recorded and set in place as loops. The data streams from the left-hand side of the screen were generated from the duration of how long one of my fingers was touching the screen (or how long the touch was 'alive'), the X-axis position of my finger, the Y-axis position of my finger, the rotation of my finger around the center of the screen section, and the total distance travelled by my finger. The right-hand side of the screen generated two data streams: the average distance and position of my finger in relation to the center of the screen. The data streams remained consistent through each of *Multios*' four sections, though the data streams were assigned to different musical parameters in each section.

MIDI data generated by TC-Data was routed into one of two apps: iDensity (in the case of the left-hand side of the performative screen) or AUM (in the case of the right-hand side of the performative screen) (see Figure 27). In iDensity, the touch 'alive' (how long my finger was in contact with the screen) data stream was assigned to the playing and stopping of the each sound, so that as long as my finger was in physical contact with the screen, the assigned sound would be heard. The X-axis position data stream was assigned to both the currently played portion of the assigned sound and the panning of the audio within the stereo field. The data generated by my finger's rotation around the center of the screen was assigned to the granular density parameter of each sound in iDensity as it was played. As my finger moved around and across the iPad's screen, TC-Data generated a data stream that corresponded to the distance travelled by

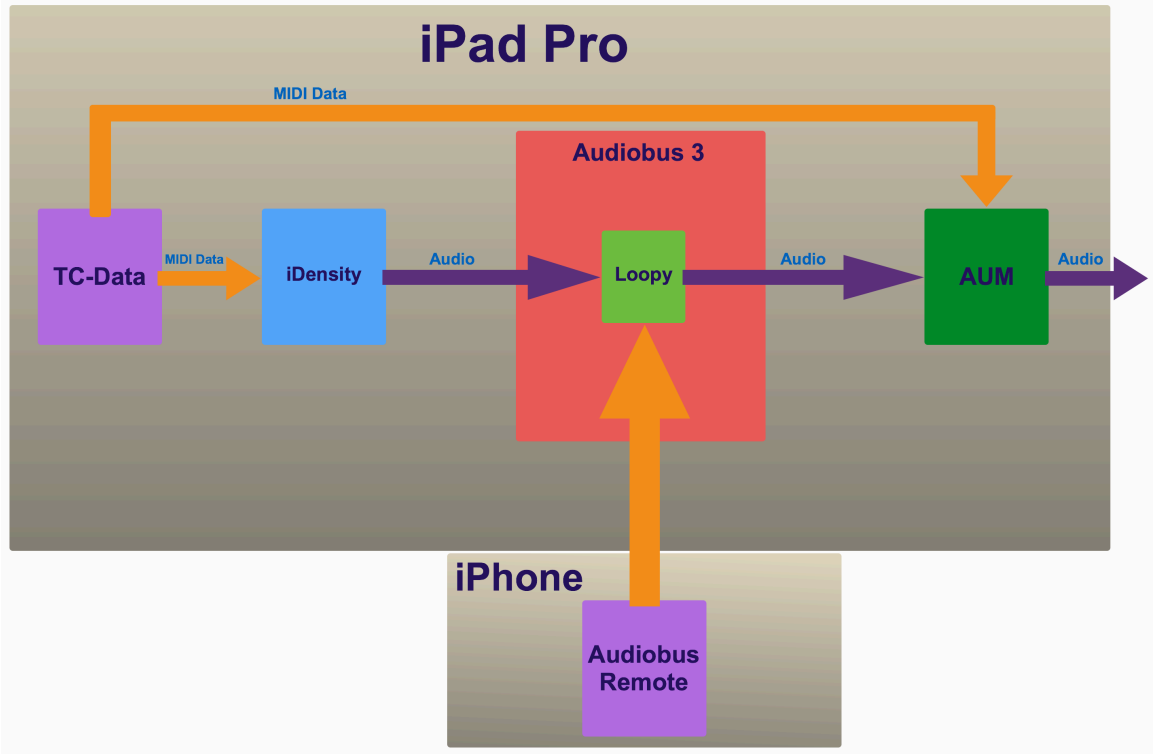


Figure 27: Overview of my DDI for *Multios*

my finger, which was assigned to the grain duration parameter of each sound in iDensity. Finally, the Y-axis position data stream was assigned to the amplitude of each sound, allowing me to dynamically control the volume of the piece in real time. The fourth section of the piece only utilized three of the aforementioned data streams: touch 'alive,' X-axis position, and Y-axis position.

The right-hand section of the TC-Data performance screen controlled the audio effects plugins in AUM that I applied after each section's sound was looped. I used AUM as the final stage in the chain of apps that made up my DDI's software environment. "AUM is a flexible audio mixer, plugin host, recorder, and connection hub" (AUM - Audio Mixer, n.d.). Within AUM were individual iOS plugins (AUv3 Audio Units) that I used throughout each section of the piece. I assigned the average distance to center data stream to the reverb mix parameter of a granulator plugin, and the rotation around center data stream to the dry/wet mix of a rhythmic gating plugin. The dry/wet mix of a second instance of the rhythmic gating plugin (with a different customized rhythmic preset) was controlled by the average distance to center data stream, and the dry/wet mix of a reverb plugin was controlled by the rotation around center data stream. The dry/wet mix of a feedback delay network plugin was controlled by the average distance to center data stream, and the maximum distance parameter was controlled by the rotation around center data stream. The dry/wet mix of an audio mangler plugin was controlled by the average distance to center data stream, and the dry/wet mix of another instance of the reverb plugin was controlled by the rotation around center data stream. Finally, I

controlled the recording, timing, and fade-out of each loop in *Multios* with the Audiobus Remote on my iPhone.

Audiobus is an audio-routing app that let me route the original audio generated by iDensity into Loopy (see Figure 28). Loopy could then record, store, and playback up to twelve discrete loops, giving me up to three loops to work with for each of the four sections of the piece. Each loop was assigned a specific channel in Audiobus, which acted as the central audio-routing hub of the software component of my DDI. I routed each of the twelve individual Loopy channels into AUM, where I was able to group the individual Loopy channels into sub-mixes of three loops, with each sub-mix representing one of the four sections of the piece (see Figure 29). AUM's capacity to create sub-mixes was essential because it allowed me to simultaneously assign the plugins to multiple loops, which created a distinct sonic signature for each of the four sections.

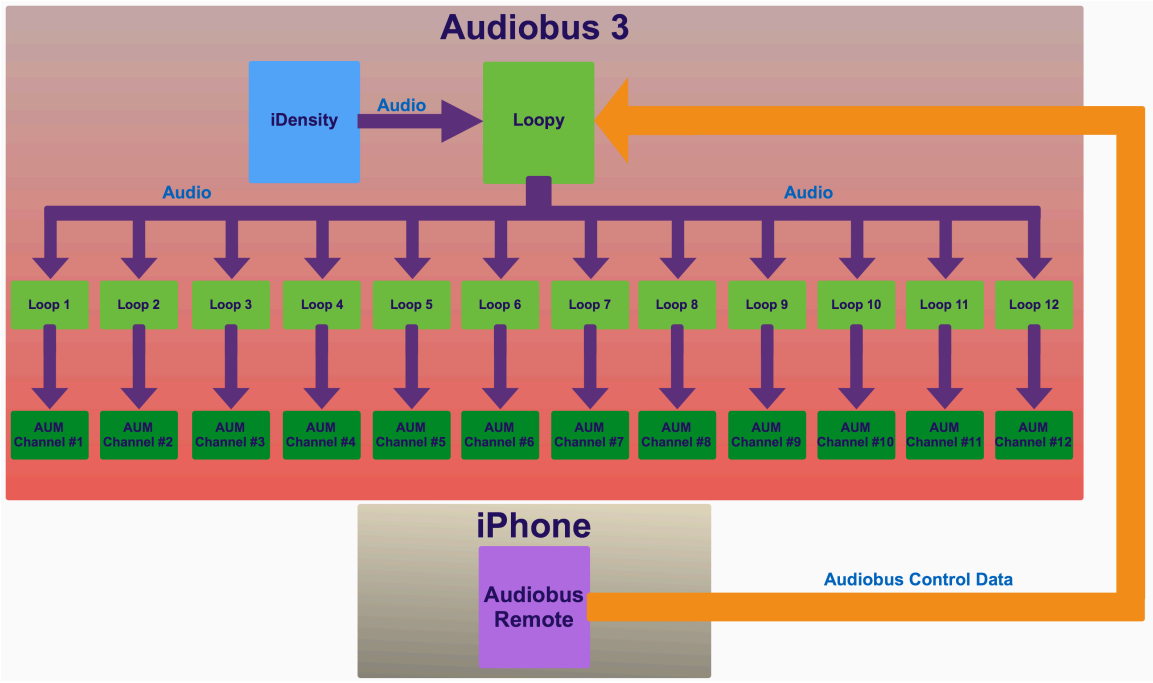


Figure 28: Diagram of audio routing within Audiobus 3

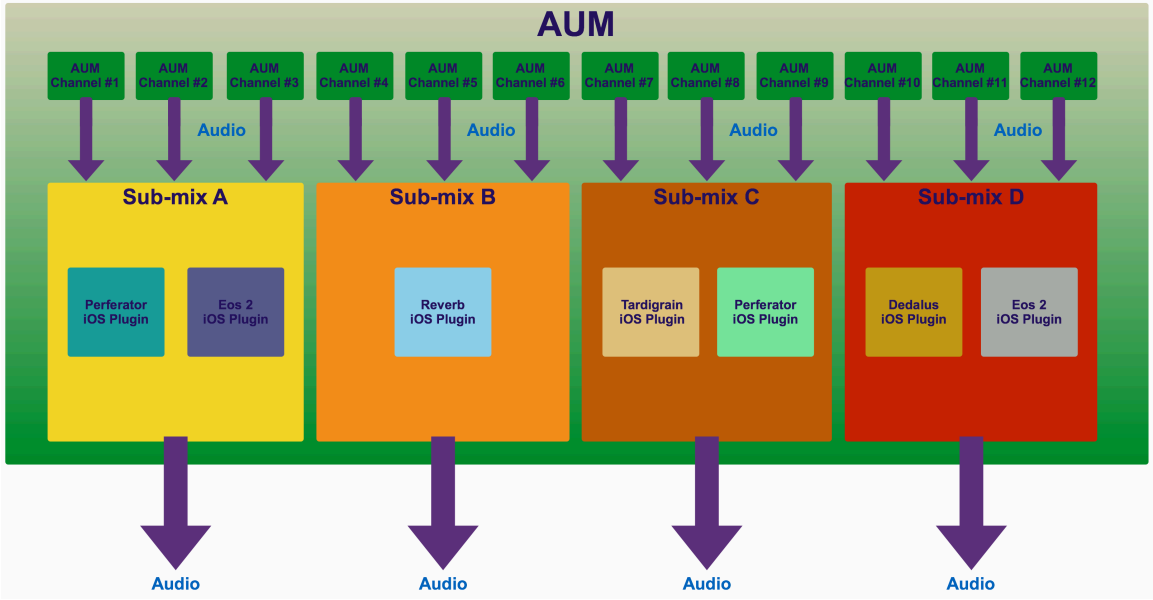


Figure 29: Diagram of audio routing within AUM

Sound Design

Each of the four sounds that I used in *Multios* were originally based on recordings of my shruti box. These four specific recordings were chosen because of their sonic variety and the way they worked together and complimented each other. iDensity was the nexus of the sound generation for the piece and is comprised of six discrete audio channels each possessing its own controls and parameters. I used four of the six audio channels for *Multios* with each channel containing one of the four original samples. I designed the first sound to be a subtle, rhythmic introduction to the soundscape of *Multios* (occurring first at 0:36 in the fifth supplemental video). The original sample was a quasi-melodic recording containing different pitches of the shruti box over an underlying drone. Processing the sound through a granular algorithm that specified short grain durations using a 'Hanning'[20] grain envelope created the sound's sonic signature.

[20] "Hann (Hanning) Window - MATLAB Hann," accessed April 11, 2020, https://www.mathworks.com/help/signal/ref/hann.html.

The second of the sample-based sounds (occurring first at 3:21 in the fifth supplemental video) was created from a sample centered around a chord of droning pitches being processed through a granular algorithm that specified longer grains, higher cloud density, a 'Barlett'[21] grain envelope, and delays for both the right and left audio channels. The third sound (occurring first at 5:32 in the fifth supplemental video) started as another droning sample, but with more pronounced overtones that were created through the combination of pitches being played at the time of recording. The sample was processed in manner similar to the previous sample, but the overtones heard in this sample yielded a much different sonic signature after processing. The fourth and final sound (occurring first at 7:33 in the fifth supplemental video) featured a melodic droning sample in a lower register that provided the bass-heavy melodic structure that I used to complete the overall soundscape. I processed the sample through a granular algorithm that specified longer grains, higher cloud density, and a 'Barlett' grain envelope.

Once each of the four sounds were recorded, looped, and playing, they were routed into their respective sub-mixes in AUM and assigned their plugins. I wanted each sub-mix's plugin(s) to enhance and transform the original sounds, adding complexity and variety to the soundscape as the piece evolved over time. The plugins that I used for the first sub-mix were the granulator and rhythmic gating plugins (occurring at 2:22 in the fifth supplemental video). The combination of both plugins heard together granulated the original sound while adding rhythm and depth. I used another instance of the rhythmic gating plugin for the second sub-mix, in combination with the reverb plugin (occurring at

[21] "Appendix F. Window Functions," accessed April 11, 2020, http://www.csounds.com/manual/html/MiscWindows.html.

4:42 in the fifth supplemental video). Together, these two plugins added a noticeable rhythmic layer to the soundscape while also adding more space and depth via the reverb. The third sub-mix (occurring at 6:41 in the fifth supplemental video) used the feedback delay network plugin, which modulated and transformed the pitch and timbre of the original sound through a series of built-in feedback delay networks. Finally, the fourth sub-mix (occurring at 8:36 in the fifth supplemental video) used a second instance of the reverb plugin that added a greater sense of space and size in combination with the audio mangler plugin that added distortion and overdrive to the sound. Because this final section of the piece featured the most bass-heavy sound, I magnified the bass frequencies' prominence through the use of this distortion and reverb.

Composition and Performance

Multios begins with a relatively quiet first section, introducing a subtle polyrhythm comprised of short, granulated loops that sets the tone for the rest of the piece. The looped granular sounds create a sense of sonic motion due to their short, repetitive nature, and lay the timbral foundation for the remainder of the composition. Each section builds upon the prior section, creating a tapestry of sound as each loop is set into place. The loops of the first section function like that first thread in a tapestry; small and thin on their own, but when heard together, produce a sonic richness. Once each loop was recorded and playing, I began to control the audio plugins using the right-hand side of the iPad's screen. The plugins for this section blurred and blended the short grains together, adding a hint of droning to the overall soundscape.

The second section introduced a longer, brighter, and louder sound to the sonic tapestry (occurring at 3:21 in the fifth supplemental video). The sound for the second section reflects the shruti box's capacity to serve as a drone-based instrument. Unlike the stuttering, subtle, rhythmic quality of the first section's sound, the sound for the second section is clearly heard whenever my finger is in contact with the iPad's screen. As I recorded each loop, another droning layer of sound was added to the sound field, adding sonic variation and new timbral qualities to the musical tapestry that I wove. As the last loop was recorded, I started to control the plugins assigned to this section. These plugins added a more pronounced rhythmic layer for the remainder of *Multios*, helping me establish a pulse onto which the audience and I can lock.

The sound of the third section was a result from a blending of rhythm and drone, acting as an amalgamation of the first two sections of the piece (occurring at 5:32 in the fifth supplemental video). If I held my finger on the screen of the iPad, the sound would continuously play, but I chose to perform this section in a more abrupt fashion, using a slashing-like motion to create bursts of sound instead of a single droning sound. This performative action sets this section apart visually and aurally. I recorded each of the loops for this section while using the slashing motion, adding both drone and rhythmic elements to the tapestry. The plugin I utilized for this section then modulated the pitch of the loops and added a layer of reverb to separate the sonic qualities of the section from the previous two sections.

The final section of the piece established the lower frequency spectrum and added the first true melodic element into the musical tapestry (occurring at 7:33 in the fifth

supplemental video). After slowly raising the amplitude of the sound, I focused on the droning, bass-heavy, melodic quality of the original sample. My finger rarely left the iPad's screen during this section until I finished recording the final loop. Once the melody of the final loop had been established, I engaged the plugins that I assigned to the right-hand side of the screen. The audio effect that was generated by the plugins was centered around distortion and overdrive, gradually adding depth and amplitude to the final loop. I introduced the effect very slowly over time, marking a departure from the previous three sections. Reaching the full effect of the plugin completed the sonic tapestry and journey of *Multios*. I then began to gradually remove one loop at a time from the soundscape, ending the piece with only a single rhythmic loop from the second section.

Using the iPad Pro as my main performance interface created a unique set of challenges regarding the observability of my performance to an audience. Because an audience was not always going to be able to see the iPad from my point of view, they would not always be able to observe my interactions with the screen during a performance. I was acutely aware of the need to express my physical and performative actions in a way that was observable by the audience. This observability allowed an audience to better follow my performance while hearing the piece unfold. To that end, each of my performative actions moved across the iPad's screen to the maximum degree possible, showing the audience the movement of my arm and hand even though they may not necessarily see the screen.

III.6 ***ArcMorph***

Introduction

The sixth piece of my DPD is called *ArcMorph* and is written for the Arc and Sensel Morph, Max, and Ableton. Both the Morph and the Arc are highly customizable and adaptable interfaces. The Arc consists simply of two endless encoders, each with its own set of sixty-four LED lights that can be used to visualize the position of each dial. The Morph provides button-type control. The original sounds for *ArcMorph* came from shruti box recordings. Utilizing improvised loops created in real time I was able to explore the nuances and minutiae of the shruti box through the use of granular synthesis.

Instrumental Forces

The Arc was designed and built by Brian Crabtree and Kelli Cain, the same team who created and built the Monome (see Figure 30). Similarly to the Monome, the Arc simply sends and receives OSC data that I customized inside of Max. However, the Arc instead consists of two endless encoders (implemented as large metal dials), each with their own set of individually addressable LED lights that could be used to visualize the position of each dial.

> A rotary knob used to digitally control a parameter in a software or hardware device. An endless encoder does not have a 'stop' point, it simply keeps turning until the user stops moving it. This allows an endless encoder to support a very wide range of parameter values or very fine control, as the physical range of the knob rotation is not limited (Sweetwater, n.d.).

The Sensel Morph (see Figure 31) completed my performance interface by adding button and slider control. "The Sensel Morph is a multi-touch, pressure sensitive and

Figure 30: The Arc 2 (Monome, n.d.)

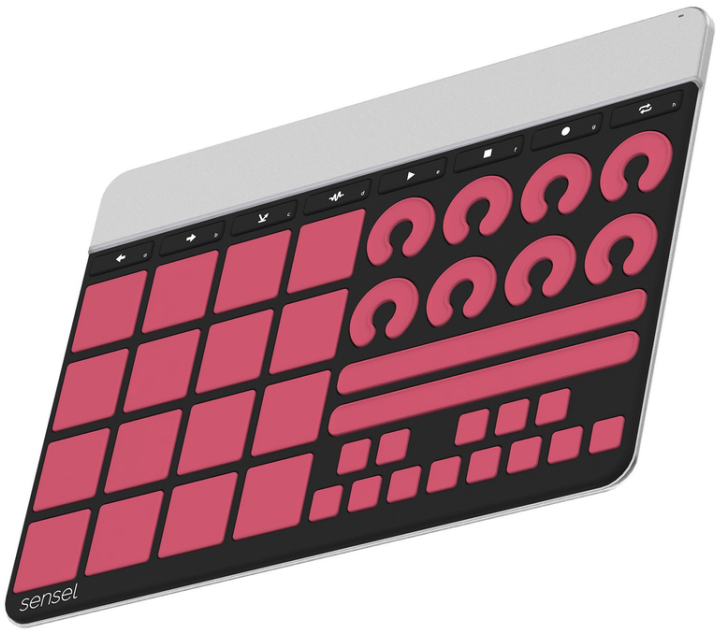


Figure 31: The Sensel Morph with Music Production Overlay (Sensel, n.d.)

reconfigurable control surface" (Sensel Morph Documentation, n.d.). Associated with the reconfigurable control surface are a set of overlays that provide visual articulation about how the surface has been configured. I used the Music Production overlay for *ArcMorph*, comprised of a grid of sixteen buttons/pads, eight rotary-type faders, two linear faders, and an octave's worth of smaller buttons arranged in the manner of a traditional piano keyboard.

The data from each of the Arc's dials was brought into Max and split into two main data streams; one that was directed into Ableton as MIDI data, and one that was routed to control the position and intensity of each dials' LED lights. Because the Arc's encoders are endless, scaling the data generated by each dial was not as easy as using a single 'scale' object in Max. The data enters Max as OSC data that is then scaled to floating-point numbers between 1.0 and -1.0. In order to create a usable data stream with a range of values between 0 and 127, I used a Max object called 'accum'. The 'accum' object stores a value and outputs it, and then incrementally changes its output (and stored value) upward or downward based on the new value input into the 'accum's' inlets and whether addition or multiplication is used to execute its calculation. When input to the 'modulo' object with an argument of 128, I was able to constrain the output range to between -127 and 0 and 0 and 127. Each dial's data streams were assigned to two musical parameters within Ableton, with the Sensel Morph controlling the switching mechanism between parameters. The parameters that the dials controlled were the scrubbing of the samples and the control of an audio effect that modulated and transformed the sound.

Using the Sensel App that comes with the Morph I created a custom template that was able to send MIDI data into Max (see Figure 32). Because the Morph was configured to send MIDI data, less data mapping was needed within Max. Instead, I was able to simply pick which data streams needed to be assigned to which outputs to create an effective control system. I assigned the top two rows of the main button/pads to specific notes of the bass instrument, and the bottom two rows to the Looper controls. Two of the rotary-like faders controlled the amplitude of the bass instrument and Ableton's master output, and one of the linear faders controlled the audio effect on the master output. Finally, the smaller, keyboard-like buttons on the bottom of the overlay were mapped to the switching mechanism I designed in Max so that the data streams from each of the Arc's dials could switch between and among control of the parameters within Ableton.

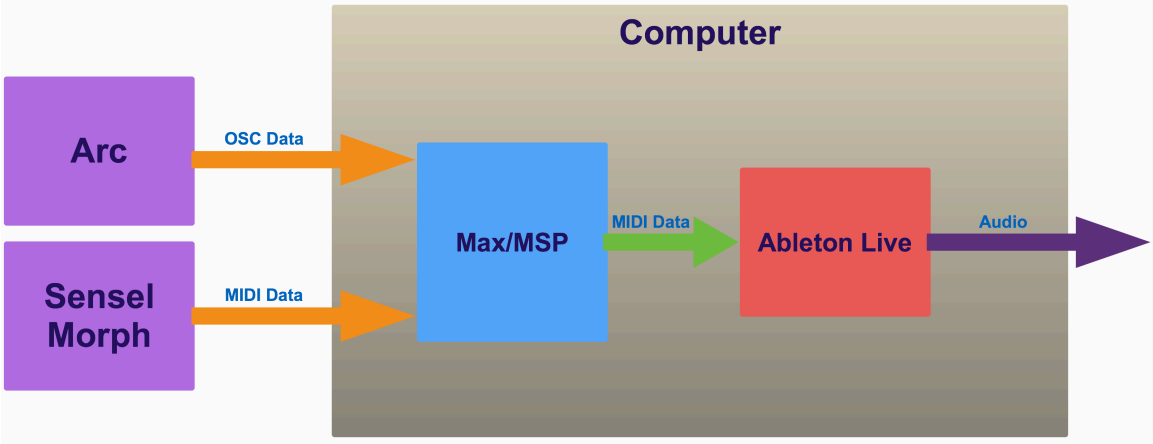


Figure 32: Overview of my DDI for *ArcMorph*

Sound Design

All of the sounds heard in *ArcMorph* originated as samples from my shruti box. My overall conceptualization of the piece was for it to be a kind of 'zooming in' on the sounds of the shruti box, exploring the sounds of the instrument on a micro-level and

uncovering the minutia and complexities that are hidden within its sound. I wanted the piece to be a slowly unfolding exploration of the sonic characteristics of the shruti box, but also to maintain an aural connection to the original nature of the shruti box.

Ableton served as the main synthesis environment for this piece and generated all of the audio. I used a Max for Live granulator device as the main software instrument. I used two audio samples as the main source of sound material, each featuring short melodic phrases over droning chords. The two samples were different from one another and had a different device setting. This difference aided me in creating sounds that had distinctive voices within the sound field. The samples were granulated each time I turned one of the dials (occurring at 0:35 in the sixth supplemental video). I was then able to record loops at different points of each sample. Each loop was also sent through another Max for Live granular device, further granulating and transforming the audio.

Once the final loop was recorded and playing back, I used a Max for Live reverb device to further transform the audio from each channel, propelling the piece forward into the next section (occurring at 4:35 in the sixth supplemental video). After the effects of the reverb device were fully introduced into the sound field, I utilized two other effects devices: a Max for Live rhythmic gating device and an Ableton resonator device. The rhythmic gating and stuttering device added a layer of rhythm to the sound field, while the resonator device emphasized specific overtones, creating a more complex, richer, and resonant sound. Once I had brought in the rhythmic gating and resonator devices (occurring at 5:51 in the sixth supplemental video), I began playing the individual bass pitches (occurring at 7:15 in the sixth supplemental video). The software instrument I

used to produce these pitches sounded like a cross between a Hammond B3 organ and a Moog Voyager synthesizer.

Composition and Performance

ArcMorph was composed in C# major, and is loosely made up of three sections. Because I wanted a gentle, flowing character to the piece, I blur the lines between each section to bolster the connection to the droning nature of the original sound source. The Arc especially lends itself to a slow, gradual performance due to the continuous control the dials offer, as do the faders on the Sensel Morph. These controls helped me maintain a performative connection to the droning nature of the shruti box.

The first section of *ArcMorph* focused on the introduction and looping of the sounds that comprised the foundation of the piece (occurring at 0:35 in the sixth supplemental video). I slowly introduced each sound layer through my interactions with the Arc, alternating between the two dials with each new layer of sound that was introduced. I built the layers and loops from low to high in terms of frequency, with each new inclusion adding another layer of sound. As I rotated each dial (reflected by the LEDs), I honed in on a portion of the sample that I looped using the buttons on the Morph, forming that particular layer of sound. After four layers of sound were playing back, I ended the section by moving my performative attention to one of the linear faders on the Morph. Over the course of about a minute, I gradually transformed the existing soundscape, creating a new timbral area of sonic exploration to begin the second section of the piece.

The second section is the shortest of the three sections, and I moved from the Morph interface back to the dials on the Arc (occurring at 5:51 in the sixth supplemental video). I switched their functionality to control a new set of audio effects that were the focus of the second section, where each dial controlled a different audio effect. The left-hand dial added the first rhythmic element to the piece and allowed me to introduce a faster, more pronounced performative interaction with the Arc than was previously seen in the first section. The right-hand dial then reinforced the rhythm and added another timbre into the sound field.

As I continued to rotate the right-hand dial on the Arc, I began the third section of the piece by introducing a bass line using the top two rows of larger buttons on the Morph (occurring at 7:15 in the sixth supplemental video). By continuing my interactions with the Arc while I began playing the bass melody, I was again able to make the sectional transition seamless. The melody that I played was improvised with each new performance of *ArcMorph*, but broadly focused on C#, F#, G#, and A#. I played the entirety of the bass melody with my right hand so that my left could continue playing the Arc throughout the section, uniting the two interfaces into a single, cohesive conceptual unit. I manually faded out the bass as the melody concluded, ushering in the end of the piece. Once the bass was no longer heard, I slowly removed the two audio effects that were controlled by each dial, and then slowly removed the audio effect controlled by the linear fader on the Morph, returning to the original layers of sound that I had laid down during the first section. The piece concluded by gradually fading out the master audio, ending the piece as slowly as it began.

III.7 ***Kaurios***

Introduction

The focal point of a performance of *Kaurios* is a custom-built interface that I designed and fabricated. This custom interface is combined with Max and Ableton to form the complete DDI that I used to perform the composition. Kaurios (the interface) is comprised of two wooden pieces, one for each hand. Each piece is the mirror image of the other and contains identical sets of wireless electronics, with the exception of an extra distance sensor in the right-hand piece. Ergonomics and performability played large roles in the overall design of the interface. I wanted *Kaurios* to incorporate as many of the ideas, concepts, and techniques that I learned during my time as a student of music technology and digital arts.

Instrumental Forces

The design and creation of Kaurios (the interface) was born of my love for the Monome. The portability, versatility, and minimalistic qualities of the Monome made it one of my favorite and most-used interfaces. However, as I progressed through my studies at the University of Oregon, I garnered the knowledge and skills to design and fabricate my own interfaces and DDIs. Because I designed and built Kaurios from raw materials and unassembled components, my initial conceptions were simple. With the help of John Park, I was able to envision a performance interface that was functional, ergonomic, and intuitive. The central elements of my conception concentrated on smallness of size and portability. I also strove to engage the interface with an economy of

motion. To achieve these goals, John and I developed the idea to split the interface into two pieces, one for each hand. I experimented with different sizes and shapes for each stone (as I call them), and ultimately settled on the final form after 3D printing a prototype (see Figure 33) that comfortably and functionally fit my hands. Once the size and shape were finalized I was able to run my wood through a computer numerical control (CNC) router and create the two bodies into which the sensors would be installed (see Figure 33). After sanding and smoothing the surface of the wood, I began installing, wiring, and soldering my sensors into the wooden forms.

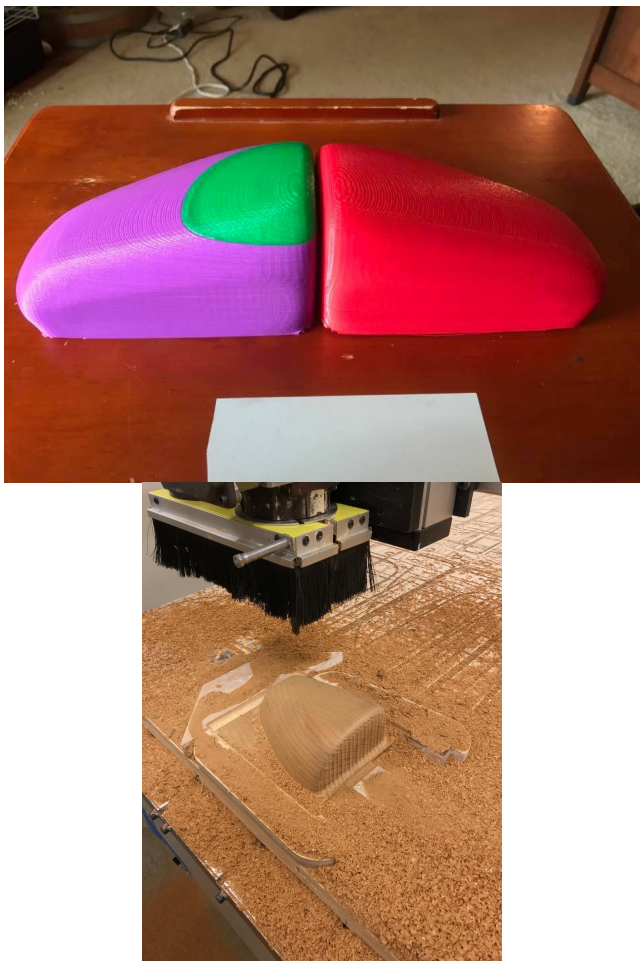


Figure 33: 3D prototype of the Kaurios interface (above), CNC routing of wood blocks (below)

The wood I used in the construction of Kaurios is called Ancient Kauri. Originating from one specific location in northern New Zealand, Ancient Kauri is uncommonly rare and known the world over as the oldest workable wood on the planet (see Figure 34). The actual blocks I used were carbon dated at between 30,000 and 50,000 years old, predating the last Ice Age by more than 20,000 years. The Kauri forest was felled by completely natural forces (possibly an ancient tsunami), and was preserved just below ground level in the water of a peat bog. The bog turned out to be the perfect resting place for these giant trees, sealing the wood from the air and creating the perfect cocoon to prevent the wood from petrifying or turning into coal. I happened upon the blocks by complete luck, discovering them at a local wood shop. By incorporating Ancient Kauri wood in the creation of Kaurios, I was able to turn something incredibly ancient and organic into a brand new DDI that highlights the technology of the present.

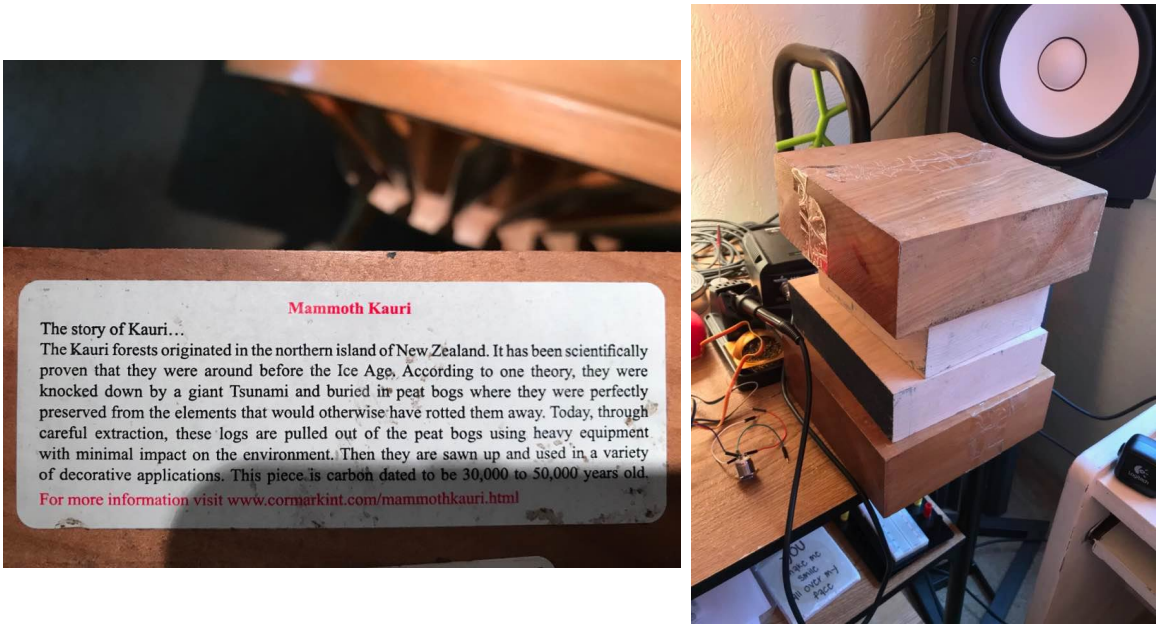


Figure 34: Raw blocks of Ancient Kauri wood that I used for the interface

The button-centric characteristics of the Monome inspired my decision to utilize buttons as one of the focal points of the Kaurios interface. I chose a heavy-duty momentary metal push button from Adafruit that featured RGB LED rings around the centers of the buttons to enhance visual feedback. To support and augment the performative options that Kaurios offered, I added an array of sensors to each stone comprised of a 9DoF motion sensor, a touch-fader, a small joystick, and a distance sensor (in the right-hand stone only) (see Figure 35).

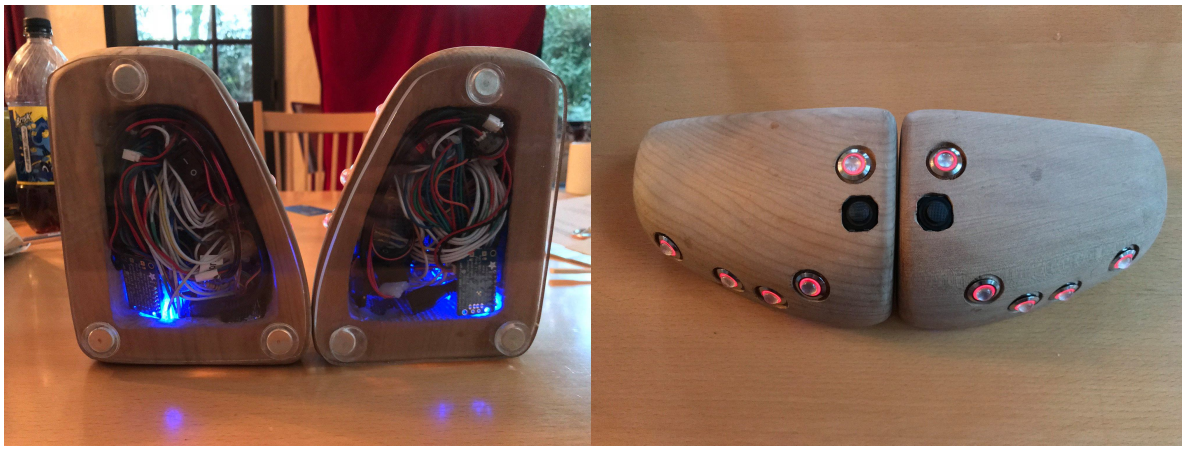


Figure 35: The finished Kaurios interface

Using the Feather M0's BLE (Bluetooth Low Energy) capabilities I was able to make the Kaurios interface wireless. BLE MIDI allows each stone to be connected to any computer as a Bluetooth MIDI device. BLE MIDI also allowed me to transfer MIDI data with very low latency into Max. Because the data streams that were generated by Kaurios were already being sent as MIDI data into Max, the Max patch functioned primarily as a calibration hub before routing the data into Ableton (see Figure 36).

I began the data mapping by creating a calibration routine for the motion sensors because their data streams vary based on where I am physically located on the planet. I

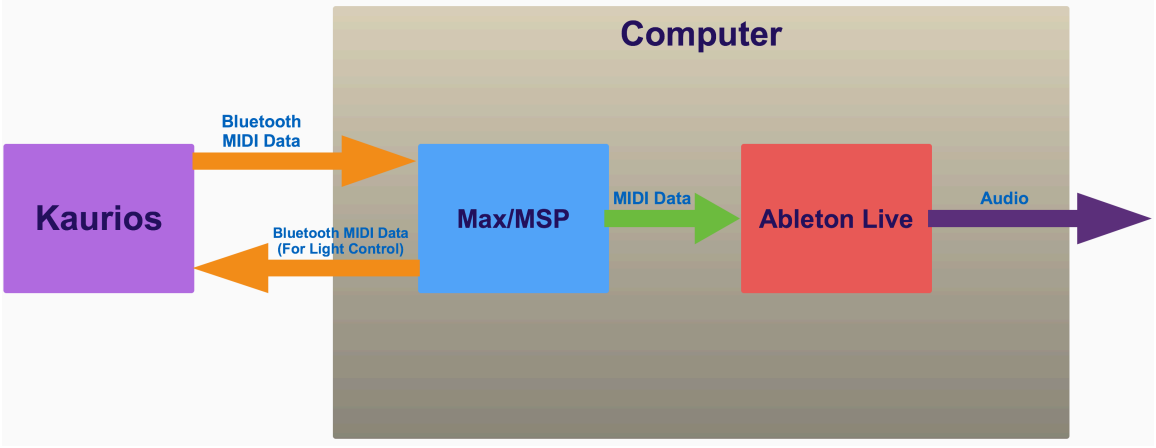


Figure 36: Overview of my DDI for *Kaurios*

chose to utilize quaternion data due to its high accuracy in regards to spatial orientation. This decision made it possible for me to acquire sensor data from four discrete streams, 'W,' 'X,' 'Y,' and 'Z.' After calibration, I assigned the data streams to specific parameters in Ableton (depending on which data stream is most usable in that particular performance). The distance sensor and touch-faders only required minor amounts of scaling after which they were assigned to their specific parameters in Ableton. The joysticks on each stone are small and relatively unobservable to an audience, so instead of using them as continuous controllers I decided to turn them into 4-pole switches through the implementation of thresholding. By giving myself four distinct switches per stone (up, down, left, and right), I then had the ability to switch between sections of the piece and create and control my loops without having to use any of the five main buttons.

I programmed and uploaded to the microcontroller software that changed the color of each of the five primary buttons on each of the stones depending on which button is being pressed. For instance, when I pushed the thumb button, that stone's buttons all turned blue, and when I pushed the middle finger button, the LED rings turned purple. I

use the same color scheme for both stones. Additionally, each button was mapped to a specific note in Ableton, providing me the ability to play the chords and melodies that I composed. Each of the metal buttons was mapped to a specific note in the key of F# major. The left-hand stone's notes were (in ascending order) F#2, A#2, B2, C#2, and D#3. For the right-hand stone, the notes were F#3, G#3, B3, C#4, and E#4. Each of the touch-faders were mapped to control a specific audio effect. Because the touch-faders were affixed to the 'back' of each stone, out of sight to the audience, I also mapped this data to control the LEDs on their respective stones to provide additional visual information about my performative actions. The distance sensor was mapped to both an audio effect and the LEDs on both stones, also to highlight my performative actions.

Sound Design

All of the sounds heard in *Kaurios* again originated from my shruti box and were crafted into the soundscape of the piece within Ableton. I began with a seven-second audio sample of my shruti box playing a single F#, that I brought into a Max for Live instrument and processed with a granular processing algorithm (one instance of the instrument per stone). Within that granulating software instrument, I transformed the original audio sample into the referential sound and note that each stone triggered throughout the piece. Then, as I progressed through the performance of the piece, these initial referential sounds were further manipulated and transformed according to each stone's particular effects chain.

I built similar, but subtly different audio effects chains for each stone, resulting in two distinct sonic characters (see Figure 37). Because the sound controlled by my right hand differed in terms of frequency and timbre from the sound controlled by my left, a richer, more complex soundscape emerged than if both hands produced identical sounds. The notes assigned to the left-hand stone occupied the lower part of the frequency spectrum. After boosting the lower frequencies with equalization, I inserted a Max for Live rhythmic sequencer device that chopped up the sound and rhythmically re-articulated it under the control of the motion sensor contained in the left-hand stone. Next in the audio effects chain I placed a lowpass filter whose cutoff frequency was controlled by another axis of the left-hand stone's motion sensor. Next, I inserted Ableton's Looper effect, which allowed me to record and control the loops that built the underlying structure of the piece under the control of the 4-pole switch from the joystick. Another Max for Live rhythmic sequencer device followed the Looper, both rhythmically re-articulating the sound and filtering it under the control of the data from the distance

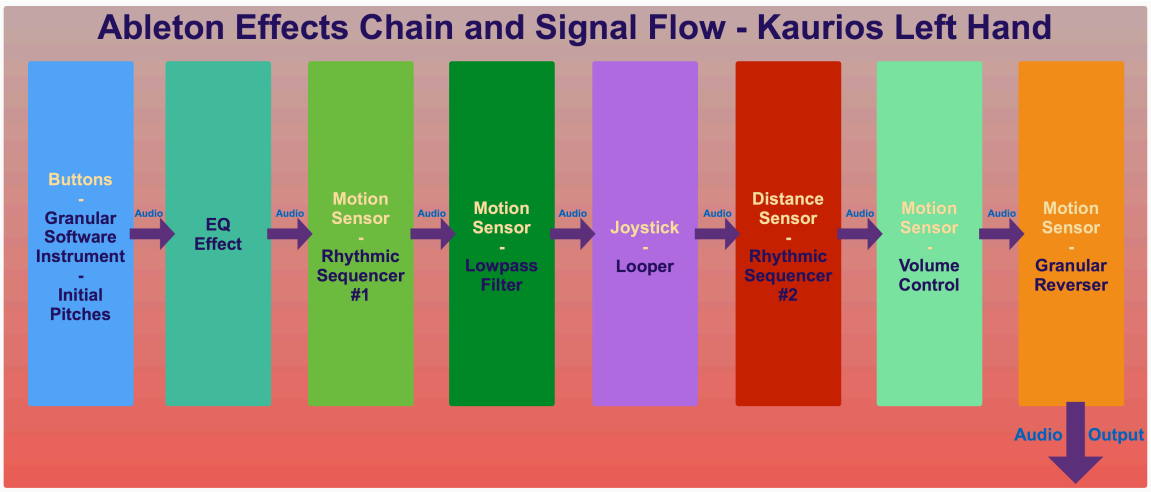


Figure 37: Effects chain and signal flow of Kaurios' left hand

sensor. Finally, in the last section of the piece, I combined an Ableton volume control device and a Max for Live granular reverser device that I mapped to two of the motion sensor's axes, which facilitated the introduction of contrasting performance techniques that led to new sonic outcomes.

The right-hand stone's audio effects chain was very similar to the left, but differed in a few key aspects (see Figure 38). First, the notes assigned to the buttons for the right hand were an octave higher than the left, so they did not require equalization in the same way as the bass tones of the left hand had. Additionally, because the right-hand stone's notes were higher, I instead utilized a highpass filter, allowing more of the upper range of the frequency spectrum to become enunciated. The right-hand stone's motion sensor also controlled both the highpass filter and the rhythmic sequencer. The Looper effect and second Max for Live rhythmic sequencer device came next in the audio effects chain and were controlled by the joystick-switch and distance sensor. I used the same volume effect and Max for Live granular reverser device as the left hand, but also added an overdrive/

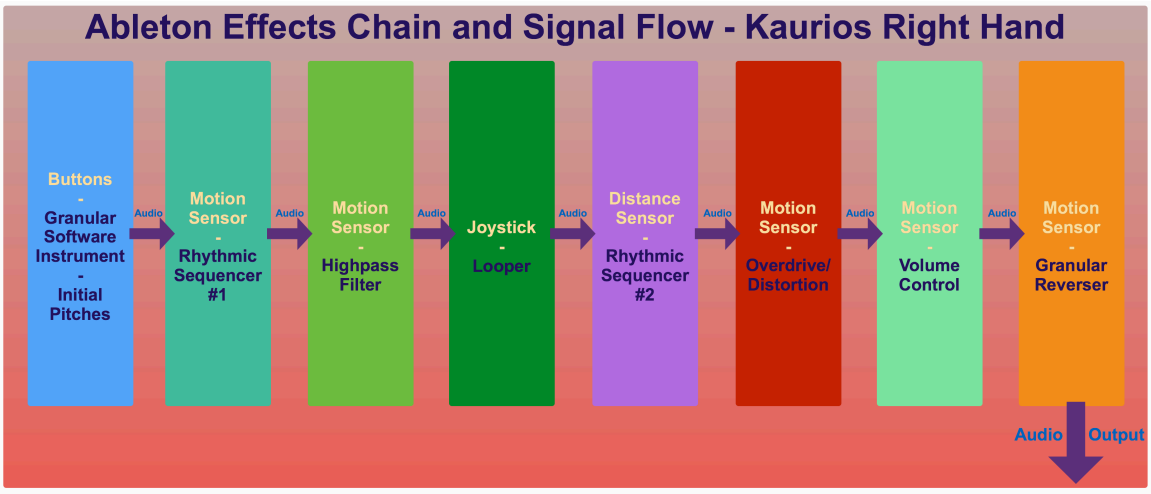


Figure 38: Effects chain and signal flow of Kaurios' right hand

distortion effect that further enhanced the soundscape. All three of the last effects in the audio effects chain were also controlled by two of the axes of the left-hand stone's motion sensor. Finally, after the last main section, I introduced a new audio effects chain for the epilogue on a return track in Ableton, consisting of three different convolution reverb devices that I ran in series. Because each reverb unit was input to the next, the sound that emerged at the end of the chain was substantially transformed.

Composition and Performance

Kaurios consists of three main sections, with a short epilogue. Each section highlights new sonic material and a different mode of performative interaction. The first section began with an introduction, slowly revealing the notes, sounds, and melodies around which the piece revolves (occurring at 0:39 in the seventh supplemental video). I began with the tonic of the piece on a low F# in my left hand, gradually moving upwards through the other notes of that stone. Once I introduced each of the left-hand notes, I then moved to the right-hand stone, introducing the notes in a higher octave than the other stone (occurring at 1:52 in the seventh supplemental video). Once each individual note had been introduced I played both stones simultaneously, creating the first chordal material of the piece (occurring at 2:44 in the seventh supplemental video). After I established an improvised melody, I end the introduction of the first section with both hands sustaining the tonic F#. I then utilized a gating function that I devised to open the gate (triggered by one of the poles of my joystick switch) that allowed two of the motion sensor's data streams to control the dry/wet parameter of the first rhythmic sequencer and the cutoff frequency of the highpass filter. The two new effects transformed the sound

being controlled by my right hand and allowed me to engage the motion sensor for the first time in the piece (occurring at 3:45 in the seventh supplemental video). Strategically thinking, my use of the motion sensors in this section was minimal, and I constrained the movement of each stone to allow for me to better build on these movements during the final section of the piece. I also began to record the loops, recording and looping two or three melodic phrases that continue to repeat and lay the foundation for the rest of the piece. The bass tones controlled by my left hand continued during the looping of the right, providing continuity as the texture of the music began to change. Once the loops controlled by my right hand had been set, I completed the melody I was playing with my left hand and recorded a loop of the low F# drone. I then used the gating function that allowed the motion sensor's data to control the rhythmic sequencer and lowpass filter, and recorded additional loops of the bass (occurring at 4:55 in the seventh supplemental video). I completed the first section of the piece with the two stones flush against one another, setting the stage for the next section.

In the second section I focused on using the distance sensors embedded in the interface (occurring at 5:30 in the seventh supplemental video). The stones began this section physically placed against one another. As I slowly began to pull away the stones from each other the sounds were gradually transformed and gave way to new rhythms and filtered sounds. Because the distance sensor is unidirectional, the only performative action that was available involved changing the distance of a stone in relation to the distance sensor. Therefore, to achieve an interesting and thoughtful performance for this section, I varied the speed, distance, and location of each stone's movement while still

keeping them pointed at one another. I did this so that the sensor could measure distance accurately and my movement would yield visually and aurally interesting results.

The third section featured the motion sensors in each stone, but in a visually and aurally more robust fashion than in the first section (occurring at 8:10 in the seventh supplemental video). I began by moving the stones in the same manner as the first section, introducing new sonic material via a new chain of audio effects. Once the new effects had been introduced, I lifted the stones completely off of the playing surface, extending my physical motion considerably as compared to my movements of the first section. Enlarging the space in which I performed not only extended my palette of performative actions, but also allowed me to develop a more nuanced performance within this larger space. I was able to explore more fully the minutia of the data being generated by the motion sensors and its effect on the sounds. Because of the heavier, denser sounds and effects that I chose for this section, I subconsciously found myself moving the stones as though they had an added *weight*; pushing, pulling, and lifting them through the air as opposed to simply moving them without any perceived sense of resistance or effort. I think of this phenomenon as a sort of 'performative feedback loop,' where the music and sounds that I composed begin informing and dictating how my performative actions respond to the music, and vice versa where my interactions with the interface dictate how I find myself shaping and performing that music, all in real time.

I abruptly ended the third section by engaging the touch-faders as the stones were in the air, quickly bringing them back down to the playing surface at the same moment that I first placed my fingers on the touch-faders. Not only was the physical movement of

this motion jarring because of how quickly I changed my movement style, but the change in sound was also dramatic because of how abruptly the sound changed. As soon as I placed my fingers on the touch-faders, the three convolution reverb devices began their operation, transforming the soundscape into an ethereal, light, other-worldly sonic realm that had not previously been heard. However, I wanted to emphasize the idea of having embarked upon a musical journey, so I presented again previously heard sounds back into the soundscape under the control of my fingers that moved up and down the touch-faders to add a reminder to the listener of where we had musically travelled. The piece ended as I slowly faded out the music, having wholly arrived at a new musical destination.

CHAPTER IV

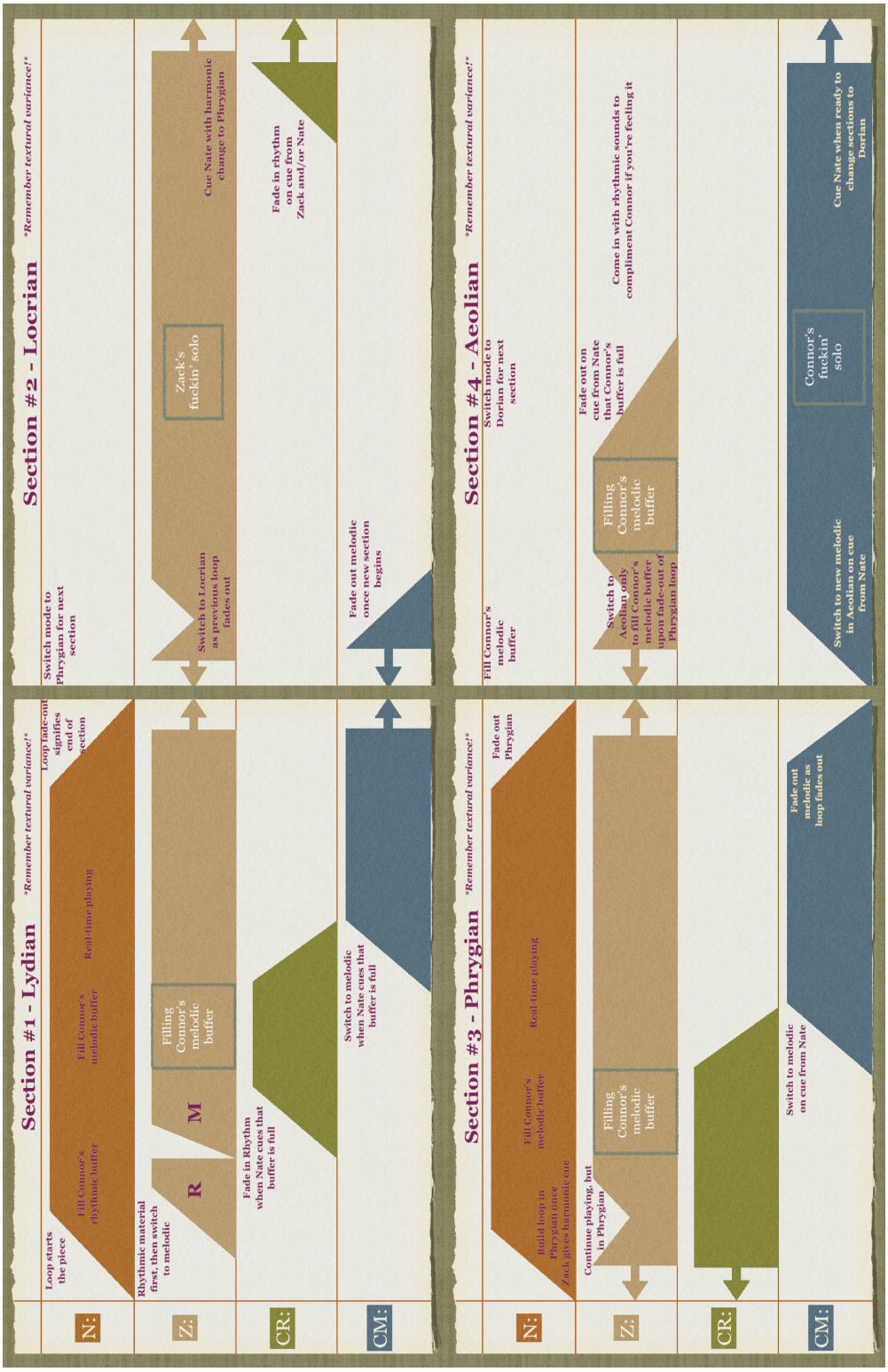
Summary

Even though this Digital Portfolio Dissertation marks the culmination of my graduate studies at the University of Oregon, I am by no means finished creating new instruments or composing new music. Everything I have learned, each interface I have built or designed, each piece of music I have composed, has become embedded and integrated into my artistic practice, building up my repertoire of creative tools from which to draw during my creative endeavors. Loops and improvisation will, in some way, shape, form, or play a role in the music that I compose for the foreseeable future. That being said, I have realized through my work creating the pieces discussed in this document that although I have my own distinct style and artistic fingerprint, my music and instrument design can, and should, always continue to grow and evolve.

At the time of this writing, I feel extremely fortunate to have accepted a tenure-track teaching position at the State University of New York in Oneonta, New York as an Assistant Professor of Audio Arts. Not only does this position personally validate the last ten years of my work as a graduate student, but it has reinvigorated and renewed my passion and desire to keep creating new instruments and music. Inventing and designing successful new data-driven instruments requires knowledge and mastery of not only electronics, music technology, and sound engineering, but also a comprehensive understanding of music composition, theory, and performance practices. Working as an instrument builder and electroacoustic composer for the last ten-plus years has enabled me to develop a cohesive, comprehensive artistic practice and teaching philosophy that I

can now utilize and leverage as an educator in the classroom for my future students. I hope to pass on what I have learned and created during my time as a student, and also to continue to create, explore, and advance the field of music and technology within the arts. The work I have discussed in this document not only represents what I have been working towards as a graduate student of music technology, but also illuminates the path I hope to continue upon as a musician, composer, creator, and educator of music, technology, and the arts.

APPENDIX: VISUAL SCORE FOR *TRIO 1-465*



(Front)

Section #5 - Dorian
Remember textural variance!
Section #6 - Mixolydian
Remember textural variance!
N:
Begin new loop in Dorian on cue from Connor
Fill Connor's melodic buffer
Real-time playing
Fade out Dorian
Start loop in Mixolydian
After loop is going play in real-time as well
Nate's fuckin' solo
Fade out loop if it feels right
Cue Connor BEFORE fading out loop
Z:
Begin section in Dorian as Nate builds new loop
Filling Connor's melodic buffer
Fade out with building of new loop, not before new loop has begun
CR:
Fade out melodic and switch to rhythmic as Nate builds new loop in Dorian
Fade in rhythmic on cue from Nate
CM:
Switch back to melodic on cue from Nate
Fade out with building of new loop, not before new loop has begun
Section #7 - Ionian
Remember textural variance!
N:
Start loop in Ionian as Connor brings in rhythmic
Fill Connor's melodic buffer
Build, in an epic manner!
Fade out loop first, to signify ending
Z:
Start Ionian as Nate builds new loop
Filling Connor's melodic buffer
Build, in an epic manner!
Fade out second, listen to loop fade out for cue
CR:
Switch to melodic on cue from Nate, then switch freely between rhythmic and melodic. "Freeescratch" section
Build, in an epic manner!
Fade out third, listen to Zack fade out for cue
CM:
Build, in an epic manner!
Melodic is last to fade out, ends piece

(Back)

REFERENCES CITED

Bert Bongers, "Electronic Musical Instruments: Experiences of a New Luthier," *Leonardo Music Journal* 17 (2007): 9–16.

Electrical4U. "Potentiometer: Definition, Types, And Working Principle." *Https://www.Electrical4u.Com/*. Accessed February 13, 2020. https://www.electrical4u.com/potentiometer/.

Hunt, Andy, and Marcelo Wanderley. "Mapping Performer Parameters to Synthesis Engines." *Organised Sound* 7, no. 2 (August 2002): 97–108.

Miranda, Eduardo, Marcelo Wanderley, and Ross Kirk. *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. Vol. 21. The Computer Music and Digital Audio Series. A-R Editions, Inc., 2006.

Nettl, Bruno, and Melinda Russell. *In the Course of Performance: Studies in the World of Musical Improvisation*. University of Chicago Press, 1998.

Reich, Steve, and Paul Hillier. *Writings on Music, 1965 - 2000*. Oxford University Press, 2002.

Roads, Curtis. "Introduction to Granular Synthesis." *Computer Music Journal* 12, no. 2 (Summer 1988): 11–13.

———. *The Computer Music Tutorial*. The MIT Press, 1996.

Udell, Chet. "SensorMusik Week 7 Lecture." Lecture, University of Oregon, November 2014.

Weisstein, Eric W. "Interpolation." Text. Accessed February 9, 2020. http://mathworld.wolfram.com/Interpolation.html.

Wolaver, Dan H. *Phase-Locked Loop Circuit Design*. Prentice Hall, 1991.
"Accelerometer." Accessed February 12, 2020. https://docs.idew.org/code-internet-of-things/references/physical-inputs/accelerometer.

"Accelerometer," accessed February 12, 2020, https://docs.idew.org/code-internet-of-things/references/physical-inputs/accelerometer.

"Ancient Kauri – World's Oldest Wood Now at Woodcraft." Accessed February 22, 2020. https://www.woodcraft.com/blog_entries/ancient-kauri-world-s-oldest-wood-now-at-woodcraft.

"Appendix F. Window Functions," accessed April 11, 2020, http://www.csounds.com/manual/html/MiscWindows.html.

"Arc." *Docs*. Accessed February 22, 2020. https://monome.org/docs/arc/.

"Arduino - Software." Accessed February 13, 2020. https://www.arduino.cc/en/main/software.

"Audiobus: Live, App-to-App Audio.," accessed April 4, 2020, https://audiob.us/.

"AUM - Audio Mixer." *Kymatica.Com*. Last modified February 4, 2020. Accessed February 10, 2020. http://kymatica.com/apps/aum.html.

"Bengal | Ableton." Accessed February 10, 2020. https://www.ableton.com/en/packs/bengal/.

"Bluetooth LE MIDI Specification," accessed April 11, 2020, https://www.midi.org/specifications/item/bluetooth-le-midi.

"Dear Earth, Love Moon, by Hamilton Beach," *Hamilton Beach*, accessed April 10, 2020, https://hamiltonbeach.bandcamp.com.

"Definition of INTERPOLATE." Accessed January 20, 2020. https://www.merriam-webster.com/dictionary/interpolate.

"Definition of MUTABLE." Accessed February 27, 2020. https://www.merriam-webster.com/dictionary/mutable.

"Definition of SENSOR." Accessed February 13, 2020. https://www.merriam-webster.com/dictionary/sensor.

"Endless Encoder." *Sweetwater*. Accessed August 20, 2019. https://www.sweetwater.com/insync/endless-encoder/.

"Grid." *Docs*. Accessed February 22, 2020. https://monome.org/docs/grid/.

"Hann (Hanning) Window - MATLAB Hann," accessed April 11, 2020, https://www.mathworks.com/help/signal/ref/hann.html.

"High Resolution MIDI or OSC Controllers? - MaxMSP Forum | Cycling '74." Accessed February 28, 2020. https://cycling74.com/forums/high-resolution-midi-or-osc-controllers.

"IDensity." *Electronic Music Software*. Accessed February 10, 2020. https://www.apesoft.it/idensity/.

"Improvisation | Music." *Encyclopedia Britannica*. Accessed January 24, 2020. https://www.britannica.com/art/improvisation-music.

"Introduction to OSC | Opensoundcontrol.Org." Accessed January 20, 2020. http://opensoundcontrol.org/introduction-osc.

"IRCAMAX 1 | Ableton." Accessed February 10, 2020. https://www.ableton.com/en/packs/ircamax-1/.

"Jitter." *Wikipedia*, November 12, 2019. Accessed January 20, 2020. https://en.wikipedia.org/w/index.php?title=Jitter&oldid=925856848.

"Loopy, the Live Looper App for IPhone and IPad," accessed April 4, 2020, https://loopyapp.com/.

"Max for Live | Ableton." Accessed January 21, 2020. https://www.ableton.com/en/live/max-for-live/.

"Monome." Accessed February 22, 2020. https://monome.org/.

"Morph - Sensel Morph Documentation." Accessed February 10, 2020. https://guide.sensel.com/morph/.

"Morpheus | Ableton." Accessed February 10, 2020. https://www.ableton.com/en/packs/morpheus/.

"Music Loop | Glossary of Music Production Terms | Media Music Now." Accessed February 13, 2020. https://www.mediamusicnow.co.uk/information/glossary-of-music-production-terms/what-is-a-loop.aspx.

"Music Thing: Monome Controller." *Engadget*. Accessed February 10, 2020. https://www.engadget.com/2006/04/14/music-thing-monome-controller/.

"Operator | Ableton." Accessed February 10, 2020. https://www.ableton.com/en/packs/operator/.

"RollerMouse Red Series," *Contour Design Inc*., n.d., accessed April 11, 2020, https://www.contourdesign.com/product/rollermouse-red/.

"Sample-Based Synthesis." *SoundBridge*, April 18, 2017. Accessed February 21, 2020. https://soundbridge.io/sample-based-synthesis/.

"Specs." *MIDI Association: Specifications*. Accessed January 20, 2020. https://www.midi.org/specifications.

"Switch Basics - Learn.Sparkfun.Com." Accessed February 12, 2020. https://learn.sparkfun.com/tutorials/switch-basics/all.

"TC-Data - Bit Shape." Accessed February 10, 2020. http://www.bitshapesoftware.com/instruments/tc-data/.

"The Band Cello." *Headway Music Audio*, n.d. Accessed February 10, 2020. https://www.headwaymusicaudio.com/product/the-band-cello/.

"The Complete MIDI 1.0 Detailed Specification." The MIDI Manufacturers Association, 1996.

"The Interdimensional Wrecked System (IWS) — Ms Pinky's Playhouse." Accessed February 10, 2020. https://mspinky.com/about/.

"The Sensel Morph." *Sensel*. Accessed February 22, 2020. https://sensel.com/pages/the-sensel-morph.

"The Sensel Morph: INTERACTION, EVOLVED." *Kickstarter*. Accessed February 22, 2020. https://www.kickstarter.com/projects/1152958674/the-sensel-morph-interaction-evolved.

"Vinyl — Ms Pinky's Playhouse." Accessed February 10, 2020. https://mspinky.com/vinyl/.

"What Is Ancient Kauri Wood? | Ancientwood, Ltd," n.d. Accessed February 22, 2020. https://www.ancientwood.com/what-is-ancient-kauri-wood/.

"What Is Jitter?" Accessed February 28, 2020. https://www.speedcheck.org/wiki/jitter/.

"What Is Max? | Cycling '74." Accessed February 13, 2020. https://cycling74.com/products/max.