Ripple Synth

- An Investigation into Novel Interaction Techniques for Scanned Synthesis -

Thesis Report

Michael Anthony Castanieto

Aalborg University
School of Information and Communication Technology
Niels Jernes Vej 12
DK-9220 Aalborg

School of Information and Communication Technology
Niels Jernes Vej 12
DK-9220 Aalborg Ø
http://sict.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Ripple Synth: An Investigation into Novel Interaction Techniques for Scanned Synthesis

Theme:
Scientific Theme

Project Period:
Spring Semester 2018

Project Group:

Participant(s):
Michael Castanieto

Supervisor(s):
Stefania Serafin

Copies: 1

Page Numbers: 37

Date of Completion:
September 12, 2018

Abstract:

Ripple Synth is an application developed for the purpose of evaluating whether a novel gestural mapping technique, based on scanned synthesis, will add expressive value to the experience of playing a new digital musical instrument.

Contents

Preface

Aalborg University, September 12, 2018

Michael Castanieto
<mcasta16@student.aau.dk>

Chapter 1

Introduction

Humans will always strive to be explorers and will seek to find their own truth and meaning in this world, and in this process will uncover unique forms of expression that will serve to enrich their own experience and, in turn, that of others. The everlasting pursuit for new forms of expression has led musicians and artists to seek new mechanisms that will allow them to unlock their creative potential. One such pursuit has led some down the path of sonic exploration. The way in which people have expressed themselves with music has always evolved with new advances in technology. From innovations in instrument design to inventions of new ones, these advances have opened up new possibilities for the artist. Recent advances in digital technology have made it possible to develop new methods of digital synthesis that have allowed people to explore more uncharted territory in the sonic landscape. One of these more recent developments is in a technique called scanned synthesis.

1.1 Motivations for Scanned Synthesis Research

Scanned synthesis is a synthesis technique that is still quite in its infancy. With only two decades of development, there is much left to be explored, in terms of new possibilities for audio synthesis as well as novel ways of interaction. Much of this development focuses on simulating the physical behavior of and interaction between a configuration of masses and springs, known as a string in one dimension or a mesh when configured in more than one dimension. Different audio extraction methods can then be employed to scan the behavior of the mesh at several different points, known as the orbital. Even though the foundation of scanned synthesis is quite straightforward, it leaves room for much exploration. New modes of interaction, the seemingly endless audio extraction techniques, and the exploration of new mapping strategies between the mesh behavior and audiovisual feedback leaves the door open for a myriad of research opportunities.

1.2 Research Goal

The aim of this research is to explore the relatively uncharted territory of scanned synthesis and investigate whether a new gestural mapping strategy will add value to the musical performance experience in the context of a new digital musical instrument called Ripple Synth. The new gestural technique utilizes the gestural movements associated with generating ripples on the surface of water, which is a gesture that one should naturally associate with when exciting the behavior of a haptically moving mesh membrane [Erlach et al., 2011].

Chapter 2

Background

This section will discuss much of the history and theoretical development of scanned synthesis, including most of the prominent methods used for this technique. The expressive gestures used in the Ripple Synth application focuses on a new mode of interaction for exciting the two dimensional mesh structure. Important considerations must be given in order to evaluate the use of this interaction in the context of *digital instrument design*. This will also be discussed in detail.

Much of the groundwork and fundamental development of scanned synthesis was accomplished with two seminal papers, Verplank et al. [2000] and [Boulanger et al., 2000]. Further development typically relies on novel interaction methods, new methods of audio extraction, and new mapping procedures from the interface to the mesh and to audiovisual parameters, but they are all rooted in the same techniques developed in these two papers.

2.1 The Early History and Development of Scanned Synthesis

Scanned synthesis is a technique that was first introduced by Bill Verplank et al., in their paper "Scanned Synthesis" [Verplank et al., 2000]. This paper describes a new method of synthesizing audio by controlling a dynamic and continuously evolving system that vibrates at haptic frequencies, which the authors describe as frequencies that are at or below 15 hz. This system can be viewed as a dynamic wavetable that is under the control of the user. By definition, the dynamic wavetable is constantly being updated, whose values are determined by the method in which this vibrating system is being scanned. These values can then be used to synthesize audio, either by using the extracted values themselves as the raw audio data, or by mapping these values to other parameters that manipulate or synthesize audio. The reason for developing such a method was to give the performer a new way to explore the timbre of a sound. The haptic vibrations with which this timbre evolves is meant to give an impression of a sound that changes naturally over time [Verplank et al., 2000].

Early developments of scanned synthesis use a one dimensional closed string model as the vibrating system [Verplank et al., 2000] [Boulanger et al., 2000]. This model is comprised of masses and springs that are connected together in series, forming a string that is connected together at both ends in a closed loop. The vibrating behavior of this string is determined by using the Newtonian laws of motion to derive equations that use a finite element approximation of the mass spring system [Verplank et al., 2000]. Certain properties of the string can be modified, which in turn effect the vibrating behavior of the string. These properties include the weight of the masses, the stiffness of the springs connected to the masses, the stiffness of the springs connected to the ground, the damping behavior of the springs connected to the ground, the horizontal length between masses, and the position at which force is applied to a mass by the user [Verplank et al., 2000]. The relation between these properties can be seen in figure 2.1. Early developments in extracting audio

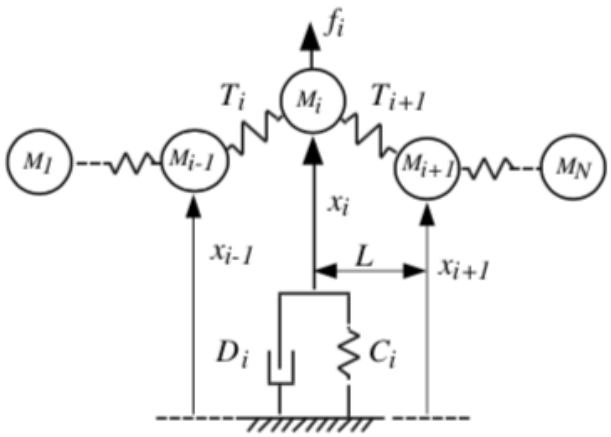

Figure 2.1: Relation between properties of a one dimensional mass spring model. M, mass; T, spring stiffness between masses; C, spring to earth; D, damping to earth; L, length between masses; x, position; f, force Verplank et al. [2000]

from this string behavior were rather straightforward. Verplank et al. updated a wavetable using the vertical displacement values of the masses, and used interpolation to generate enough values in the wavetable to scan them at audio rate [Verplank et al., 2000]. The rate at which these values are being scanned determines the pitch of the sound, much like in wavetable synthesis.

2.2 Scanned Synthesis Methods

Further development yields a two dimensional mass spring configuration, known as a mesh. Expanding the vibrating system to this two dimensional system opens up new modes of interaction as well as new types of behavior to the system. It also introduces new methods of audio extraction. These methods will be described in the sections that follow.

2.2.1 Mass Spring Connection Schemes

Richard Boulanger et al. were the next to explore scanned synthesis using this two dimensional mesh configuration [Boulanger et al., 2000]. They explored novel ways of constructing the mesh, exciting the mesh, and generating audio from its behavior. Opening up the string model to two dimensions lends itself to new mass spring connection schemes. Since masses and springs are no longer confined to a sequential order in the two dimensional model, this allows for arbitrary connections to be made. Two extreme cases of this connection scheme can be seen in figure 2.2, which

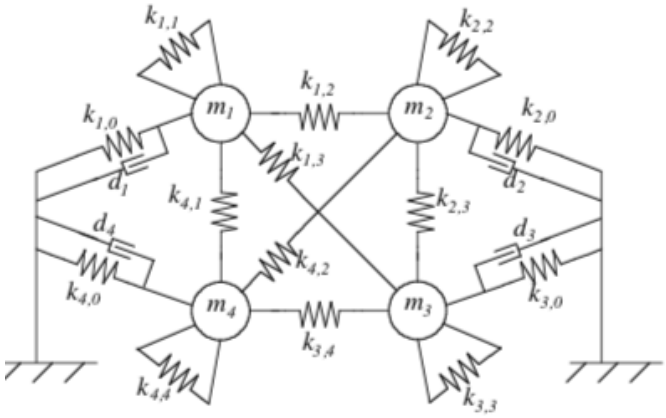

Figure 2.2: Connection scheme where every mass is connected to every other mass. [Boulanger et al., 2000]

shows a system where each mass is connected to every other mass with a spring, and figure 2.3, which shows a connection scheme that reverts back to the one dimensional

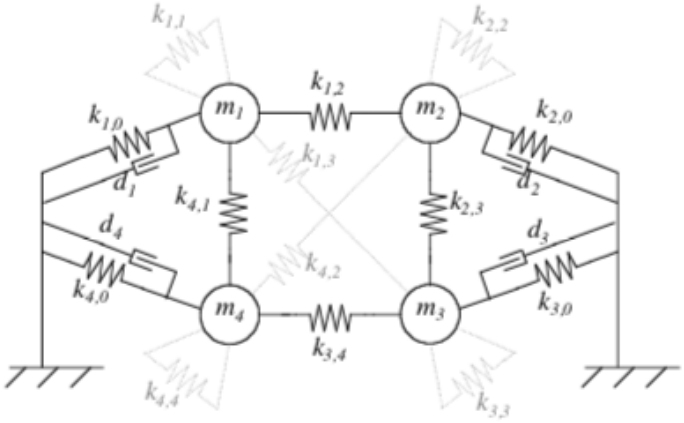

Figure 2.3: Connection scheme that reverts back to the one dimensional string model. [Boulanger et al., 2000]

closed string model. Richard Boulanger et al.'s findings showed that the most inter-

esting and dynamic timbres resulted from non-uniform stiffness between mass spring connections. In general, having non-uniform parameter distributions throughout the mesh has proven to produce more interesting results sonically [Boulanger et al., 2000].

2.2.2 Mesh Excitation

One of the ways Richard Boulanger et al. experimented with exciting the mesh was by driving it with an audio signal, where the value of the next audio sample in the sequence will be added to the displacement of the next mass that will be scanned in the sequence. This results in the audio input affecting resonant frequencies within the mesh structure [Boulanger et al., 2000]. The authors suggest, but did not experiment in other novel methods of exciting the mesh model, such as the use of stochastic mathematical models for influencing mesh behavior, including a chaotic system, a genetic algorithm, or cellular automata [Boulanger et al., 2000]. Verplank et al. claims to have used chaotic equations, namely heat equations such as the Kuramoto-Shivashinski equation for exciting their own mesh model, but does not elaborate on the results of their findings [Verplank et al., 2000].

2.2.3 Orbital Scanning

The two dimensional mesh configuration also lends itself to novel audio scanning methods. Not only is it possible to set up arbitrary connection schemes, it also opens up the possibility to traverse through the mass spring connections using arbitrary pathways to extract values for audio synthesis. Research into these audio scanning trajectories was first done by [Boulanger et al., 2000] and was later given the name "orbital" by [Tubb, 2011], which is a term that was originally used to described the audio scanning behavior of wave terrain synthesis [Mitsuhashi, 1982].

As was previously mentioned, it was the work of Boulanger et al. which led to the first experimentation of different scanning trajectories [Boulanger et al., 2000]. Some examples of these orbital scanning configurations can be seen in figure 2.4. Boulanger et al.'s conclusion was that there was no correct way to which an orbital should be scanned. The only exception to this was that the scanning pathway must be between connected masses in order to insure a smooth and continuous sound, unless the user wishes to introduce additional periodicities within the sound. The authors have also concluded that the creation of audio scanning trajectories is a non-intuitive procedure where the desired result depends on what the user wishes to achieve.

Wave Terrain Synthesis also explores using many different audio scanning trajectories. It is a synthesis technique which extends the wavetable lookup principle by scanning various orbital trajectories on three-dimensional surfaces [Roads, 1996]. The main difference between Wave Terrain Synthesis and Scanned Synthesis is that the three dimensional terrain in Wave Terrain Synthesis is static while the orbital position that is projected on the terrain typically is not. Stuart G. James has researched the sonic effects of different orbital trajectories projected on various three-

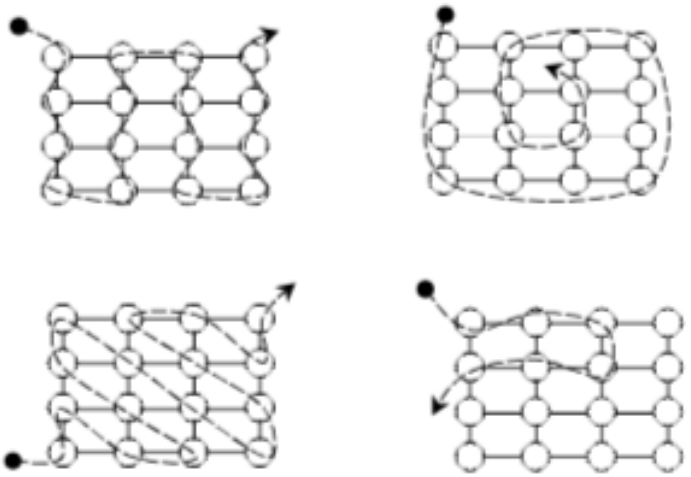

Figure 2.4: Different orbital pathway configurations. [Boulanger et al., 2000]

dimensional surfaces. His conclusions reveal that more complex trajectory signals, e.g. ones that reflect a high level of spectral complexity, tend to generate noise results. Moreover, the aspects of the original signal tend to be lost when reshaped by the terrain [James, 2005]. James also suggests that certain trajectories lend themselves to generating more musical results than other trajectories, such as Rhodea curves, spirographs, and spiral curves. Mills and De Souza, who have also investigated the effects of different trajectories on various terrains, have come to the conclusion that rectangular orbits allow for the best control of harmonic content over time [Mills and de Souza, 1999].

Tubb experiments with different orbital scanning trajectories in scanned synthesis by allowing the user to select different scanning patterns in his Wablet application. His findings showed that the most interesting harmonic effects were produced when the orbital shape was symmetric, overlapped on itself (meaning that at some paths of the orbital were scanned more than once), and the start and end points of the orbital were joined together smoothly [Tubb, 2011]. Tubb experimented with Rhodonea curves to generate complex orbital patterns that adhered to these three criteria. The formula for generating the Rhodonea curves is:

$$r = a \sin(\frac{n}{d}\theta)$$

What he discovered was that although large values for n and d produced more complex patterns which led to stronger harmonics, values higher than 16 led to sounds that were too harsh and atonal. The more complex and abstract the patterns became, the less it became associated with the vibrating mesh model. Tubb also opened up the possibility for the user to create their own scan path trajectories by allowing them to trace their own orbital patterns graphically onto the mesh. He concluded,

however, that the process ended up being too awkward and time consuming for the user [Tubb, 2011].

2.2.4 Dynamic Wavetable

The values that are scanned and updated from the orbital scanning trajectories are used to generate a wavetable. This wavetable functions very much in the same way as in wavetable synthesis, where the stored values represent audio data that is periodically scanned to produce the samples of an audio sound wave [Verplank et al., 2000]. The main difference between wavetable synthesis and scanned synthesis is that the wavetable is constantly being updated in order to reflect the audio data that is extracted from the dynamically changing orbital shape. This dynamic wavetable evolves over time, reflecting the haptic vibrations of the changing mesh structure. The rate at which the wavetable is being updated is at haptic rates [Verplank et al., 2000], meaning that the table is updated as soon as new values are changed and updated within the vibrating mesh structure.

The data stored in this dynamic wavetable does not have to be limited to raw audio data. Much like in wavetable synthesis, this data can be abstracted to allow for the control many various algorithms or parameters. Rhoads suggests that frequency modulation synthesis and granular synthesis are prime candidates for using dynamic wavetables for dynamic audio control [Roads, 1996]. Boulanger et al. suggests using such control for spectral shaping. This will allow, for example, the control of a bank of oscillators that will act as multiple sweeping filters on the sound [Boulanger et al., 2000].

2.2.5 Finite Element Approximations

Verplank et al. derived a finite element approximation to a one-dimensional closed string of springs and masses by applying Newton's equations on a generalized string and discretizing the values. They argue that such an approximation need not be accurate for scanned synthesis, only that it should be stable and musically interesting. Their derivation involves a series of difference equations that describe the motion of each mass element of the string as well as the spring forces interacting on them [Verplank et al., 2000]:

$$a_i = \frac{1}{M_i}\left[K_i(x_{i-1} - 2x_i + x_{i+1}) - C_i x_i - D_i v_i + f_i\right] \tag{2.1}$$

$$v_i = \int a_i dt \tag{2.2}$$

$$x_i = \int v_i dt \tag{2.3}$$

Where (2.1), (2.2), and (2.3) represent the acceleration, velocity, and position of the i th element, respectively. Other parameters include $f_i(t)$, the haptic force on the i

th element; M_i, the mass of the i th element, $K_i = \frac{T_i}{L_i}$, the effective spring constant between i and *i-1*; C_i, the spring constant to the earth for the i th element; and D_i, the damping of the i th element. Since this is a closed string, the boundary conditions for the two ends are $x_0 = v_0 = 0$ and $x_N = v_N = 0$, where N represents the last element of the string. Discretizing the equations yields numerical solutions which can be used in a computer, the details of which are documented in Appendix A of [Verplank et al., 2000].

2.2.6 Physical Modelling

Scanned synthesis may utilize techniques found in physical modelling to develop the haptically vibrating system. Physical modelling is a method of synthesis which focuses on modelling the physical behavior of both electronic and acoustic musical instruments. The methods used for modelling this behavior vary according to the goal of what must be achieved, but generally these models fall under one of two types: *lumped* and *distributed*. Lumped models approximate physical systems with masses, springs, dampers, and nonlinear elements, while distributed models use digital waveguides to model wave propagation in distributed media [Smith, 1996]. The model which is more suitable for membranes and plates is the distributed model. Wave propogation of a vibrating string and vibrating membrane can be modelled by the one-dimensional and two-dimensional wave equation, respectively.

The One-Dimensional Wave Equation

Digital waveguides can be described as a network of bidirectional delay lines which move energy from one part of the network to another. These delay lines are connected at scattering junctions which simulate the behavior of the distributed system [Bilbao, 2001]. The same model can be used to describe the motions of a vibrating string. The displacement of an ideal vibrating string can be described by the following second-order partial differential equation [Duyne and III, 1993]:

$$u_{tt}(t,x) = c^2 u_{xx}(t,x) \tag{2.4}$$

which says that the transverse acceleration u_{tt} of a point on the string is directly proportional to the string's curvature u_{xx} at that point. Solving for the transverse displacement $u(t,x)$ of the string, where t is time and x is the displacement of the string at a given point, will yield a solution that shows the summation of two left and right traveling waves along the string [Bilbao, 2001][Duyne and III, 1993]:

$$u(t,x) = g^+(x - ct) + g^-(x + ct) \tag{2.5}$$

where g^+ and g^- denote waves travelling to the right and left, respectively. The waves travel at a speed of $c = \left(\frac{T}{\epsilon}\right)^{0.5}$, where T is the constant tension of the string and ϵ is the mass per unit length [Duyne and III, 1993]. From this, it is possible to obtain the velocity, $v = v^+ + v^-$, and force, $f = f^+ + f^-$, components of the traveling waves, which Julius O. Smith III fully derives in [Smith, 1992].

The Two-Dimensional Wave Equation

The second-order partial differential equation for describing the wave propagation of a string can be extended to describe the wave propagation of a membrane Bilbao [2001]Duyne and III [1993]:

$$u_{tt}(t,x,y) = c^2[u_{xx}(t,x,y) + u_{yy}(t,x,y)] \tag{2.6}$$

where x and y denote the horizontal and vertical spacial coordinates along the membrane.

Solving the equation for the wave displacement $u(t,x,y)$ of the two-dimensional membrane yields [Duyne and III, 1993]:

$$u(t,x,y) = \int g_\alpha(x\cos\alpha + y\sin\alpha - ct)d\alpha \tag{2.7}$$

where g_α is a function which represents the velocity potential of the propagating waves, in which the direction of propagation is inclined at an angle α and is traveling at speed c [Morse and Ingard, 1968]. This solution, however, requires an infinite number of arbitrary plane waves to be traveling in all directions, which in turn requires an infinite number of waveguides, one assigned to each plane wave [Duyne and III, 1993]. One solution to this problem is by formulating a two-dimensional waveguide mesh, which describes the behavior of the vibrating membrane in terms of a 2-D network of biderectional delay lines and 4-port scattering junctions. The details of formulating this mesh network will not be discussed as they are beyond the scope of this paper, but solutions to the digital waveguide mesh can be found in [Bilbao, 2001] and [Duyne and III, 1993].

2.3 Human Computer Interaction Applied to Digital Instrument Design

The progression of instrument design has reached a new paradigm with the invention of computer music. This new shift in digital technology has introduced new challenges as to how and what to design to conjure new modes of musical expression. One of the main challenges of digital instrument design stems from the fact that the physical interface for generating sounds and the device from which sound is generated are completely separate. This separation of controller and sound generator is what lies at the core of a *digital musical instrument*, or DMI [McGlynn, 2014] [Wanderley and Depalle, 2004]. The reason this poses a challenge is because the relationship between the performer's gestures onto the controller and the resulting musical output are arbitrary and designed [McGlynn, 2014]. This relationship is defined by the way the designer chooses to map parameters from the gestural controller to audio parameters. Moreover, there is the challenge of evaluating such a design. Since mapping schemes can be arbitrarily defined with limitless possibilities, how are we to know which schemes are appropriate for musical expression? This section will discuss some of

the primary aims that designers wish to achieve with the creation of a new digital musical instrument as well as the challenges that go with evaluating such novel designs.

2.3.1 Aims of Digital Instrument Design

Traditional acoustic instruments can offer such a wide variety of complex, nuanced, and creative potential for a performer that the possibilities are seemingly endless. The guitar, for example, is the perfect example of an instrument that has lead to the exploration of new genres, styles, and modes of expression, which continues to be explored and performed in new and inventive ways. With such creative potential at hand, why bother with this new paradigm of digital instruments? What can digital musical instruments possibly offer that acoustic instruments can't?

According to Sergi Jordà Puig, the creation of a new digital musical instrument can lead to new ways of thinking and approaching an instrument, new ways of interaction, and new ways of organizing textures, which in turn will ultimately lead to new forms of music [Puig, 2005]. In other words, DMIs can offer new forms of creative expression, which in turn will lead to a level of musical creativity that cannot possibly be reached with the aid of acoustic instruments alone. Another goal, according to Marcelo Wanderley, is that DMIs should offer similar levels of subtle control, but also extend the capabilities of the instrument [Wanderley and Depalle, 2004]. Reaching these goals are no trivial task. Since all DMIs start with the problem of how to map the control interface to the sonic output, there remains the challenge of how to develop a successful mapping scheme that will satisfy these goals.

2.3.2 Challenges in Evaluating DMIs

The physical interface of an acoustic instrument is inextricably linked to its sound source in such a way that it is virtually impossible to separate one from the other. With DMIs, however, this link is initially severed and as a consequence connections between the them must be carefully considered before bringing them together. Many considerations need to be made before attempting a design. There are questions of context, learning curve, reproducibility, and efficiency, to name a few [McGlynn, 2014] [Puig, 2005]. Puig arguies that the crafting of new digital musical instruments cannot be considered a science. However, the closest we can come to a clear evaluation of a digital instrument is by assessing its mapping strategy, even though there is no clear guideline as to how something should be mapped [Puig, 2005]. Tubb makes the argument that having unbounded freedom has the consequence of having an excess of mapping options, and that scanned synthesis is meant for a more inflexible design: "the more intrinsic constraints apply to the control mapping, the more specific playing techniques will be developed, and the more longevity the instrument will possess."

Chapter 3

Related Works

This chapter will discuss the projects that are relevant to the design and development of the Ripple Synth application. The projects related to scanned synthesis are, unfortunately, few and far between. Aside from the fact that scanned synthesis is a relatively new technique, the research and development that has been done has been very sparse throughout the two decades of its existence. The lack of published works and projects has also been noted by Tubb during his research and development of the Wablet scanned synthesis application [Tubb, 2011]. His spectulation is that this is due to several factors including patent issues with the scanned synthesis algorithm itself, the lack of general favorability with utilizing such techniques, and that, until recently, researchers and developers have not had access to the advanced computational and interactive hardware that can allow them to fully exploit the creative power of this technique. Despite this, there have been several notable developments which have had a contributing influence on the Ripple Synth design, both former and recent.

3.1 Max Mathews' Baton Scanned Synthesis String

Max Mathews has demonstrated how to perform with the one-dimensional finite element string model he developed in [Verplank et al., 2000] by using his radio baton invention to excite the string in various locations with different types of force interactions [Mathews, 1998]. Mathews' radio baton was developed as a MIDI controller for use in live musical performances. It consists of a flat, rectangular-shaped box which contains a processor that uses radio communication to track the x, y, and z movements of two batons and are also used as triggers when striking the box [Mathews, 2000]. Figure 3.1 shows Mathews demonstrating his radio baton instrument. Before each demonstration, Mathews sets up various initial conditions to control the type of string displacement that occurs when one of the batons strikes the box and to control certain characteristics of the string vibrations. There are three ways in which string displacements can be initiated: sine wave displacements, pulse wave displacements, and haptic displacements. Sine wave displacements generate more

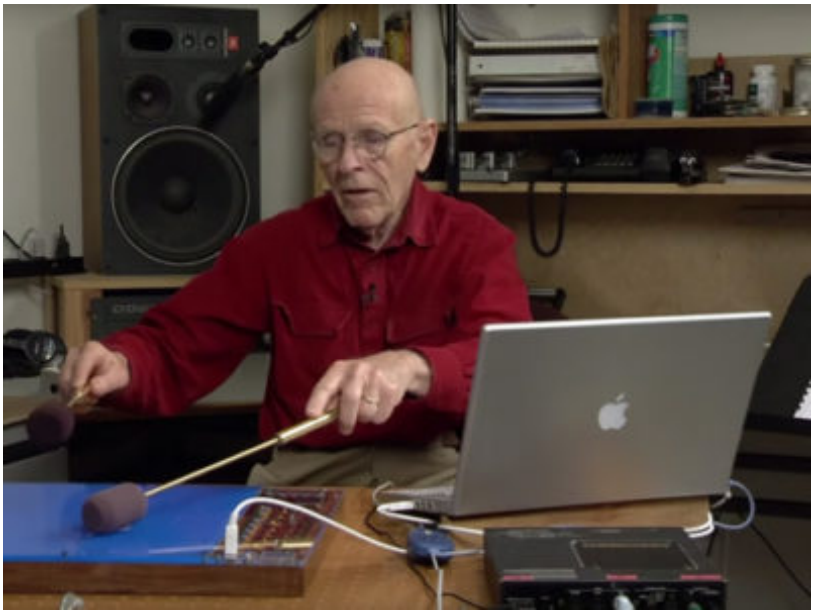

Figure 3.1: Max Mathews demonstrating his radio baton instrument Boulanger Labs [2018]

percussive sounds. Mathews controls the percussion hit and pitch with one baton, by striking the box at various points along the x position, and controls the centering and damping forces of the string by sliding the other baton along the box's x and y position. Plucked sounds are controlled in the same manner, but with initial displacements being instigated with a pulse. Haptic movements are controlled by sliding the two batons along the surface of the box across both the x and y plane simultaneously to instigate various haptic displacements on two points of the string. Here the two batons represent forces which displace the string at various locations by pushing the string up or pushing it down.

3.2 Paris Smaragdis' Csound Opcodes

The algorithm Mathews used for controlling the finite element string model in [Verplank et al., 2000] with his radio baton has also been coded in Csound by Paris Smaragdis and has been made widely available for public use [Smaragdis, 1999b]. This software allows for the user to experiment with different characteristics of the string model and to interact with this model in real time. The parameters that can be modified include the dampness, centering force, and stiffness of the springs, as well as the weight of the masses. The user also has control over the shape, position, and intensity of the striking force. The author warns that experimenting with certain characteristics of the string, namely the mass values, centering force, and damping can lead to a highly unstable string which can potentially "blow up" the simulation [Smaragdis, 1999a].

3.3 Scanned Synth Pro

Humanoid Sound Systems' Scanned Synth Pro was the first commercially available VST which uses scanned synthesis [Humanoid Sound Systems, 2018]. This plugin allows the user access to different scanned synthesis parameters to effect the weight of the masses, centering force of the springs, and the quality of the striking force, as well as the type of mass spring configuration topology. Unfortunately, the proprietary nature of this software disallows any further investigation into the details of this program. In addition, the software is featured along with a myriad of different filters and effects that obscure the actual scanned synthesis sound. Since the plugin's graphical layout only consists of the typical knobs and sliders, the lack of visual feedback makes it difficult to understand how the waveform is being effected and how it's evolving. Similar criticism has been noted by Tubb in [Tubb, 2011]. Despite these criticisms, Scanned Synth Pro remains one of the few successful commercial products that features scanned synthesis.

3.4 Scanned

The latest in the development of a commercial product to feature scanned synthesis is Scanned by Qu-Bit Electronix. This product is implemented as a modular synthesizer component and claims to be the first product to fully implement scanned synthesis in hardware. They describe their product as an 'organic wavetable VCO' that uses the 'unexplored synthesis technique known as Scanned Synthesis' to excite a 'vibrating string' which 'generates wavetables on the fly' [Qu-Bit Electronix, 2018]. Although such marketing jargon may be slightly deceptive, e.g. the 'organic' nature of the wavetable and the fact that this technique has been explored for two decades (perhaps 'scarcely explored' is the more apt terminology), it is descriptive enough to give an idea of how this product functions. The knobs on this modular synthesizer change the characteristics and force interactions of a one-dimensional vibrating string. These parameters are the type that one expects to have control over, such as the centering force, dampening, and stiffness of the springs, the weight of the masses, and the scanning frequency. Other parameters make it possible to affect the shape, position, and strength of the hammering force on the string. As is often found in modular synthesizer hardware, VCO inputs and outputs make it possible to patch any of these parameters to other modular synthesizer components. As to date, this product has not been released to the general public and is expected to be available on the market sometime in September 2018.

3.5 Multidimensional Physical Modelling with the Sensel Morph

Tissieres et al. developed a digital percussion instrument that uses physical modelling techniques to produce various percussion sounds controlled by a Sensel Morph, a flat rectangular touch controller that uses pressure sensitivity to capture various user touch interactions [Tissieres et al., 2018]. The relevance of this project to the Ripple Synth project isn't so much the physical modelling techniques involved, albeit they do use a two-dimensional waveguide mesh as one of the resonator systems to their model, but in their evaluation of user interaction to their system with the Sensel Morph. Their project is the first to publish usability testing results involving this particular controller. According to the authors, physical interactions and haptic feedback response were important for the percussive nature of their physical modelling system, which needed to capture and respond to various nuanced gestural movements. The Sensel Morph allowed them to develop interactions that utilized such nuanced activity. One thing they conclude from their evaluation is that by abstracting the input sensor data in more complex ways and by using them for a variety of different gestural mapping schemes, their instrument could be extended to allow the performer even more expressive control over the system's parameters.

3.6 Body Controlled Scanned Synthesis

Yoichi Nagashima developed an application which focuses on the expressive real time control of scanned synthesis parameters using movements of the arms [Nagashima, 2004]. This novel interaction technique uses a MiniBioMuse-III, a sensor which tracks eight independent channels of EMG information on the arm. One MiniBioMuse-III is strapped to each arm, mapping 16 channels of EMG data to 16 values that affect scanned synthesis parameters. The simplest mapping technique implemented was using the 16 channels of EMG data to effect the displacement values of the scanned waveform. However, such mapping led to results that did were not exactly deemed meaningful and were hard to control. The author proposed to expermiment further with other mapping techniques for future compositions, but has yet to verify his results. Despite the lack of further research in this area, it does represent an attempt to explore the mapping of meaningful gestural movements to scanned synthesis parameters.

3.7 Tüb

Tüb is an installation that uses methods to sonify the haptic vibrating forces of a real object, namely that of water waves flowing in a tub [Erlach et al., 2011]. These methods involve positioning a light and a webcam from above the tub and capturing the shadow ripple patterns that are produced at the bottom of the tub. The ripple

pattern images are scanned using GEM software and audio is processed using Pure Data. Elliptical paths were used to scan the image frames and update the wavetable. Tüb is the first scanned synthesis project that uses the behavioral properties of water as the haptic vibrational medium for scanned synthesis. This opens up interesting possibilities in two ways. Firstly, it gives an idea of the different expressive ways in which the performer can interact with such a medium, and how the these properties should respond to such an interaction. For instance, experimenting with different fluid viscosities, volumes of water, and ways of exciting this system could lead to many interesting artistic possibilities. The authors leave much of this interaction to future work. Secondly, it shows how the behavior of a fluid like mesh will affect its sonic properties, in terms of scanned synthesis. The authors observed that the richness and complexity of the wave patterns produced by the myriad of ways in which one can interact with the water was reflected in the sonic output. This in turn lead to a more engaging and rich experience for the performer [Erlach et al., 2011].

3.8 Wablet

Wablet is an application that use's an iPad's multi-touch interface to allow touch interaction with a graphically displayed two-dimensional mesh structure, which generates sounds according to scanned synthesis methods [Tubb, 2011]. It utilizes certain touch interaction gestures for exiting the mesh membrane, such as grabbing and releasing point masses, creating mass-repelling force fields around the area of touch contact, constraining masses into a locked position, freeing masses from a locked position, inscribing orbital paths, creating sinusoidal or radial spacial harmonics in the structure, and shaking the entire mesh using the iPad itself. The audio extraction techniques mainly relied on the shape of the orbital pathway (discussed in 2.2.3) and the different types of motion one can give to the mesh structure. For instance, spinning a free floating mesh will introduce centrifugal force. This force interaction will be scanned along the orbital pathway and used for sound generation. In general, oscillations in various force and speed interactions reflected the final output of the sound.

Wablet's design, particularly it's code structure, was an influence on the design of the Mesh, Mass, and Spring classes of the Ripple Synth application. Particularly in the way force interactions behave, which applies forces in an object-oriented fashion in order to update the mesh structure, which is closer to the *lumped* model. Although this is a very elegant solution in terms of code design, it is not a very efficient solution. Tubb also implements another design according to the *distributed* model. Although the Wablet code itself is proprietary and therefore its details not disclosed, it gives a good idea of how to simulate mesh behavior in the object-oriented *lumped* model.

3.9 Mass-Spring System Simulation

This project gives an example of how to run a mass-spring simulation in Qt [Guy, 2018]. What's more, it show's how to render and display objects using the `QtOpenGL` module in Qt.

Chapter 4

Design and Implementation

The development of the application known as Ripple Synth will be described in this chapter in detail. This will include a description of the technologies used, a detailed discussion of the design of the application and its code structure, and an overview of the mapping strategies used.

4.1 Technologies Used

Ripple Synth utilizes several technologies, both in hardware and software, to form one complete application. The following sections will give a general description of each of these technologies as well as the purpose of its use in the application.

4.1.1 Sensel Morph

The Sensel Morph is a pressure sensitive multi-touch control interface. It has along its surface a grid of approximately 20,000 pressure sensors with a resolution of 32,000 levels of pressure sensitivity ranging from 5 grams to 5 kilograms [Sensel, 2018b]. Sensel provides its own API, which is freely available on its GitHub repository [Sensel, 2018a], so that developers have access to the real time input pressure data. This API does the job of parsing the raw input data and storing it as touch gesture information, giving the developer access to various gesture parameters such as the number of contact points, the total force applied at each point, and the location of contact. Touch data is delivered in frames, which can deliver 125 to 1000 frames per second, and has a latency of 2 to 8 milliseconds [Sensel, 2018b].

The Sensel API is available in three different programming languages: C, C#, and Python. Ripple Synth is programmed in C++ and therefore integrates the Sensel API in C by installing the appropriate runtime libraries and including the necessary C header files, namely `<sensel.h>`, which provides access to the different gestural control parameters, and `<sensel_device.h>`, which provides a handler for accessing the Sensel Morph device. Once the API is installed, the input data can then be extracted for further processing in Qt.

4.1.2 Qt

Qt is a C++ Software Development Kit for developing cross-platform GUI environments and embedded UI software. It provides a framework for developing applications that need to process 3D graphics and mathematical simulations in realtime [Qt, 2018a]. Software applications can be ported to a multitude of environments and embedded devices with relative ease. It has been accepted over the years by many notable software companies as a useful tool for developing UI software and graphical animations, including audio companies such as Native Instruments[1] and Ableton[2].

The input data from the Sensel Morph is processed in Qt by mapping the touch gestures to force interaction parameters which causes wave dispersion in a two-dimensional mass spring mesh network. The physical interaction forces instigate wave propagation throughout the membrane, which vibrate according to the properties of the mesh. The mesh behavior is defined and simulated within the Qt portion of the Ripple Synth application and details of its implementation will be further discussed in 4.3. Once the forces for all the masses and springs in the mesh network are calculated, they are updated to reflect the current state of the mesh and then displayed graphically in OpenGL.

4.1.3 OpenGL

OpenGL is a cross-platform open-source API that is used for rendering 2D and 3D graphics. Originally written in C, it contains a wide variety of other language bindings, including C++, Java, and Fortran, making it extensible to for use in many other environments and contexts [OpenGL, 2018]. Qt provides wrappers in their API allowing programmers to incorporate OpenGL functionality with relative ease, the most important being the `QtOpenGl` module [Qt, 2018b]. OpenGL's purpose is to only render graphics and does not contain any physics engine of its own, but it can work alongside Qt so that calculations for physical simulations can be done within Qt and rendered and displayed in OpenGL.

4.1.4 OSC

Once the current state of the mesh is updated in Qt and is rendered and displayed in OpenGL, the displacement, velocity, and acceleration data of each mass that lies along an orbital pathway are then transmitted using the OSC protocol.

Open Sound Control is a networking protocol that allows for realtime communication between various multimedia devices [Open Sound Control, 2018]. Even though this functionality is meant to communicate streams of data to other devices over a network, Ripple Synth uses it to only communicate updated mesh parameter data in realtime to the JUCE audio application. It does this by sending the data streams

[1] https://www.native-instruments.com/en/career-center/berlin/product-creation-development/software-engineer-c/

[2] http://blog.qt.io/blog/2015/12/15/ableton-push-qt-in-music-making/

over the computer's home IP address (127.0.0.1) at a specified port, sending data from one application to another over the local network. Ripple Synth makes use of Oscpack for sending OSC packets. Oscpack is a simple and lightweight API written in C++ that contains just enough functionality to allow for basic OSC communication [Bencina, 2018]. JUCE contains its own built in OSC functionality for receiving OSC packets.

4.1.5 JUCE

JUCE is a cross-platform open-source API for developing audio plugins and other various audio applications, which can be ported for use in different software and hardware environments [JUCE, 2018]. It's robust functionality makes it flexible for programming audio and other media in a variety of contexts and its use of the C++ programming paradigm makes it ideal for developing algorithms that need efficient audio processing.

The orbital data, having been transmitted via the OSC protocol, is sent to the JUCE application where audio processing will generate the sonic output of the program. OSC data is parsed and sent to various wavetables, which will store the updated parameter values. These values are then interpolated and sent to a high-pass filter with a low frequency cutoff to remove the DC offset (since we are only interested in fluctuations for generating audio content). The data is finally sent to an audio buffer to be processed at audio rate.

4.1.6 MacOS

Every software component that has been described was implemented and tested on MacOS High Sierra (version 10.13.6).

4.2 Design

Ripple Synth is comprised of several distinct parts, which come together to serve one cohesive purpose: to provide a musical interface that simulates the interaction and behavior of a fluid-like vibrating membrane and sonifying the result using the methods of scanned synthesis. It does this by gathering gestural touch information from the Sensel Morph interface, mapping these gesture parameters to interaction parameters that excite a haptically vibrating mesh structure, and sending visual and audio feedback data of the mesh to the performer, who will in turn respond to the current feedback data. This feedback loop is illustrated in figure 4.1.

4.2.1 Gesture Extraction

The process begins with collecting pressure input data from the Sensel Morph. As was previously mentioned in 4.1.1, the Sensel Morph provides an API for collecting high level gestural content from the realtime touch interaction of the device. The

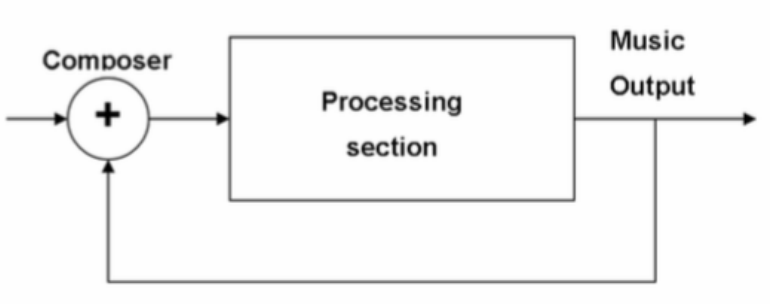

Figure 4.1: Classical model of the feedback loop. [Puig, 2005]

gestural content extracted for use includes the total number, ID, total pressure, and x and y coordinates of each contact point. An illustration of how the Sensel Morph receives contact data can be seen in figure 4.2. The gesture parameters are then

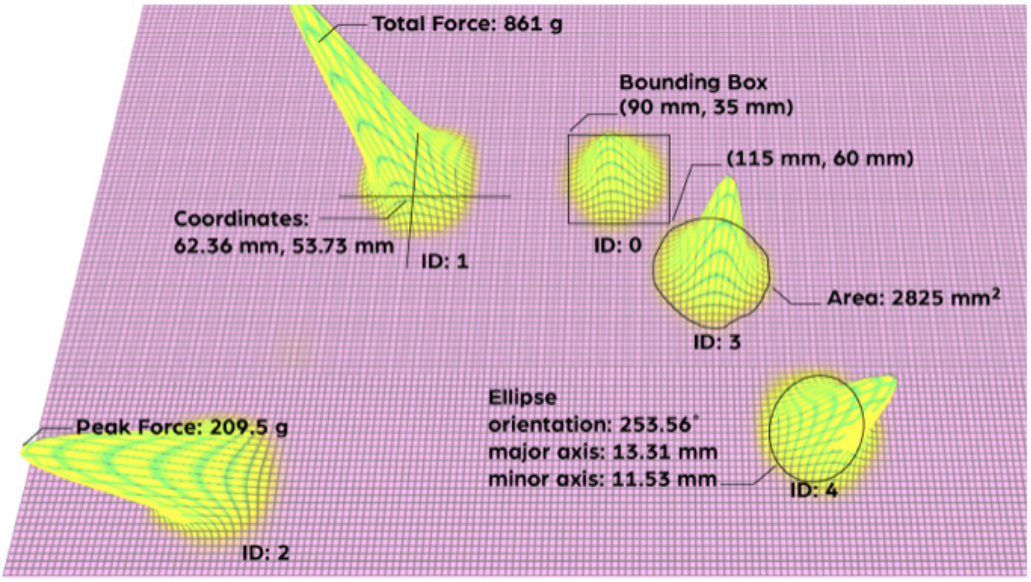

Figure 4.2: A visual description of the type of frame data that can be received on the Sensel Morph Sensel [2018b]

mapped to interact with the mesh simulation.

4.2.2 Mapping Gestures to the Mesh Interface

Gesture interaction with the mesh takes place by first mapping the x and y position of each contact point to its corresponding mass on the two-dimensional mesh grid. Three separate types of contact force are used to instigate a downward force on each gesture-mass contact point, causing mass displacement to initially occur in a downward motion: `DOWN_FORCE`, `GLIDE_FORCE`, and `DOWN_AND_GLIDE_FORCE`. These

purpose of these three force parameters are to simulate the different force interactions that one can make while causing ripples on a fluid surface. `DOWN_FORCE` uses the downward pressure of a contact point to generate ripple vibrations throughout the mesh, the greater pressure there is on a contact point, the more downward force will be exerted on a mass. `GLIDE_FORCE` uses a contact point's velocity to exert downward pressure on the mesh. In short, the strength of the downward force on a mass at the point of contact depends on how fast the user glides their fingers across the surface. This is analogous to gliding one's fingers across a fluid surface, where the strength of the vibrations correspond to the quickness of the glide. `DOWN_AND_GLIDE_FORCE` combines these two force interactions to include both downward and glide force responses. The intensity of the mesh interaction in both downward and horizontal motion will contribute to the overall intensity of the haptic mesh vibrations.

4.2.3 Two-Dimensional Mass Spring Mesh

The behavior of the two-dimensional mass spring mesh network is calculated using the Newtonian laws of motion for the masses and using Hooke's law for the spring force interactions between the connected masses. The downward force displacement is combined with spring force vectors, which combine both a spring and damping constant, to calculate the acceleration force of the mass. Mesh parameter values for mass, spring constant forces where chosen to reflect the fluid motion behavior of the mesh and to ensure that the mesh structure does not become unstable. As a further precaution, an invisible barrier was set up to be boxed around the mesh structure so that mass values do not go beyond a certain point. A similar barrier has been implemented in Tubb's Wablet application [Tubb, 2011]. Once mass acceleration is calculated, the velocity and position of the mass can be obtained by its first and second integral of the mass acceleration with respect to time. Once the position of the masses are calculated, they can be updated graphically to reflect the current state of the mesh in OpenGL.

4.2.4 Graphical Feedback

Every time the state of the mesh is updated, the position of every mass is sent to OpenGL to be rendered. The orbital mass positions are also updated at this point, by keeping track of the indices of these positions and updating them in OpenGL using a seperate function. It is within this function that orbital mass information is collected and transmitted via OSC. This will be discussed in further detail in 4.3.1.

4.2.5 Mapping Orbital Data to the Dynamic Wavetable

The orbital pathway lies within the center of the mesh structure in a square shape as can be seen in figure 4.3. This shape was chosen based on the conclusion found in [Mills and de Souza, 1999], which has been discussed in 2.2.3, and keeps in the spirit of scanned synthesis which is about synthesizing and shaping the behavior of timbres

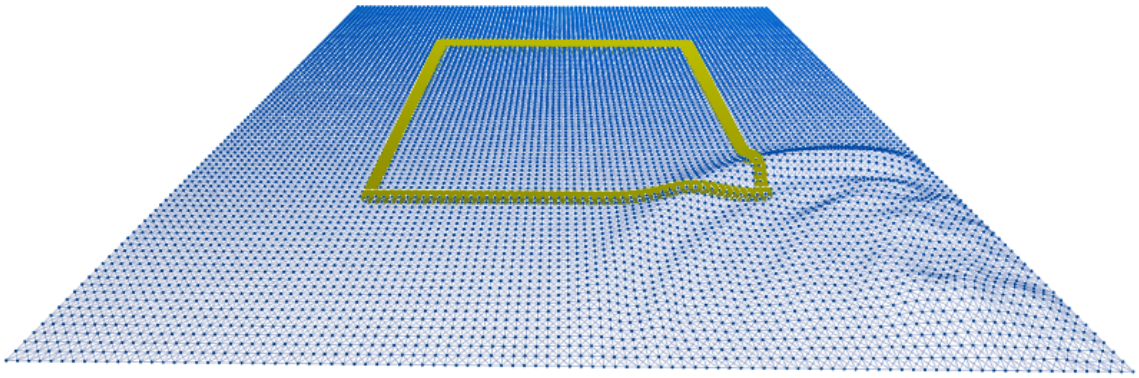

Figure 4.3: The yellow rectangular shape represents orbital pathway in the mesh structure

throughout time. Orbital data is updated along with the mass spring data for the entire mesh network in the same fashion. Once this data is updated graphically and transmitted via OSC, it is received on the JUCE end and orbital position, velocity, and acceleration values for each orbital mass are concurrently stored in an a position, velocity, and acceleration wavetable, respectively, so the wavetables reflect the current state of the orbital behavior. These dynamic wavetables are then processed in a realtime audio buffer, the details of which are discussed in 4.3.1.

4.3 Implementation

This section will describe the implementation details of the Ripple Synth application. This will include an overview of the architecture of the system for both hardware and software and a discussion about the structure of the code base.

4.3.1 System Architecture

The architecture of the system is structured in four distinct parts: gesture parameter extraction on the Sensel Morph, mesh behavior and graphical display in Qt, OSC transmission from Qt to JUCE, and audio extraction from the orbital to JUCE.

Gesture Parameter Extraction on the Sensel Morph

The Sensel Morph API is used within Qt to extract gesture parameter information for mapping to mesh interaction parameters. Input data is first extracted in the `GLWidget` class and `frame` data, which is all the gesture data for one particular time frame, is sent to the `Mesh` class, where `frame` data is parsed and force interaction parameters are applied to the masses.

Mesh Behavior and Graphical Display in Qt

The mesh behavior is dictated by both the force interaction parameters and the characteristics of the mesh itself, which include the weight of the masses and the spring constant and damping forces. Masses and springs behave according to the Newtonian laws of physics, which calculate forces according to $F = m * a$. Spring force is calculated according to Hooke's law $F = k * x$, where x is the position of the mass and k is the spring constant. Damping is calculated according to $D = -d * \vec{v}$, where d is the damping constant and $\vec{v}$ is the vector sum of the velocity of the masses at each end $\vec{v} = \vec{v_1} + \vec{v_2}$. Once the forces are calculated, the mass positions are updated and the new mesh state is rendered and displayed. A single call to render the entire mesh is done, which then calls every spring and mass to render its new position individually. A separate function is called to both render the orbital positions and transmit the orbital behavior via OSC.

OSC Transmission from Qt to JUCE

OSC transmission is set up in Qt by including the `oscpack` API in the Mass class. Orbital masses can then transmit data in OSC packets. This data is sent over the local address 127.0.0.1 over port 9001. Packets are received over the same protocol in JUCE in the `Wavetable` class. In order to receive these packets of data, `Wavetable` must inherit from the `OSCReceiver` class, which contains listener callback functions for receiving OSC messages. `Wavetable` then parses the data and sends them to various wavetables for audio processing.

Audio Extraction from the Orbital to JUCE

Audio data is extracted from `wavetableBuffer`, which is a buffer that loads audio data from one of three different wavetables: `orbitalPosition`, `orbitalVelocity`, `orbitalAcceleration`. These wavetables store updated information about the state of every orbital masses spacial coordinates, velocity, and acceleration. Once this data is loaded in `wavetableBuffer` it then uses an interpolation function which produces linearly interpolated audio data that is then stored in the realtime audio buffer.

4.3.2 Code Structure

The code base can be separated into two separate pieces of functionality. The first runs in Qt, which processes the realtime gesture input from the Sensel Morph, maps the gesture interaction to the mesh, handles the physics calculations of the mesh, updates the state of the mesh and renders the graphics in OpenGL, and finally transmits orbital parameter data to JUCE via OSC. The second runs in JUCE, which receives the orbital parameter data, parses the orbital data and updates the data to various wavetables, chooses one of the wavetables for processing loads it in a wavetable buffer, interpolates the buffer for audio processing, and finally filters the audio to remove the DC offset before being sent to the audio buffer for output.

The code base in Qt is organized in an object-oriented fashion, and can be viewed as having three parts. The first part holds the `Mesh` class, which creates a new mesh and establishes the connections between all of the masses and springs. Masses and springs are defined in the `Mass` class and the `Spring` class. Masses keep track of all the forces that are being applied to them, including spring forces and interaction forces, and also keep track of any barriers that they cannot cross and will bounce back from. They can be locked into position in order to be suspended into space. Springs contain all the spring properties and behavior that allow it to expand and contract according to spring constant and damping forces. Every spring must be connected to one mass at each end. The second part holds the `GLWidget` class, which sets up all of the necessary OpenGL graphics parameters and renders the mesh simulation in realtime. This is where the mesh is initialized and the simulation is ran. This is also where the Sensel Morph interaction data is first extracted. The third part holds the `Window` class. This class simply displays the window where the OpenGL graphics are rendered to.

The code base in JUCE holds one class, the `Wavetable` class, and a Main file for running the JUCE program. The `Wavetable` class inherits from the `AudioAppComponent` class which give the `Wavetable` class the functionality to process realtime audio. This class has three distinct purposes. First, it receives OSC packets and parses the information. Secondly, the parsed data is updated in three separate wavetables, the first holds orbital position coordinates, the second one holds orbital velocity values, and the third one holds orbital acceleration values. Finally, data is sent to a wavetable buffer, which is interpolated using linear interpolation, high pass filtered at 30 Hz, then sent to the audio output buffer.

Chapter 5

Evaluation

This section will discuss the user testing and evaluation of the Ripple Synth application. The evaluation is based on a study which explores how musicians will express themselves with a new DMI, and how different mapping schemes affect the appraisal of the DMI [Pras and Wanderley, 2013]. The evaluation will also use the System Usability Scale in order to measure the usability of the application and its reliability [Brooke, 1995].

5.1 Procedure

There were 10 participants, all with musical backgrounds who had an understanding and were familiar with using a musical instrument. They averaged 7.2 years of musical experience, with 5 being most familiar with the piano. Four were said to have been quite familiar with digital musical instruments. One of the participants had a familiarity with scanned synthesis and 2 had experience with using the Sensel Morph.

The experiment consisted of 3 sessions, each session contained a different mapping setup. Each mapping tested a different gesture parameter, which was mapped to the same audio extraction parameter. The three gesture parameters consisted of "down force", "glide force", and "down and glide force". The audio extraction parameter was the audio produced from the velocity movement of the orbital. The experiment took roughly one hour to complete for each participant.

In each session, each participant was asked to watch 3 videos corresponding to three different mesh animations, and were asked to try to reproduce the animation. The video included the audio produced from the excited mesh animations, so the participant knows the sounds they are expected to reproduce as well. This was followed by a short interview. The participant was than given about 10 minutes to experiment with the interface. This was followed by another interview.

The replication of the task corresponding to the 3 videos for each task differed according to the session. The animations were chosen according to the uniqueness of the gestures produced for each mapping. The mapping scheme for each session was

chosen in random order, as to were the three tasks as to avoid any ordering bias. For the "down force" mapping scheme, the participant was shown the following three animations: a single ripple produced in the center, a downward gliding ripple with increasing downward force moving at a constant velocity, and two gliding ripples moving in opposite directions with increasing downward force moving at a constant velocity. The last animation was chosen in order to encourage the participant to use two hands to produce the animation. For the "glide force" mapping scheme, the participant was shown: a gliding ripple with increasing velocity and downward force, a gliding ripple with decreasing velocity and downward force, and two opposite gliding ripples with increasing velocity and downward force. The "down and glide force" scheme included: a gliding ripple with increasing velocity and decreasing downward force, a gliding ripple with decreasing velocity and increasing downward force, and two opposite gliding ripples with increasing velocity and decreasing downward force. Participants were given as much time as needed to perform tasks and were given hints as how to vary the downward force or speed of the gesture interaction in order to match the animation and sound.

The questions during the interview portion of the session were asked in order to gather qualitative data of the participant's ability to experiment with the instrument and to know their mapping preference. These questions were modelled after [Pras and Wanderley, 2013], in which questions were carefully worded in order to avoid provoking a certain answer. During the first session, the following question was asked: *How would you describe your progression throughout the 3 tasks?* After the participant explored the interface for 10 minutes, the two following questions were asked: *How do you feel about the music you just created? How was your experience playing the instrument?* During session two and three, the interface exploration interview questions were changed to: *This time, how do you feel about the music you just created? How would you compare your experience playing the instrument in this session versus the last session? Which set–up would you like to spend more time on in the future?* The last question opens up suggestions for possibly improving the mapping strategy.

5.1.1 Analysis

Grounded theory was chosen to analyze the qualitative data, and categorize the reemerging concepts from this verbal data. A System Usability Test was given in order to assess the overall usability of the application.

Chapter 6

Results and Discussion

6.1 Qualitative Results

6.1.1 Task Replication

Analysis of the data revealed two main categories that were identified using Grounded Theory: learning and mapping. Seven participants found the learning curve quite challenging. It took several suggestions in order to guide them in the right direction of interaction. In terms of mapping, the overall gesture movement wasn't deemed confusing, but the nuances between mappings were hard for them to distinguish. In general, the different gesture mappings were hard to figure out at first. The small amount of time spent between tasks meant that a learning curve could not be established.

6.1.2 Instrument Exploration

Two categories were also identified for the exploration phase of the interview: sound quality and gesture response. Eight of the participants commented on the sound quality of the output as not being very musically pleasing. Seven made comments about the difficulty of distinguishing between the dynamics of the sound, other than the intensity from soft to loud. Overall, the sound quality was judged to be too harsh for the participants. Pertaining to the gesture response, four of the ten participants actually felt the nuanced behavior to be counter intuitive to what they expected. For instance, pressing and holding down on the mesh should not hold the displacement of a mass, but bounce back, much like sticking your finger in water where you poke through the water, rather than poking and holding the displacement of water. Six felt that they had a great deal of control at the beginning, but the vibrations became too overwhelming to notice their own gesture response.

6.2 System Usability Test

In order to evaluate the overall usability of the application as a whole, the System Usability Test was conducted based on [Brooke, 1995]. Questions were arranged on a 5-point scale, from 1 being strongly disagree to 5 being strongly agree. The results were then calculated with the Usability Score. This resulted in a score of 51.8, which was way than the average score of 68. This means that improvements to the usability of the system can *must* be made in order for it to even qualify as 'usable'. The following shows the mean scores to the test:

System Usability Test Question	Mean Score
I think that I would like to use this system frequently.	3.4
I found the system unnecessarily complex.	2.2
I thought the system was easy to use.	3.0
I think that I would need the support of a technical person to be able to use this system.	2.8
I found the various functions in this system were well integrated.	2.9
I thought there was too much inconsistency in this system.	3.1
I would imagine that most people would learn to use this system very quickly.	2.8
I found the system very cumbersome to use.	3.2
I felt very confident using the system.	2.6
I needed to learn a lot of things before I could get going with this system.	2.5

6.3 Discussion

The low System Usability Score reflects the general observations and responses that were made during the interview. This ultimately shows that there were fundamental flaws in the design that need to be addressed before testing an application that is deemed usable. Much of these issues pertain to the gesture response of the system and its overall intuitiveness.

Participants were able to distinguish differences between the mappings and did have preferences. General comments made about the "down and glide" mapping showed that it felt the most engaging to participants. Two participants commented about the aggressiveness with which they were allowed to play, which excited the mesh in very dynamic ways. Three participants, on the contrary, felt that this aggressive nature provoked the mesh to react in uncontrollable ways, and had showed preferences for the other mappings, namely the "glide" mapping.

Eight of the participants did not find the sonic feedback very favorable, or even informative. The harsh nature of the sound was generally off putting. The fact that most of the participants were not able to distinguish differences in the sounds,

and found little correlation between the visual and audio feedback other than the intensity of the vibrations, means that there are fundamental mapping flaws in the orbital extraction design that need to be addressed.

Chapter 7

Conclusion and Future Work

New forms of creative expression have the potential to unlock new paradigms in how we communicate ideas. Musicians and artists will always strive to be at the cutting edge of original thought and unique expression. One way to uncover such potential is to devise new mechanisms from which we communicate. As was stated by Puig, a definitive means to create unique forms art, namely music, is to create new instruments of expression that can contribute to musical performance in original and innovative ways [Puig, 2005].

Scanned synthesis is a relatively new technique in the world of audio synthesis. It's scantly tested waters, which have left much to explore in terms of developing and testing novel forms of interaction and devising new audio synthesis techniques, makes it a prime candidate for the development of a new digital musical interface. In an attempt to test these waters, a novel interaction technique for scanned synthesis was designed, implemented, and tested in order to evaluate if this gestural mapping strategy added expressive value to the experience of playing a new digital musical instrument. This gestural interaction behavior was inspired by interactions with fluid-like membranes such as the ones found in [Erlach et al., 2011] and [Tubb, 2011], and was expanded upon. Based on the qualitative data and the System Usability Score, however, it cannot be stated that this novel gesture interaction will bring expressive value to scanned synthesis. Fundamental design issues must be addressed before further evaluation can take place.

7.1 Future Work

The evaluation of Ripple Synth has exposed fundamental issues that need to be addressed before it can proceed. The low System Usability Score reflects the general response to the application which mentions the issues with being able to be in control of the gesture response and issues with gesture intuitiveness. In order to address these issues, new mapping strategies must be devised. These strategies have to carefully consider gesture intuitiveness and how it relates to the response of the mesh structure.

To address the mesh response behavior issue, the properties of the mesh behav-

ior should be redesigned to allow a more natural fluid-like response to the gesture interactions. One possibility is to do away with the crude *lump* design model, which works on basic Newtonian principles and Hooke's law for masses and springs, and to develop more sophisticated algorithms that are possibly based on fluid motion behavior. The *distributed* model will suffice for such a method and will also allow for efficient algorithms and complex haptic motion response.

The audio extraction methodology must also be addressed. First and foremost, linear interpolation is a crude method for generating audio data from wavetables, and it is one that will inevitably cause artifacts to be present in the sound. Better interpolation methods could be used, such as cubic interpolation to improve this issue. Interpolation between wavetables in time can also be a way to avoid instantaneous jumps and clips that will cause artifacts as well. A similar issue was addressed in [Tubb, 2011]. Finally, different extraction methods could be used. Again, [Tubb, 2011] addresses parts of this issue experimenting with different extraction techniques.

None of the participants noticed any lag or latency in the system, and actually one participant commented on how fast the response was. However, as was previously stated, improvements can still be made in terms of moving away from the *lump* model to a more efficient *distributed* design.

Aside from the fixing the general design issues, one future development would be to reduce the number of technological dependencies of the application. One goal is to develop the application to work entirely on the JUCE end, and remove Qt from the system. JUCE also has OpenGL functionality and is therefore able to render 3D animations on its own. Having the program run entirely in JUCE will remove also remove the dependency on OSC to transmit messages. This will reduce latency even more since there is no time spent transmitting packets of data from one program to another in order to update wavetables in realtime. The reason why the application was not developed in JUCE entirely was because the API for rendering visual data in OpenGL was much more formidable and difficult to use than it was in Qt, and due to time constraints Qt was chosen.

Another goal in development would be to continue to experiment with interaction mechanisms and audio extraction algorithms for the application. There is an endless opportunity to explore both intuitive and exotic behavior in scanned synthesis, and much of this future potential work is hinted at in the research that has already been conducted on scanned synthesis [Verplank et al., 2000] [Boulanger et al., 2000] [Tubb, 2011].

Bibliography

Bencina, R. (2018). oscpack - a simple C++ OSC packet manipulation library | Ross Bencina. http://www.rossbencina.com/code/oscpack.

Bilbao, S. D. (2001). *Wave and Scattering Methods for the Numerical Integration of Partial Differential Equations*. PhD thesis, Stanford University.

Boulanger, R., Smaragdis, P., and ffitch, J. (2000). Scanned Synthesis: An Introduction and Demonstration of a New Synthesis and Signal Processing Technique. In *ICMC*.

Boulanger Labs (2018). screen baton. http://boulangerlabs.com/wp-content/uploads/2017/03/6screen_baton-400x300.jpg.

Brooke, J. (1995). Sus: A quick and dirty usability scale. 189.

Duyne, S. A. V. and III, J. O. S. (1993). Physical Modeling with the 2-D Digital Waveguide mesh.

Erlach, B., March Evans, and Michael J. Wilson (2011). Tüb - Interactive Sonification of Water Waves.

Guy, R. L. (2018). Mass-Spring System Simulation | Ryan Guy's Portfolio. http://rlguy.com/massspringsimulation/.

Humanoid Sound Systems (2018). Scanned Synth Pro - Humanoid Sound Systems - VST Plugins, VSTi, & Audio Units Plugin Synth Music Software and Soundware. http://www.humanoidsoundsystems.com/scanned-synth-pro-vst-au-synth-plugin/.

James, S. G. (2005). Developing a Flexible and Expressive Realtime Polyphonic Wave Terrain Synthesis Instrument Based on a Visual and Multidimensional Methodology. Master's thesis, Edith Cowan University.

JUCE (2018). JUCE | JUCE. https://juce.com.

Mathews, M. (1998). Max Mathews - Lecture on Scanned Synthesis - 1998. https://vimeo.com/40910330.

Mathews, M. (2000). RADIO-BATON INSTRUCTION MANUAL. http://www.csounds.com/wp-content/uploads/2013/05/BatonManual.pdf.

McGlynn, P. (2014). *Interaction Design for Digital Musical Instruments*. PhD thesis, National University of Ireland, Maynooth.

Mills, A. and de Souza, R. C. (1999). Gestural sounds by means of wave terrain synthesis. *Proc. of the VI Brazilian Symp. on Computer Music*, 3:9–16.

Mitsuhashi, Y. (1982). Audio Synthesis by Functions of Two Variables. *Journal of the Audio Egineering Society*, 30(10):701–706.

Morse, P. M. and Ingard, K. U. (1968). *Theoretical Acoustics*. McGraw-Hill, New York, NY, USA.

Nagashima, Y. (2004). Controlling Scanned Synthesis by Body Operation.

Open Sound Control (2018). Introduction to OSC | opensoundcontrol.org. http://opensoundcontrol.org/introduction-osc.

OpenGL (2018). Opengl Overview. https://www.opengl.org/about/.

Pras, M. G. A. and Wanderley, M. (2013). Combining musical tasks and improvisation in evaluating novel Digital Musical Instruments . In *Proc. of the 10th International Symposium on Computer Music Multidisciplinary Research,*.

Puig, S. J. (2005). *Digital Lutherie: Crafting musical computers for new musics' performance and improvisation*. PhD thesis, Universitat Pompeu Fabra.

Qt (2018a). Qt | Cross-platform software development for embedded desktop. https://www.qt.io.

Qt (2018b). Qt OpenGL C++ Classes | Qt OpenGL. http://doc.qt.io/qt-5/qtopengl-module.html.

Qu-Bit Electronix (2018). Coming Soon - Qu-Bit Electronix. http://www.qubitelectronix.com/soon/.

Roads, C. (1996). *The Computer Music Tutorial*. MIT Press, Cambridge, MA, USA.

Sensel (2018a). GitHub - sensel/sensel-api: Sensel API for communicating with Sensel devices. https://github.com/sensel/sensel-api.

Sensel (2018b). Morph - Sensel Morph Documentation. http://guide.sensel.com/morph/.

Smaragdis, P. (1999a). Csound Opcodes for Scanned Syntheeis. http://www.csounds.com/scanned/scannedman.html.

Smaragdis, P. (1999b). scansyn. http://www.csounds.com/scanned/zip/scancode.zip.

Smith, J. O. (1992). Physical modeling using digital waveguides. *Computer Music Journal*, 16(4):74–91.

Smith, J. O. (1996). Physical modeling synthesis update. *Computer Music Journal*, 20(2):44–56.

Tissieres, J. Y., Vaseileiou, A., Zabetian, S., Dahl, S., and Serafin, S. (2018). An Expressive Multidimensional Physical Modelling Percussion Instrument. *Proceedings of the 15th Sound and Music Computing Conference 2018*.

Tubb, R. H. (2011). An Investigation into Scanned Synthesis and Multi-Touch. Master's thesis, Queen Mary University of London.

Verplank, B., Mathews, M., and Shaw, R. (2000). Scanned Synthesis. In *Proceedings of the International Computer Music Conference*.

Wanderley, M. M. and Depalle, P. (2004). Gestural Control of Sound Synthesis. *Proceedings of the IEEE*, 92(4):632–644.