WITH A WAVE OF THE HAND: CREATING GESTURE CONTROLLED SOUNDSCAPES AND VIDEO WITH THE LEAP MOTION CONTROLLER AND MAX/MSP/JITTER

By

THOMAS DOUGHTY

SUPERVISORY COMMITTEE:

Patrick Pagano, Chair

Angelos Barmpoutis, Member

James Paul Sain, Member

A PROJECT IN LIEU OF THESIS PRESENTED TO THE COLLEGE OF FINE ARTS OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ARTS

UNIVERSITY OF FLORIDA

2015

To My Parents, Lianne, and Nico.

ACKNOWLEDGMENTS

I would to thank my committee members for their guidance and inspiration. The support of the Gage Family, by providing me with a “thesis retreat”, allowed me to see this project to its conclusion. I would also like to thank the members of the Cycling ‘74 forums for leading me out of (and back down) the Max rabbit hole.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

GCPT	Gesture Controlled Performance Tool. An interactive control space designed to perform and present audio and visual projections via cues and hand gestures captured by the Leap Motion Controller. This tool was realized in the Max/MSP/Jitter visual programming environment.
OSC	Open Sound Control is a protocol for communication among computers, sound synthesizers, USB and wireless controllers, projection mapping software, and other multimedia devices.

Summary of Project in Lieu of Thesis
Presented to the College of Fine Arts of the University of Florida
In Partial Fulfillment of the Requirements for the
Degree of Master of Arts

WITH A WAVE OF THE HAND: CREATING GESTURE CONTROLLED SOUNDSCAPES WITH THE LEAP MOTION CONTROLLER AND MAX/MSP/JITTER

By

Thomas Doughty

December 2015

Chair: Patrick Pagano
Major: Digital Arts and Sciences

Performing with commercially available gesture control devices is an exciting new field of musical expression and creative performance. For as long as a conductors have been waving batons to guide and cue groups of musicians there has been gesture controlled musical performances. Since the advent of electronic and digital technology there has been increasing interest in utilizing alternative methods to conduct and influence music with expressive gestures. Through the latter half of the twentieth century and into the twenty-first century artists have been engineering and implementing new and creative tools which expand the reach of a conductor's and musician's ability to control and enhance musical expression.

This project in lieu of thesis begins with an examination of the historical precedence of, and current trends in, gesture controlled music. The author will survey and discuss the significance of past gesture controllers designed for musical

applications, as well as other gesture controllers that have been adapted to conduct and perform music.

The main focus of the project in lieu of thesis centers on the creation of a Gesture Controlled Performance Tool that can be programmed to cue events and instruments, as well as modulate sounds and trigger effects. The Gesture Controlled Performance Tool can also be utilized to cue and effect streams of video or live camera feeds to enhance and complement a musical performance.

The author makes use of a recent technology that tracks the fine motor movements of a user's hands and interpolates those gestures into myriad streams of data to manipulate almost any desired parameter. The Leap Motion Controller is an affordable gesture tracking technology that can output tracking data of hands, fingers, palms, and even simple tools. Hand gestures are interpreted into x, y, and z axis positions, the tilt and turn of the hands, velocities, pinch and grab strength, and many other tracking points. The vast amount of data creates an equally vast amount of potential for manipulating audio signals and conducting events for a musical performance. This data is ported to Max/MSP/Jitter software, via the *leapmotion* external object, where it can be routed and scaled and used to (a) generate sound synthesis, (b) effect video feeds, and (c) cue and control an array of sound modulating effects and performance features.

The Gesture Controlled Performance Tool is a framework that makes using the Leap Motion to perform complex audio/visual performances within the Max environment easy, customizable, and effective. Eliminating the complex processes of routing and scaling the data from the Leap Motion Controller allows an artist the freedom to create and compose their performance and concentrate on content rather than programming. The author will present the underlying structure of the Gesture Controlled Performance Tool and provide instructions for its use. Methods for extracting Leap Motion performance data for analysis with programs like Microsoft Excel will also be discussed.

CHAPTER 1
A SURVEY OF GESTURE CONTROLLERS PAST AND PRESENT

Introduction

Curtis Roads points out, in his book *The Computer Music Tutorial*, “the original remote controller for music is the conductor’s baton” (Roads, 1996). This analogy implies that performer’s eyes were the first gesture-tracking device to translate a conductor’s gestures into musical expressions. In this scenario a gesture is a movement with meaning and intention. The performer translates the gesture and transforms it into a musical expression. Today’s composers and performance artists can take advantage of new technologies to expand their gestural expressions beyond traditional means. This exciting new field of composition and interaction has a rich history, which has brought about advances in the technology of commercially available and economically viable gesture capturing devices for a great many artists and programmers. The following section will provide a brief overview of a few important technologies past and present that can track a performer’s gestures.

Electric Metronome

There are many ways to capture gestures and translate them into controls over musical performance parameters and audio signal manipulation. The first electromechanical remote controller was used by the conductor Hector Berlioz to signal musical tempo to an offstage chorus. An electronic key, when pressed, would turn a light on or off to signal the change in tempo (Roads, 1996). Berlioz wrote to Franz Liszt calling his electric metronome an “immense musical mammoth” (Austin and Monir, Web). Certainly, Berlioz would marvel at the advancements made in gesture tracking technology.

Theremin

The Theremin is an early electronic instrument performed without physical contact. The instrument has two antennas that are sensitive to the positions of the musician’s hands over the instrument. One hand controls the pitch range (the frequency of an oscillator), while the other hand controls the amplitude (the basic core of electronic sound synthesis). The Theremin was invented in 1920 (“Theremin”, n.d.) and, because of its unique sound and novelty, has experienced periods of great popularity in accentuating science fiction and horror films with acousmatic sound. The Theremin’s novelty and difficult playing technique has also been a hindrance to it being accepted like a more conventional instrument. Thanks to a few select manufacturers, namely Moog Music of synthesizer fame, the Theremin is still available to those musicians with an adventurous spirit.

The Radio Baton

Max Mathews invented the Radio-Baton and its companion program Conductor to conduct and interpret traditional scores. This system consists of two radio-transmitting batons, which are tracked above a flat sensitive plane. The Radio Baton uses a coordinate system of radio receivers to determine it’s position and sends those values to a control computer (Johannsen and Nakra, 2010). This innovative technology did not receive a great deal of attention in the musical world. Its specialized hardware and computer program were not broadly available to composers and performers. Perhaps another set back for the Radio Baton was that it required the baton’s movements to be above the relatively limited surface area, thereby making the traditional gestures of a conductor feel constrained and unnatural.

The Digital Baton and the Conductor's Jacket

A team of MIT researchers designed and built a Digital Baton sensor to conduct a MIDI- based music system for a musical composition by Tod Machover. This device was hand held like a traditional conductor' baton and was used to trigger and shape layers of sound in the performance of the piece, entitled *Brain Opera.* The technology utilized a pulse-modulated infrared tracking system, pressure detection, and acceleration sensing. Though it was reported to be robust in certain areas, the team concluded that an additional device be created to make up for some deficiencies. The Conductor's Jacket was realized to capture not only gesture, but also physiological data such as muscle tension, heart rate, and temperature. As a research tool the Digital Baton and Conductor's Jacket can yield a remarkable amount of data (Johannsen and Nakra, 2010). However, this technology is highly specialized and inaccessible, making it impractical to performers and composers.

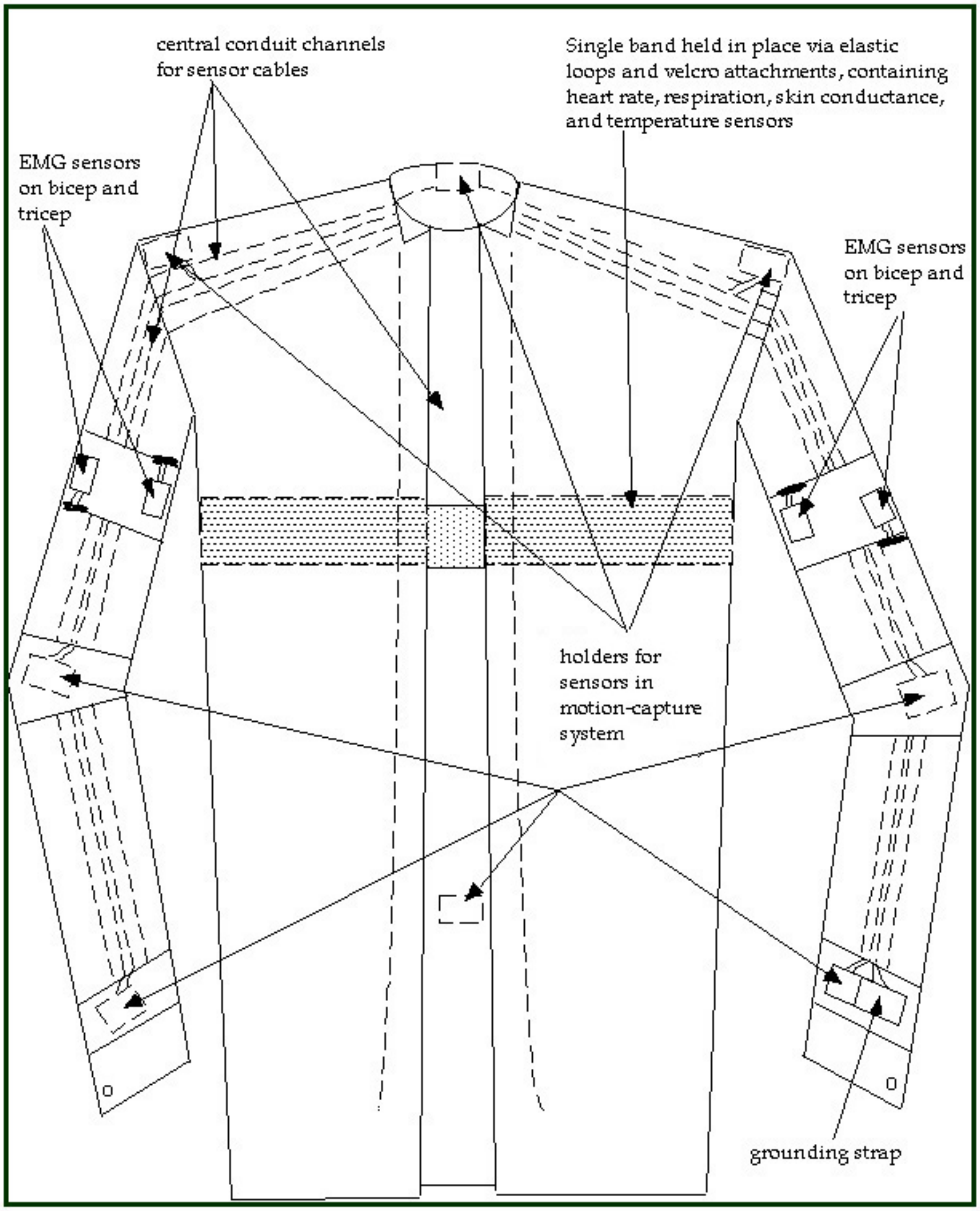

central conduit channels for sensor cables
Single band held in place via elastic loops and velcro attachments, containing heart rate, respiration, skin conductance, and temperature sensors
EMG sensors on bicep and tricep
EMG sensors on bicep and tricep
holders for sensors in motion-capture system
grounding strap

Figure 1.1. Conductor's Jacket.

Wii Remote

The Nintendo Company developed the Wii Remote for use with its Wii video game console is widely available and affordable. The Wii uses an infrared sensor and an accelerometer to capture gestural information of hand and body movements. This hand held device can be perceived by programs like Max to control music and other performance interactions. This device is similar in size to a standard television remote and also has several buttons that can be used to trigger events. Since this device is hand held it is especially useful for mimicking a conductor's arm gestures and the playing of instruments, particularly drums. Two Wii Remotes were used in the composition *Imaginary Dialogues for Two Wiimotes and a Laptop* at the Tufts University Electro/New Music Ensemble's concert on 2008. Composers Phil Acimovic and Paul Lehrman performed their electroacoustic piece facing each other while gesturing towards a computer with the Wii Remotes to trigger sound events and synthesis (Lehrman, Web).

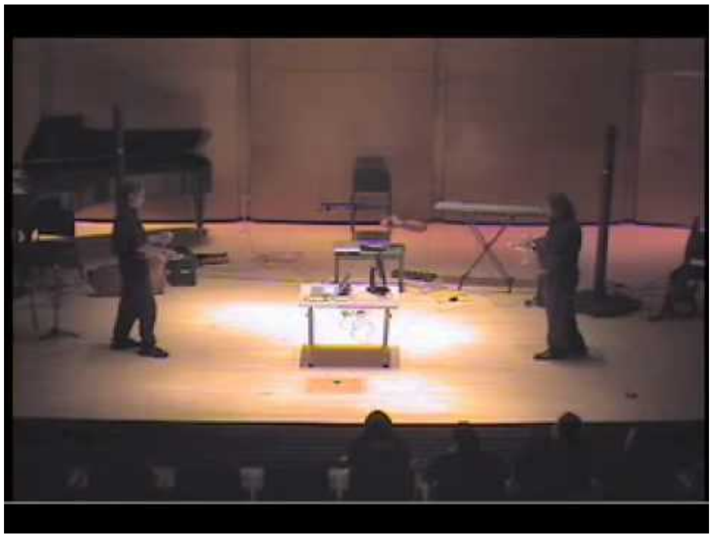


Figure 1.2. Wii performance.

Kinect

The Kinect, developed by Microsoft for their Xbox line of video game consoles, is a full body motion capture device. Like the Wii Remote the Kinect is commercially available and can be purchased at an economical price. The device is a horizontal bar that features a RGB camera and an infrared depth sensor to track body movements (“Kinect”, n.d.). The Kinect excels at tracking large body movements like waving the arms or leaning side to side and can track several performers at once. Through software like SimpleKinect, by Jon Bellona, to interface with a music program the Kinect transmits Open Sound Control (OSC) messages from its console (Bellona, Web). Since the Kinect tracks limb and trunk positions, this device, when used with music software, is best used for dramatic gestural cues or for choreographed sequences. To initiate reliable tracking, a performer must strike the “cactus” pose; standing straight with arms lifted to side and slightly above the head, which can be visually awkward during a performance. However the Kinect has been widely used to produce interactive performances and installations that can enable users to effect sound and video. One example, *Human Chimes* (Bellona, Web), is an art installation that transforms participant’s positioning into chime like sounds that are panned through the installation space. As more people enter into the field of view of the Kinect the sounds begin to interact, mirroring their position. Visuals corresponding to the participant’s position are projected to give them a sense of relationship to the mapped sounds.

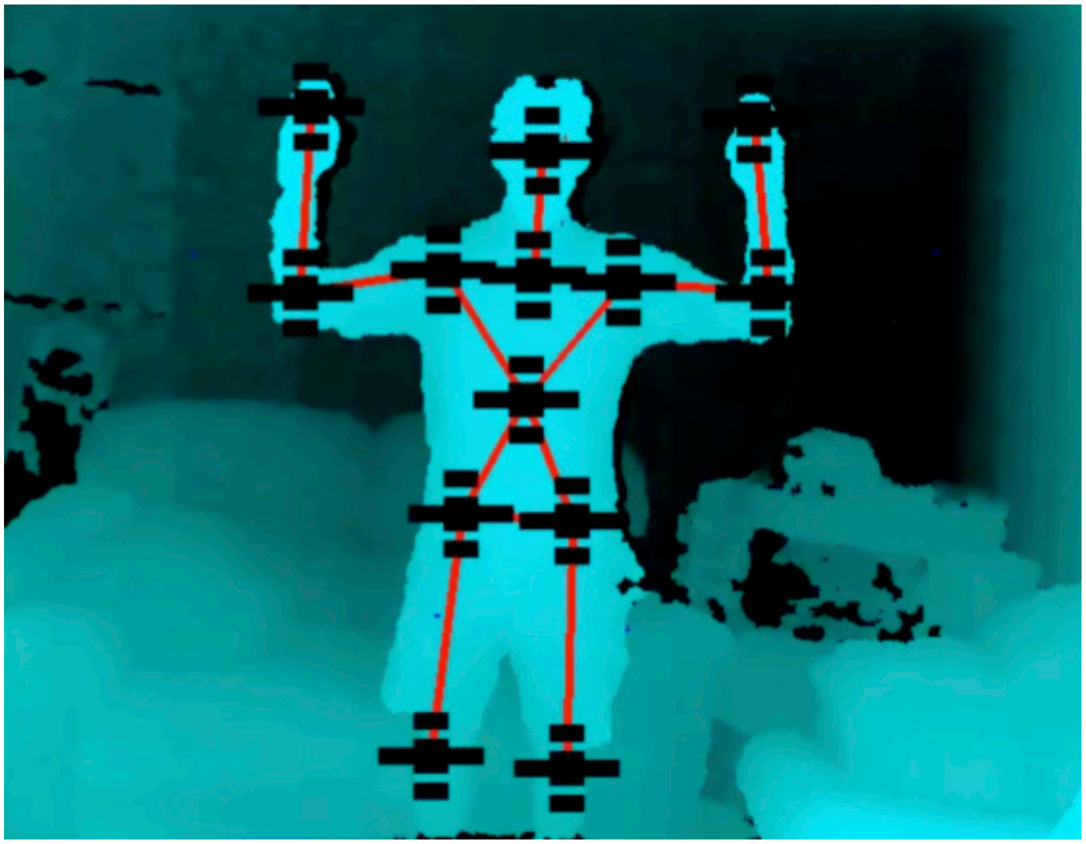


Figure 1.3. Kinect cactus pose.

CHAPTER 2
THE LEAP MOTION CONTROLLER

Leap Motion

The Leap Motion Controller is a USB peripheral device that tracks detailed hand movements and handheld tools. The Leap Motion, when paired with a computer, is a supplementary input device that compliments the user's traditional mouse and keyboard. The Leap Motion was developed by Leap Motion, Inc. and released in 2013, adding to the list of current motion tracking devices that can be utilized to enhance musical expression.

The Leap Motion obtains its fine motor tracking ability through its hardware, which consists of two fish-eye lens cameras and three infrared light-emitting diodes (LEDs). The cameras track the infrared light and the device sends the data to the Leap Motion software via the USB. According to Leap Motion's website (https://www.leapmotion.com), the interactive space captured by the Leap Motion Controller is approximately eight cubic feet. The viewing range is two feet above, two feet wide on each side, and two feet deep on each side. The viewing area is dependent on the strength of the LED lights, whose power is drawn from the USB connection. The strongest resolution and the most robust tracking data comes from interacting close to the Leap Motion Controller. This is due to the increased strength of the LED lights when interacting close, which then increases resolution of the cameras.

After the Leap Motion software receives the sensor data, a 3D representation of what the cameras are picking up is created. The 3D data is then interpreted by tracking algorithms and sends the results of all the data into a transport protocol whereby all of the tracked data is continuously streamed and updated in real-time. From here, any number of stand alone software applications designed for the Leap Motion and utilize OSC protocols to receive the streaming data from the Leap Motion Controller software can be developed, including hand gesture musical performance instruments.

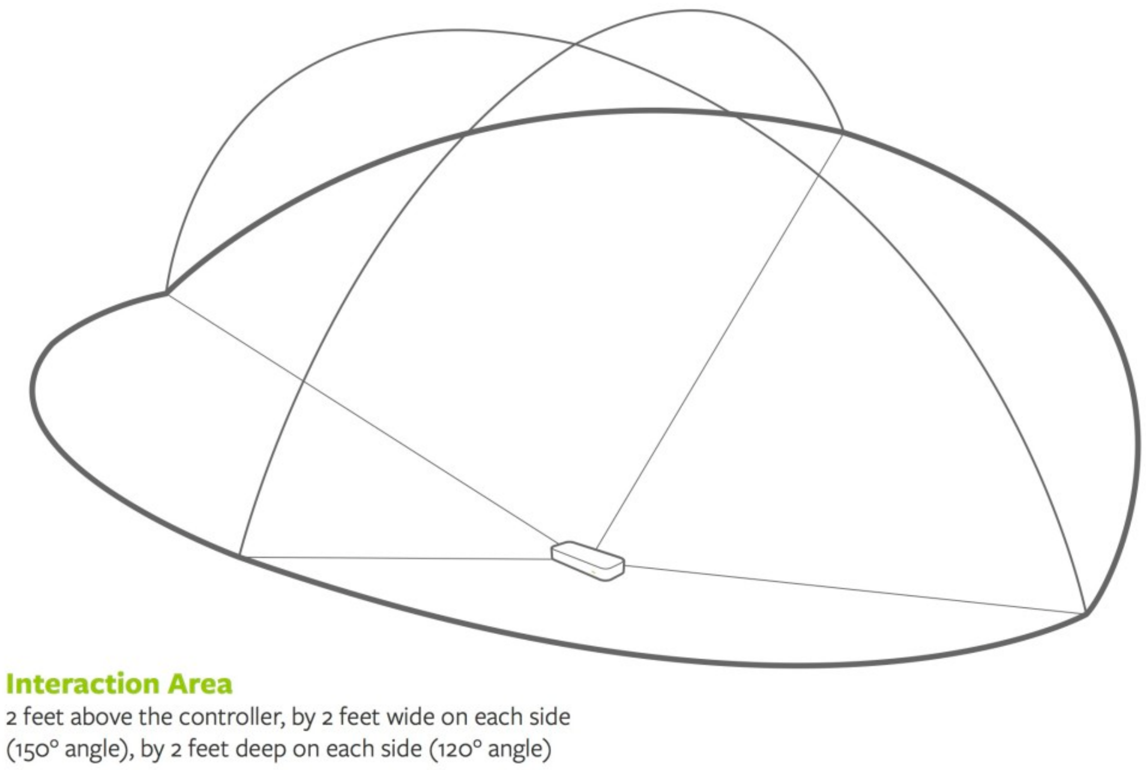

Interaction Area
2 feet above the controller, by 2 feet wide on each side (150° angle), by 2 feet deep on each side (120° angle)

Figure 2.1. Leap Motion area of interactivity.

Leap Motion OSC Applications and Max Third Party Externals

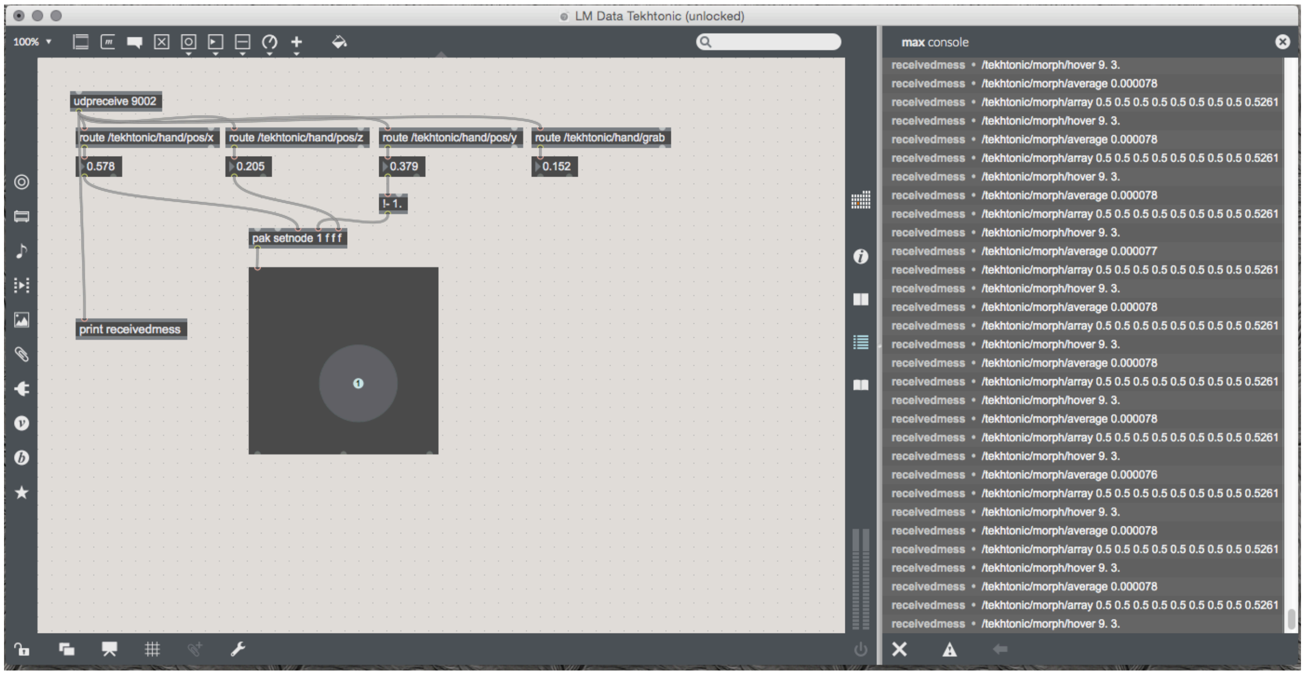
For the purposes of this project, the author required that the data acquired by the Leap Motion Controller be transported into the Max programming environment. A concise introduction to Max will be found in the next chapter. There are two main strategies for importing data from peripheral devices and controllers into Max: (1) OSC, or (2) a third party external object that has been created specifically for the Max environment.

The first strategy makes use of OSC protocol. OSC is a protocol for communication among computers, sound synthesizers, USB and wireless controllers, projection mapping software, and other multimedia devices (“Open Sound Control”, n.d.). Many performers and artists take advantage of OSC compatible devices and software to expand their control of expression and open up new possibilities of interaction. For Max to communicate with and receive data from the Leap Motion Controller, a connection must be made to the host network and port number. This can be done with the udpreceive object. Once this is completed, Leap Motion data can be parsed and routed to where the user deems necessary. This method requires additional software to be launched and running to interpret the Leap Motion Controller’s data. Running multiple applications during a live performance increases the usage of the computer’s processing unit and impacts the computer’s ability to process data smoothly and efficiently, increasing the risk of data lag and software crashes. However, OSC must be considered as existing Leap Motion applications have a high degree of refinement and user friendly interfaces and allow for streamlined data acquisition and appropriation to the desired function.

The second strategy is the use a third party external object created by a developer from the Max community that can import data from the Leap Motion Controller software directly into Max. External objects are an important part of Max. Externals can compartmentalize complex routines that users and developers often use. Operating the external object reduces the amount of programming necessary to achieve common tasks. External objects can also be created to integrate new technologies or functionality available in other software into the Max environment, as is the case with the Leap Motion Controller.

Leap Motion OSC Applications

Leap Motion's app store has four commercially available OSC applications that can be used to transport Leap Motion Controller data into Max. Geco, ManOSC, OSC Motion, and Tekhtonic. The author tested each of the applications for importing data from the Leap Motion into Max. Unfortunately, each application shared the same limitation, namely that a user would have to run the application parallel with Max and consume additional processing power. The Geco application was distinctly designed for use with digital audio workstations, like Abelton Live, rather than a programming environment like Max. Tests with ManOSC with Max resulted in very large data sets that were more applicable for use in Max, yet the application would frequently stop polling and either lag or stop receiving data altogether, making it unacceptable for a live interactive performance. The Tekhtonic OSC test with Max proved to be more easily parsed and comparatively more stable than ManOSC. Unfortunately, Tekhtonic will only receive messages from one hand. Based on the test results, the author had to use a Max third party external object to make use of the Leap Motion's data stream.


LM Data Tekhtonic (unlocked)
100%
udpreceive 9002
route /tekhtonic/hand/pos/x
route /tekhtonic/hand/pos/z
route /tekhtonic/hand/pos/y
route /tekhtonic/hand/grab
0.578
0.205
0.379
0.152
!- 1.
pak setnode 1 f f f
print receivedmess
1
max console

```
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000077
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000076
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
receivedmess • /tekhtonic/morph/average 0.000078
receivedmess • /tekhtonic/morph/array 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5261
receivedmess • /tekhtonic/morph/hover 9. 3.
```

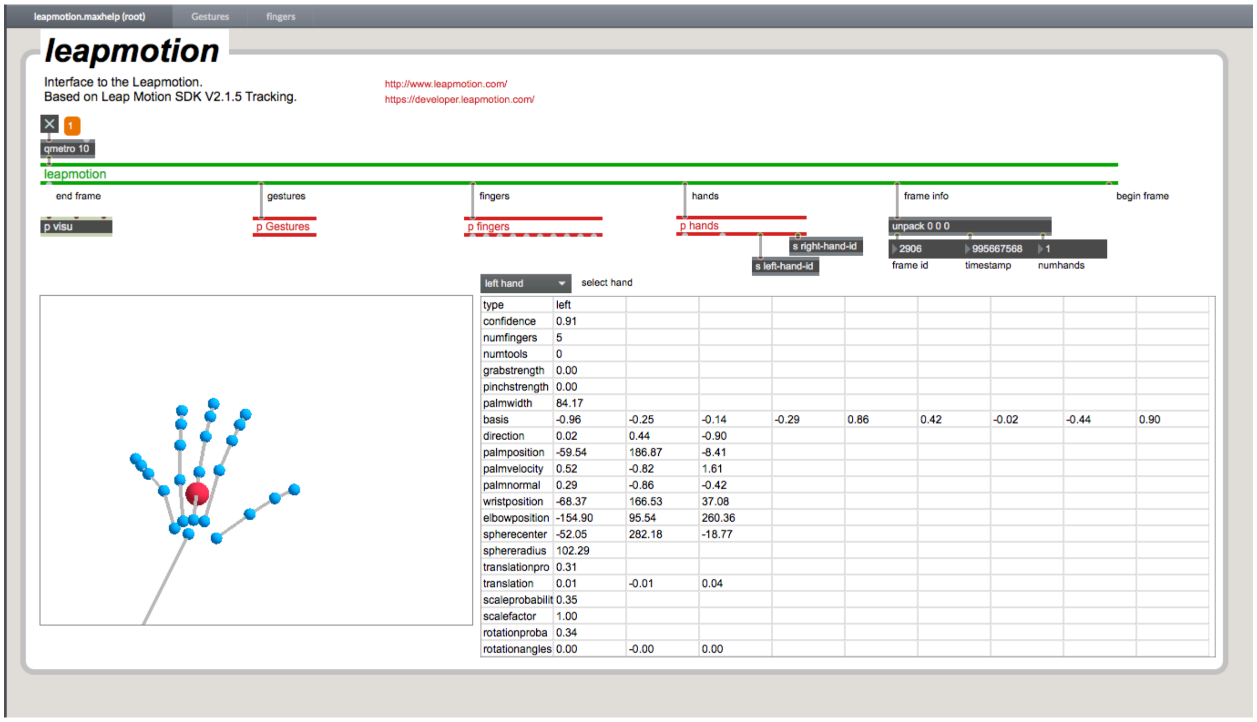
Figure 2.2. udpreceive object.

Leapmotion Object

The leapmotion external object was developed at IRCAM (Institute for Research and Coordination Acoustic / Music) the pioneering institution of music and sound science located in Paris, France ("IRCAM", n.d.). Inspired by a parent object, aka.leapmotion by Masayuki Akamatsu, IRCAM's version utilizes a new tracking API(Application Program Interface) and is therefore more a applicable to the latest software of the Leap Motion (IRCAM, 2014).

The Max help file for the leapmotion object reveals pre-routed Leap Motion tracking data for both left and right hands, fingers, and gesture recognition. There is also full skeletal tracking to help visualize the data and compare in real-time to the translation of physical hand gestures to the Max program. These features proved to very beneficial for conducting research into learning about the capabilities of the Leap

Motion and integrating its data into a sound synthesis program. Of all the methods that the author experimented with the leapmotion object had the least at amount of latency (interruptions of the data stream) and the real-time tracking was robust and reliable.



Interface to the Leapmotion.
Based on Leap Motion SDK V2.1.5 Tracking.

http://www.leapmotion.com/
https://developer.leapmotion.com/

type	left								
confidence	0.91								
numfingers	5								
numtools	0								
grabstrength	0.00								
pinchstrength	0.00								
palmwidth	84.17								
basis	-0.96	-0.25	-0.14	-0.29	0.86	0.42	-0.02	-0.44	0.90
direction	0.02	0.44	-0.90						
palmposition	-59.54	186.87	-8.41						
palmvelocity	0.52	-0.82	1.61						
palmnormal	0.29	-0.86	-0.42						
wristposition	-68.37	166.53	37.08						
elbowposition	-154.90	95.54	260.36						
spherecenter	-52.05	282.18	-18.77						
sphereradius	102.29								
translationpro	0.31								
translation	0.01	-0.01	0.04						
scaleprobabilit	0.35								
scalefactor	1.00								
rotationproba	0.34								
rotationangles	0.00	-0.00	0.00						

Figure 2.3. leapmotion object help patch.

CHAPTER 4
MAX/MSP/JITTER

Introduction

The Gesture Controlled Performance Tool (GCPT) was realized using the visual programming language Max/MSP/Jitter, which is often simply referred to as Max. Max provides the main programming elements while MSP is the audio signal processing component and Jitter provides a real-time video, 3-D, and matrix processing programming component. Abundant help files, tutorials, and a vibrant programming community combined with the versatility and simplicity of programming with Max to yield complex and responsive audio and video manipulation and presentation made it the obvious choice to build the GCPT.

Background

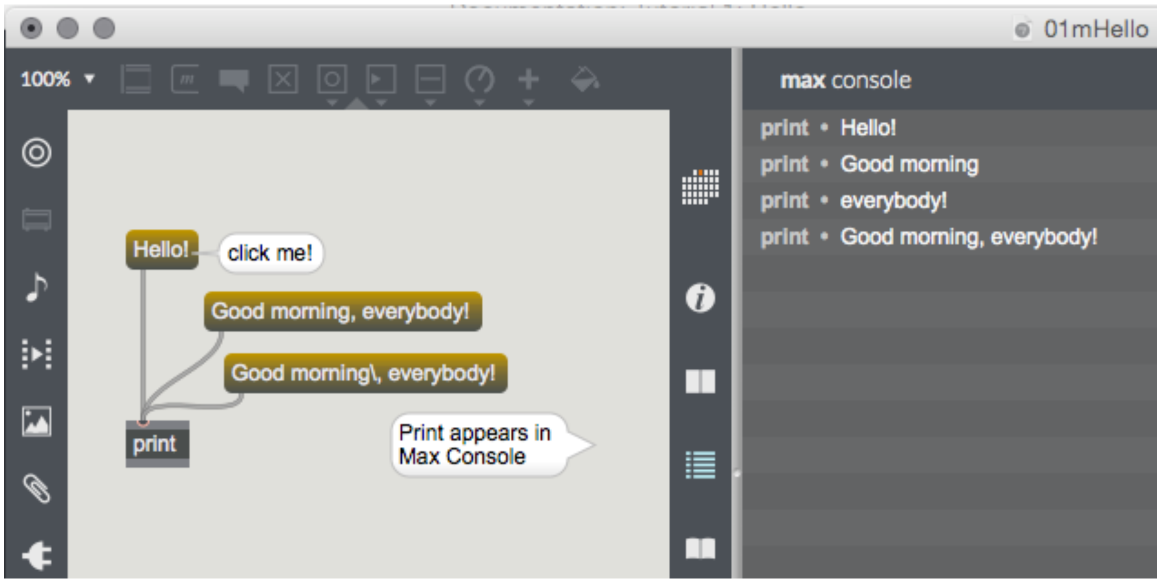
During the mid-eighties Miller Puckette created Max at IRCAM (developers of the leapmotion external used in the GCPT) as a way to compose interactive computer music (“Max (Software)”, n.d.). Max has since been revised and maintained by the Cycling ’74 software company into the polished software that is available today. Max currently has widespread use not only as a research and composition tool for electroacoustic musicians but for mainstream users of the Abelton Live digital audio workstation software. Ableton Live users can use Max For Live plugins, a collection of MIDI and audio effects, and virtual instruments created in Max.

Max Programming

Former University of California at Santa Cruz professor Peter Elsea, in his Max tutorial *Max and Programing,* confronts the question of whether Max is a real programming language. He distills his answer in this way;

“Max is a program that interprets actions of compiled objects that are coded in a procedural language. Since Max uses icons to represent chunks of high level language, Max is a meta-language. You can program Max in two ways. You can connect the existing objects, and you can write and compile your own objects in a high level language called C”(Elsea, 2007).

Max programming is modular and takes place in a programming window called a patcher. Within the patcher window the user connects objects, which are algorithms that precipitate actions (Winkler, 1998), with patch cords. Patch cords are graphically drawn lines that connect one object to another. Patch cords carry messages, actions and arguments that provide the data an object requires to execute its particular function. The end result is called a patch.



```
print • Hello!
print • Good morning
print • everybody!
print • Good morning, everybody!
```

Figure 4.1. The most common introduction to Max programming is the coding of a “Hello World” patch. It illustrates the basic concepts required to build a successful patch.

CHAPTER 5
OVERVIEW OF THE GESTURE CONTROLLED PERFORMANCE TOOL (GCPT)

Introduction

Working within the Max environment allows the user to customize their workspace and streamline tools, effects, parameters, input, and output down to the minutest detail. Custom patches can be built to achieve one goal or myriad goals. For performers looking to have flexible and precise control over their media, the Max environment is a powerful tool.

As case in point, in early 2015, the author created a customized mixer for a performance piece, which consisted of four stereo inputs directed into eight main channels. From the main channels the audio could be routed to any one of eight auxiliary channels to receive an effect or specialized treatment. From the auxiliary channels, the audio would be sent to eight return channels for final treatments of amplitude or panning and directed to the output of choice. There was also the option to send the input directly to return channels as clean, unprocessed audio. The performance was controlled and conducted with the Mira application on an iPad2 tablet. Mira is Max's proprietary interface for iPad, which can represent many of Max's data input and visual control objects. For this project, the mixer patch was repurposed and refined for use with the Leap Motion Controller. Below is a break down of all of the components, patches) that comprise the GCPT.

GCPT Root Patch (Hidden)

The GCPT was created with the menu tab feature to easily access important patches embedded within the root patch. This makes programming of a performance more efficient as each feature can be found on the menu bar rather using the common Max method locating a patcher and double clicking on the object while the patch is locked (unable to edit). The root patch contains every patcher attributed to the GCPT. The author arranged all of the major encapsulated patchers here to be displayed in the tab menu. The order of the tabs can be organized by arranging them in the order one would like them to appear on the tab menu.

In general the root patch is not displayed in the tab menu at the top of the patch, but can be accessed by pressing the control key and clicking to the right of all the tabs on the menu bar. To return to performance mode the previous steps are repeated.

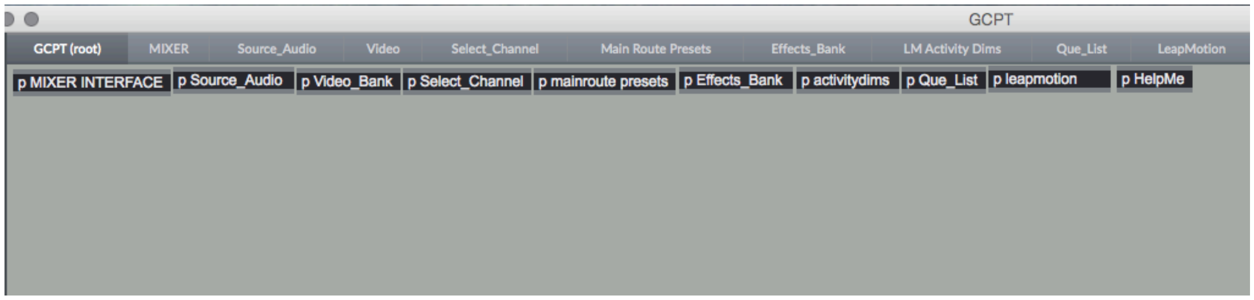
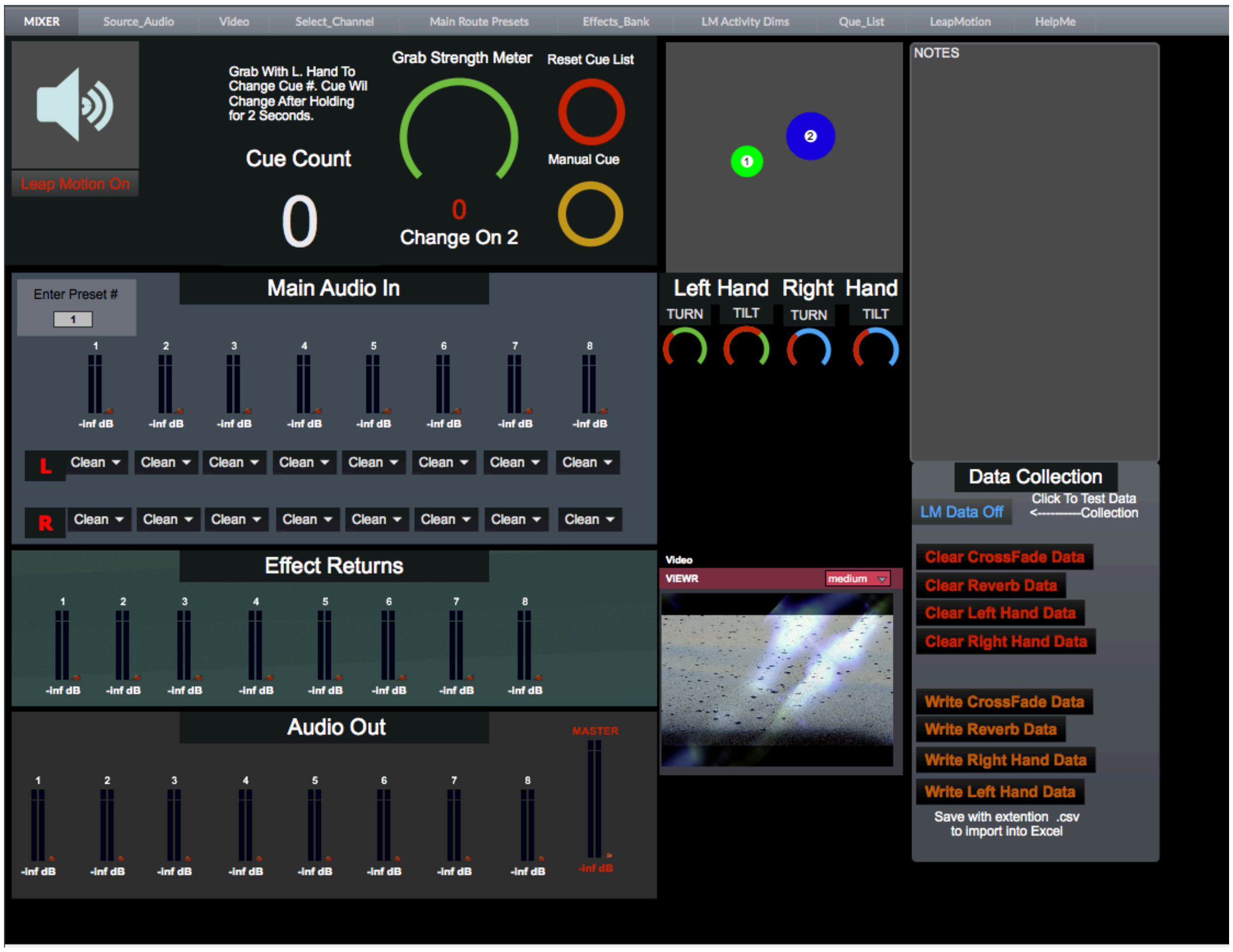


Figure 5.1. Root patch.

The Mixer Interface and Performance Control Patch

This patch is the main control and performance monitoring area of the GCPT.



Grab With L. Hand To Change Cue #. Cue Wil Change After Holding for 2 Seconds.

Save with extention .csv to import into Excel

Figure 5.2. Mixer interface and performance control patch.

Audio On/Off

The ezdac object (Figure 5.3) controls this feature. This object, an icon of an audio speaker, is controlled by a mouse click that will turn audio on or off.

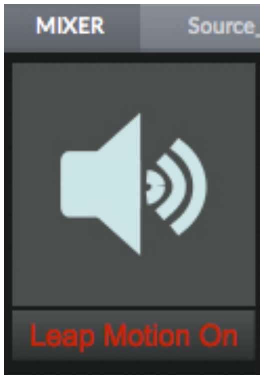


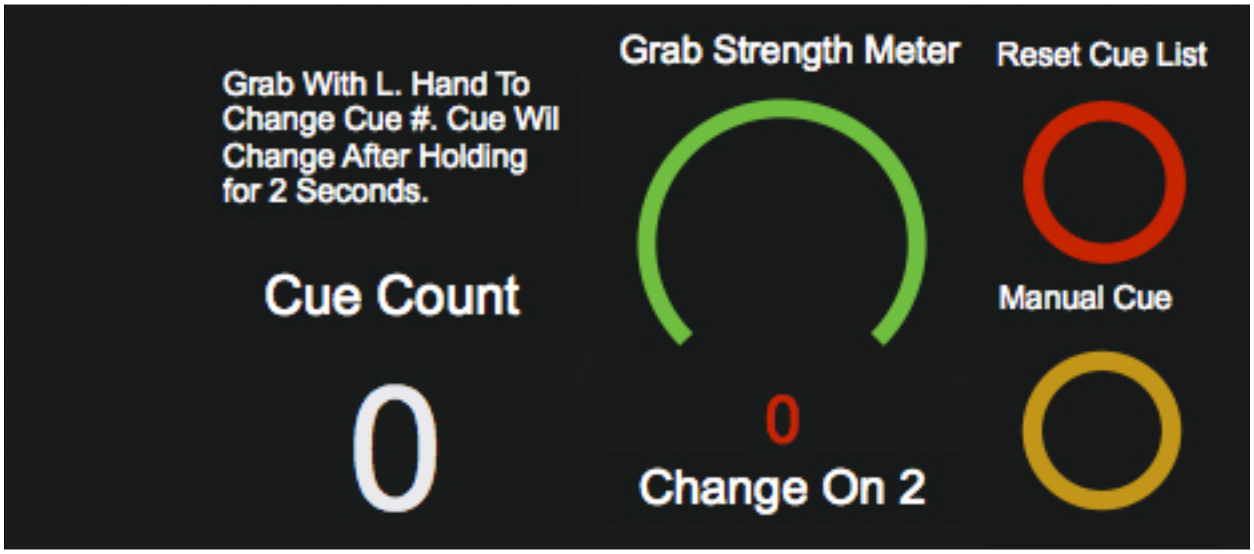
Figure 5.3. ezdac and Leap Motion toggle.

Leap Motion On/Off

To start receiving data from the Leap Motion a textbutton object (Figure 5.3) is enlisted to act as a toggle to send an on or off message to the leapmotion object.

Grab Strength Meter

This important feature (Figure 5.4) allows the grab strength of the left hand to be monitored. This gesture is what cues events inside the GCPT. This feature’s development and programming will be discussed with the overview of the Qlist patch. Below the meter is a timer that gives the hold count of a full strength grab value. The grab gesture must be held for two seconds in order for an event to be cued.


Grab With L. Hand To Change Cue #. Cue Wil Change After Holding for 2 Seconds.
Cue Count
0
Grab Strength Meter
0
Change On 2
Reset Cue List
Manual Cue
Figure 5.4. Grab Strength Meter, Cue Count Tracker, and Manual Reset and Cue Buttons.

Cue Count Tracker

This feature keeps track of cued events and lets the performer know of their place in the performance. The cue count tracker is a number box that receives the status of the counter object located in the QList patch. The counter object will be discussed with the overview of the Qlist patch.

Manual Reset and Cue Buttons

These buttons are essential for efficient programing of a performance and for overriding the cues being sent by the Leap Motion. A bang from the Reset button is sent to a receive object in the qlist patch that gives messages to the qlist object to rewind and reset the cues. This can be thought of as performance rewind button. The manual cue button steps through cues of a performance without requiring the user have the leap motion on or to have to enter the qlist patcher directly to change events. This is important for the programming of audio cues as one can observe the status of the gain levels as events are changed.

Leap Motion Hand Tracking Window

The nodes object displays a two dimensional space with two color-coded and numbered circles, or nodes. This object acts as a visualizer for hand positioning inside the Leap Motion area of interactivity. The nodes respond to hand gestures and are tracked along the x and y-axis as one would expect. Left to right along the x-axis and up and down along the y-axis. To display changes along the z-axis the nodes grow larger when the performer moves their hands close themselves and the nodes shrink as the hands move away from the body. To aide the performer the node associated with the left hand will turn from green to red to indicate that a cue event can be triggered with a two second full grab strength value.

Below this area is a row of four dial objects (Figure 5.5). The row is divided into two pairs representing the tilt and turn of the left and right hands. These are here to monitor the status of each value.

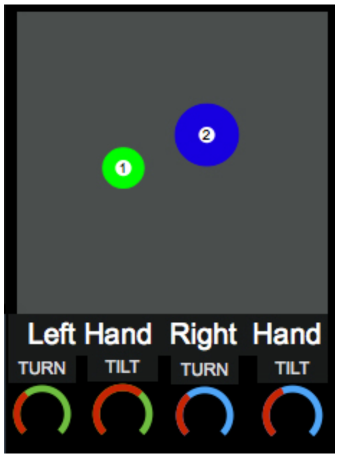


Figure 5.5. Leap Motion hand tracking window.

Mixer

The Mixer (Figure 5.6) has three banks of faders each containing eight stereo channels. The left and right channels of each fader bank can be independently routed if desired. The Main Audio In bank accepts audio from the audio source patch and can be directed to either the Effect Bank for manipulation or passed through to the Audio Out bank without any effects added to the signal. The Main Audio In channels can be configured with presets and be manually cued from a number box or from a programmed cue from the qlist. The presets are set in separate patcher.

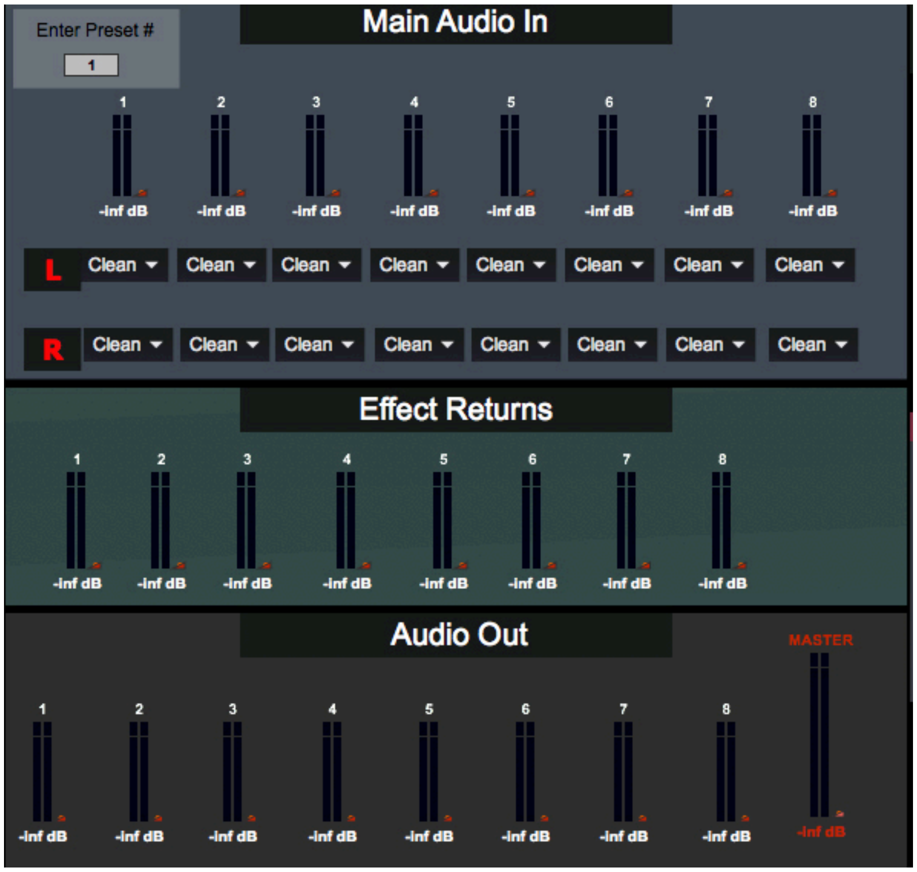


Figure 5.6. Mixer.

Data Collection and Video Preview Window

The video stream that comes from the video patch has a preview window located in this patch to provide feedback from effects and other events while being able to monitor the cue progress.

There is also a section for collecting leap motion data. There are two buttons for each parameter that will have its data collected. One acts to clear the data and the other is used to write the data to a csv (comma separated value) file which can be imported into Excel). During a performance this is initiated with cues, however if the performer wishes they can start data collection manually with the provided button.

Importing Leap Motion Data To Excel

It is useful to analyze Leap Motion data collected from the GCPT during a performance or testing phase. Data streams consisting of x, y, and z positions, hand tilt, hand turn values, and grab strength values from both the left and right hand are captured and sent to coll objects. Data can also be collected directly from any parameter that is being controlled by the particular gesture.

If a user would like save the data collected after a performance or testing phase the user simply has to click on the appropriate “Write Data” button, name the file, and add the .csv extension. Once this procedure is complete the file must be imported into Excel. Importing instead of opening allows the values to be separated into their own distinctive columns. From a new workbook page select import and choose .csv file. The page of the import menu asks for the type of file to be imported. Choose delimited and then next. The user will then choose semicolon and comma as the delimiters to correctly spirit the data streams. Clicking finish will allow the user to perform any number of data analysis strategies.

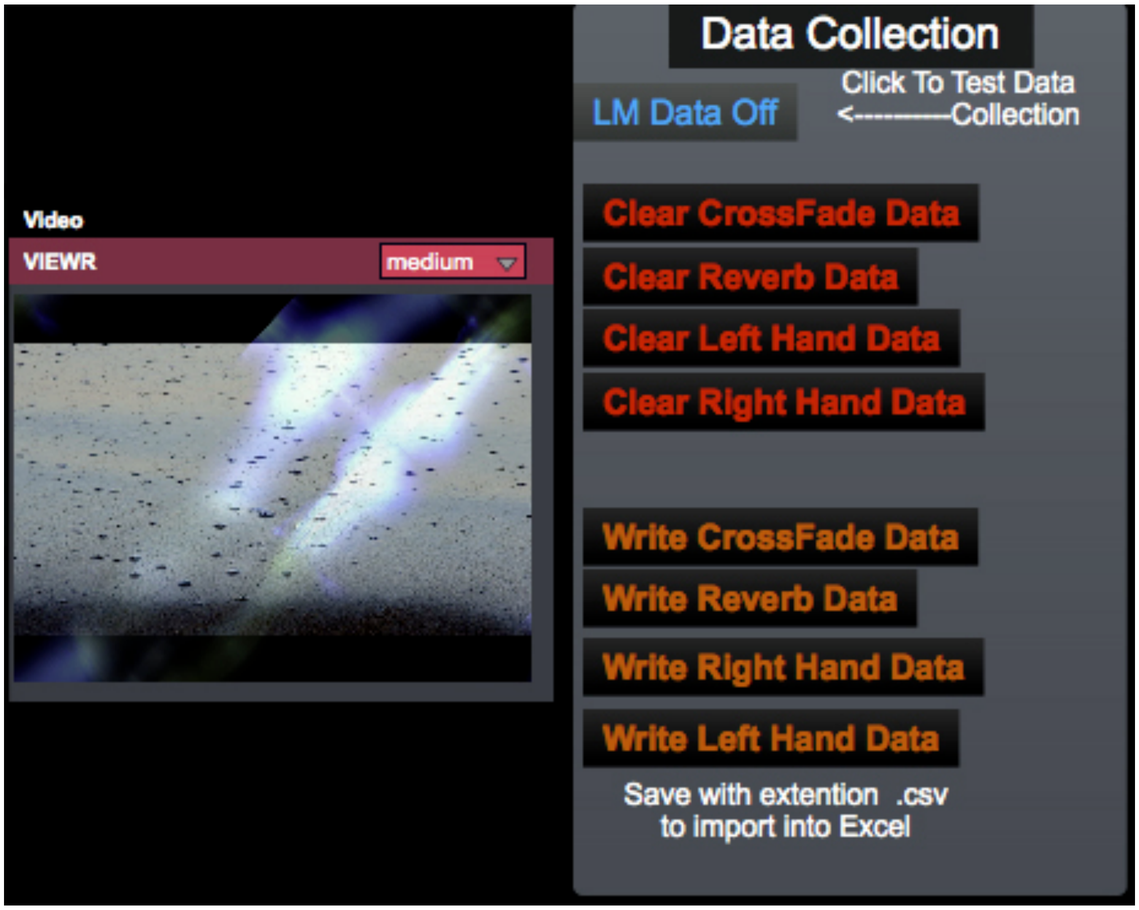


Figure 5.7. Video preview window and data collection action buttons.

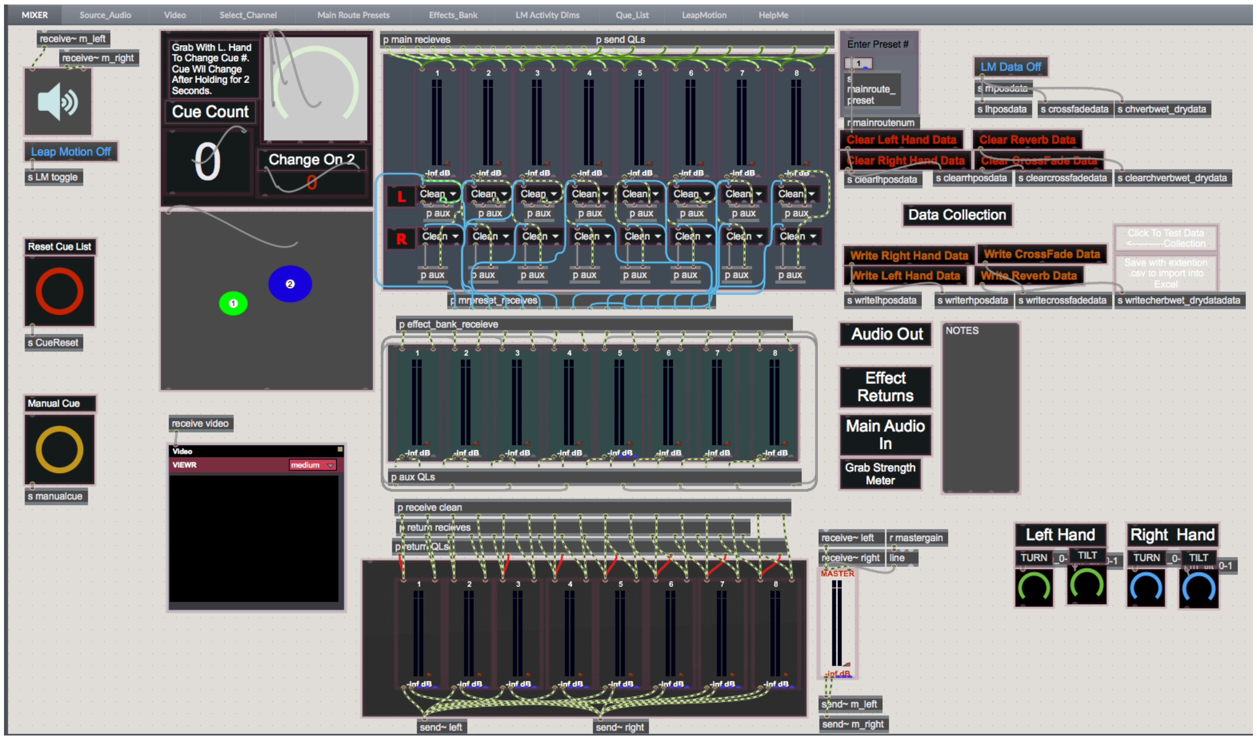


Figure 5.8. Mixer/ performance interface programming.

Audio Source Patch

This patch consists of 8 channels of audio sends. The patch in Figure in 5.9 contains eight separate playlists, though the playlists could be replaced with external audio inputs from a microphone or receives audio from any number of digital audio workstations, virtual instruments, or midi instruments. The clip selection can be controlled with a preconfigured preset and cued during performance.

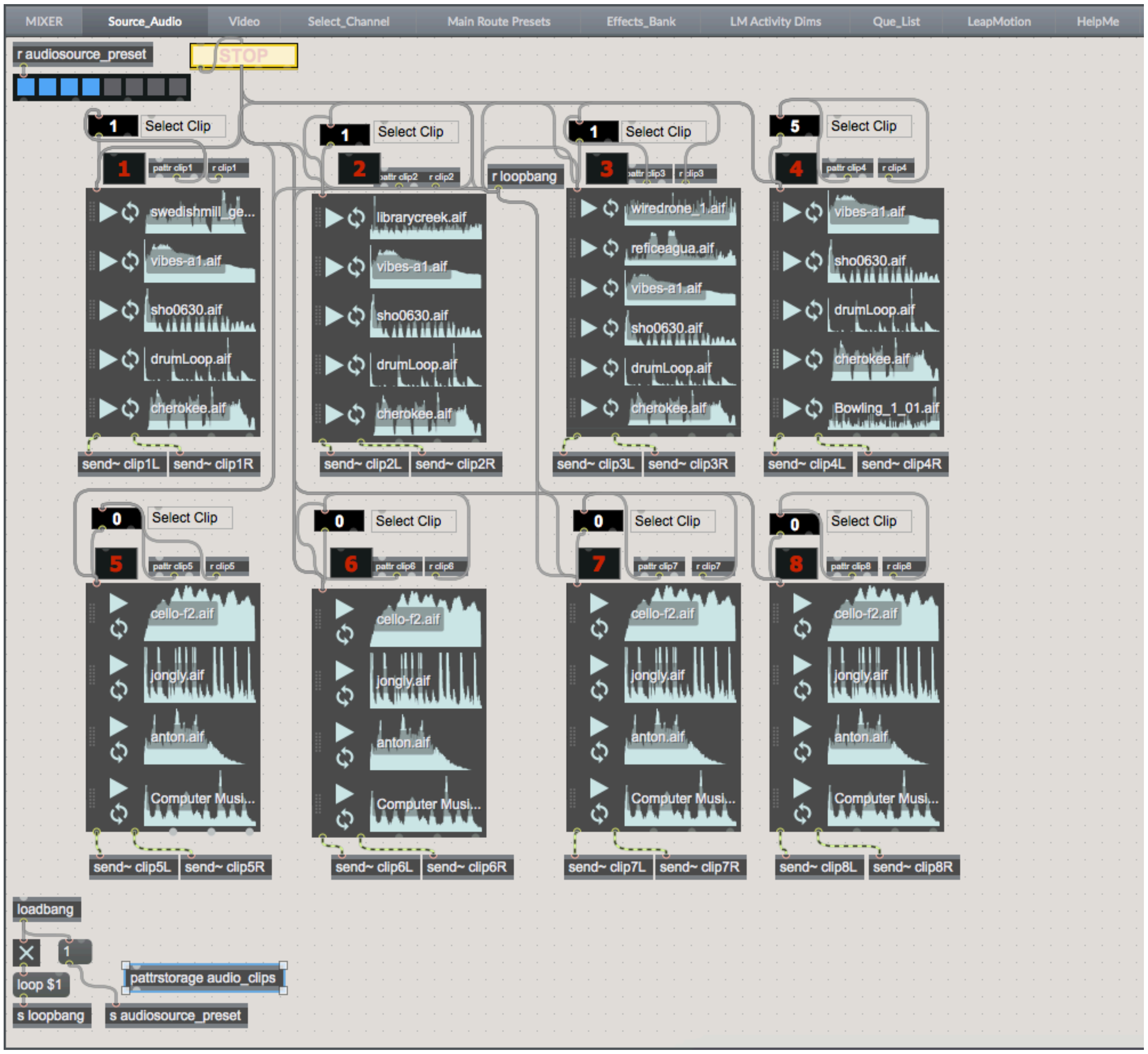
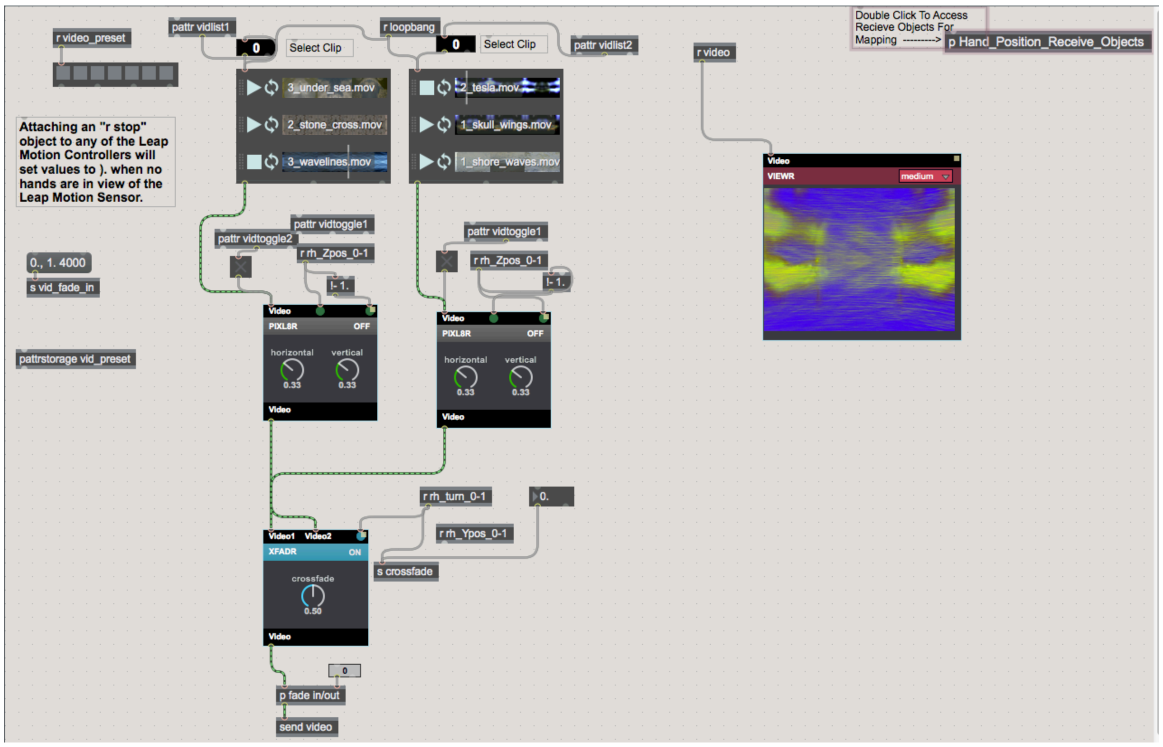


Figure 5.9. Audio source patch.

Video Patch

The video patch is made up of vizzie bpatchers, Vizzie bpatchers are essentially preconfigured jitter patches (the component of Max that is concentrated on the creation and manipulation of visual elements) that can be easily accessed from a vizzie menu. The video patch is modular and intended to be easily configured and changed without complex programming. Here there is a sub patcher that contain all of the Leap Motion receive objects that can be applied to any of the vizzie bpatchers. All input values for vizzie are 0. to 1.0.

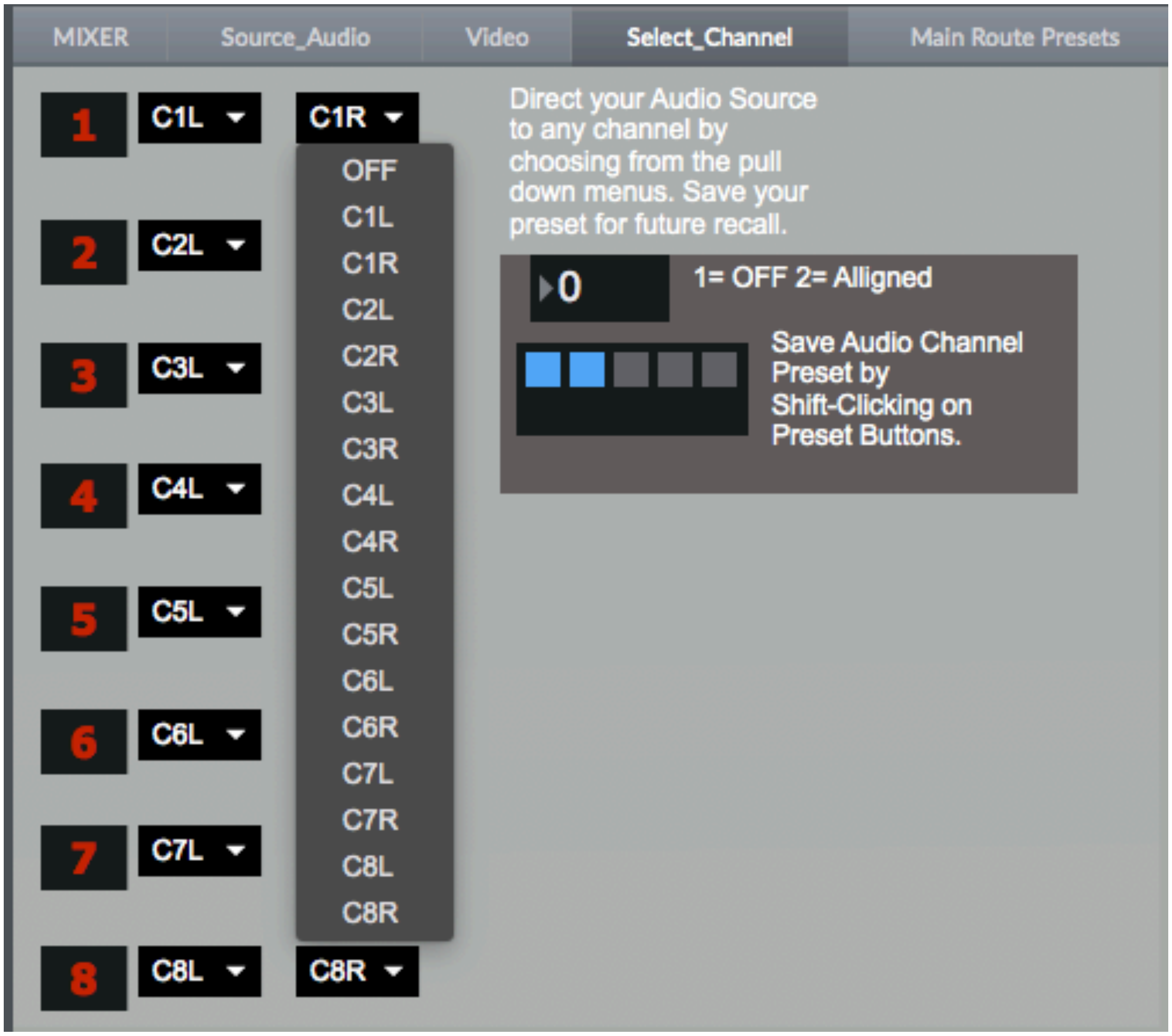

Attaching an "r stop" object to any of the Leap Motion Controllers will set values to). when no hands are in view of the Leap Motion Sensor.

Double Click To Access Recieve Objects For Mapping -------->

Figure 5.10. Video patch.

Select Channel Patch

This patch is home to where input audio can be routed. Each channel has a left and right output and can be directed toward any of the eight main send channels. There is also an off option, which keeps audio from entering the mixer. These are accessed from a pull down menu where selection can be made. These can be preconfigured with the preset object and cued from the qlist.



Direct your Audio Source to any channel by choosing from the pull down menus. Save your preset for future recall.

Save Audio Channel Preset by Shift-Clicking on Preset Buttons.

Figure 5.11. Select channel patch.

Main Send Presets Patch

The main send preset patch is nearly identical to the Select Channel tab. The main difference is that these selections direct audio to any of the effect bank channels or to the main audio out channels. These can also be preconfigured with the preset object and cued by the qlist.

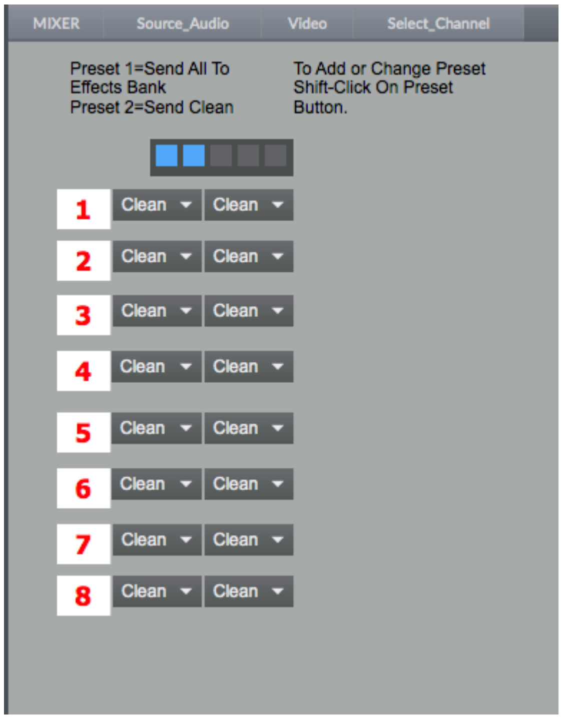

MIXER
Source_Audio
Video
Select_Channel
Preset 1=Send All To Effects Bank
Preset 2=Send Clean
To Add or Change Preset Shift-Click On Preset Button.
1
2
3
4
5
6
7
8
Clean

Figure 5.12. Main send patch.

Effect Patch

This patcher is composed of sixteen inlets and outlets where audio can be passed into and connected to an audio effect. This is similar to the video patch and is intended to be modular in nature. Any max object, Max4Live patch, or abstraction that affects audio can be placed in between the inlet and outlets and used to effect audio. Effecting the parameters of each audio effect with Leap Data may require additional scale objects or objects like the dial or slider object to scale the data appropriately. Like the Video patch this patch contains a sub patch with all the Leap Motion receive objects for ease of access.

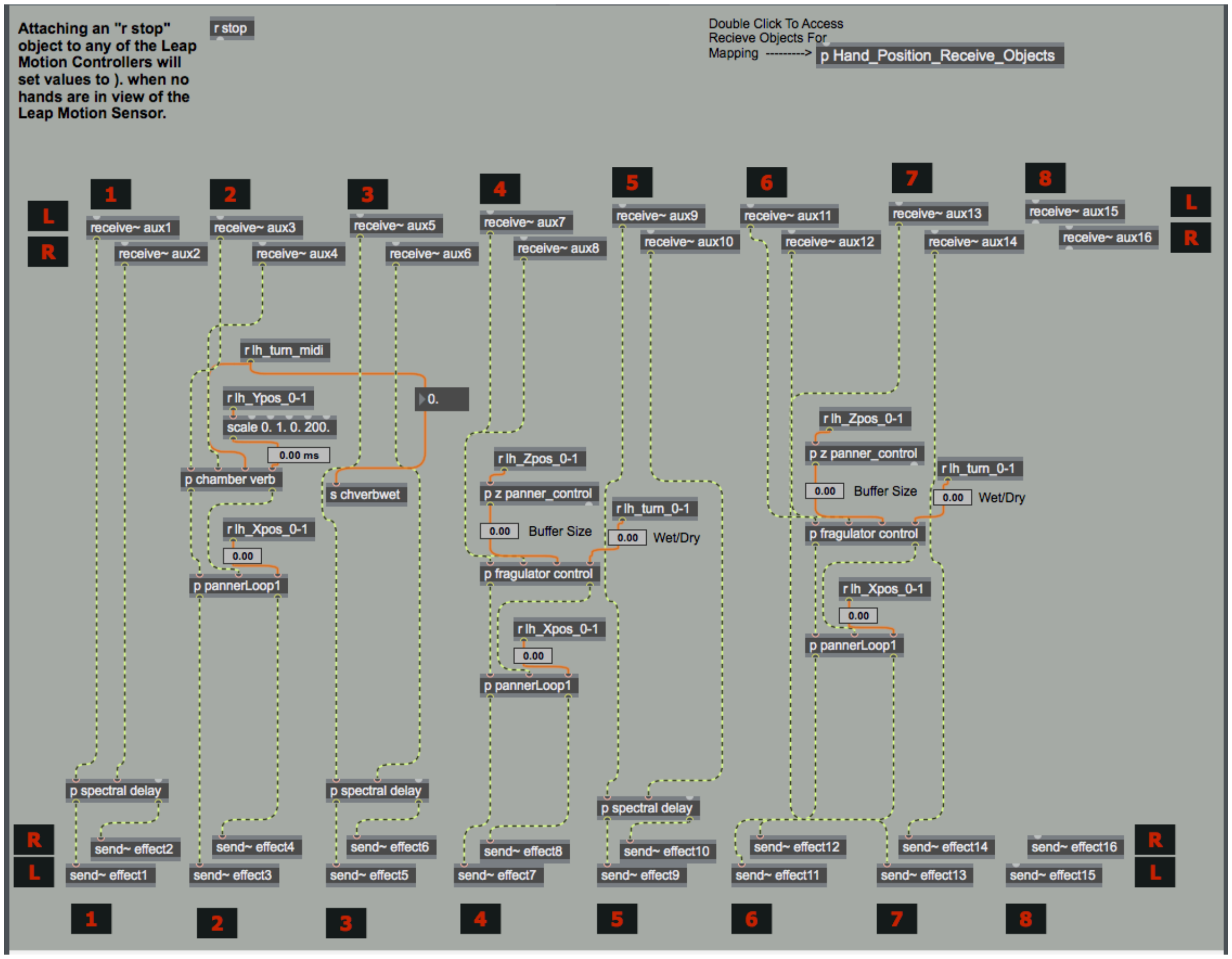


Figure 5.13. Effect Bank patch.

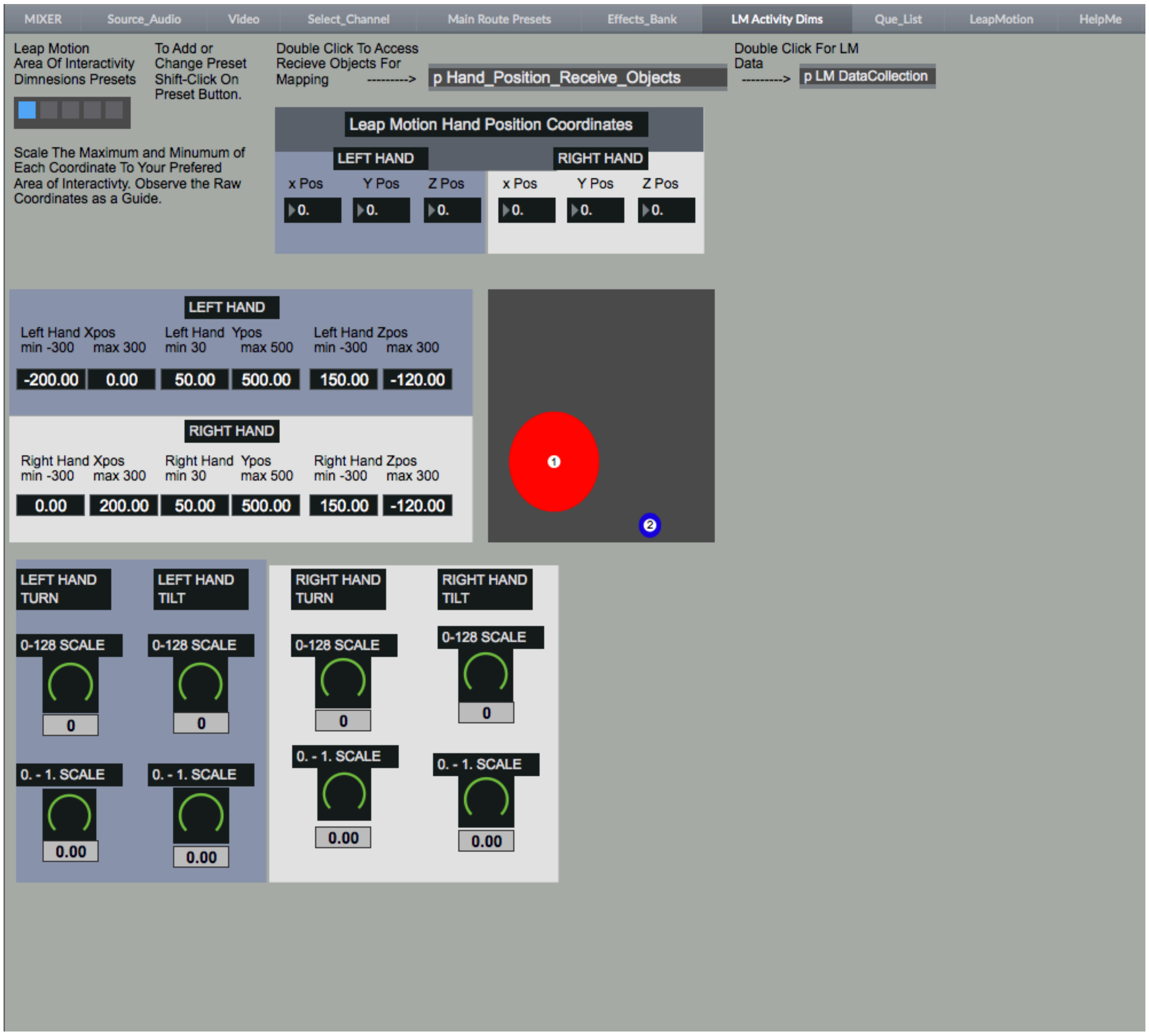
Leap Motion Activity Area Dimensions Patch

This patch accepts and scales all of the routed parameters coming in from the leapmotion object (Figure 5.14). The incoming x, y, and z hand position data of both the left and right hand is viewed in float number object. The user can observe the data and determine how to shape the area of interactivity. The users can then scale that data to an area that they see fit for their application. This is done by entering values at the number boxes at each data point for each coordinate. Presets can be set for the area of interactivity, tested and adjust as necessary. The customized values for the interactivity area then are fed to two separate scaling objects for each coordinate. One scaling object with scale the Leap motion coordinate output to values between 0. and 1.0. The scaling object scales values to between 0 and 128.

In a sub patcher scaled coordinate data is sent to the nodes object (Figure 5.15). The nodes object allows the user to visualize their hand position within the area of interactivity. This an important feature that gives visual feedback to understand how their hands are being perceived by the Leap Motion and also to what degree those positions will have on effected parameters.

This patch also contains a sub patcher where routed data can be stored for analysis. The main object in this sub patcher is the coll object. When cued this coll object accepts data and creates a list of all the incoming data. There are two coll objects that capture all of the hand position data as well as the grab strength, hand tilt, and hand turn values. These are important to analyze the users reactions to various elements of the piece and can help to root out instances of false triggers for troubleshooting. This collected data can also be used to test and observe a user's reaction to various elements of a performance. Separate coll objects can be attached to any output of an effected parameter to compare the user's performance to other parameters or even separate performances. This can be used to identify a performer's signature style of an effect for instance and be compared to other user's use of that parameter of the same piece.

This patch also contains the sub patch that lists of all of the receive objects that are related to the Leap Motion (Figure 5.17).



Leap Motion Area Of Interactivity Dimnensions Presets

To Add or Change Preset Shift-Click On Preset Button.

Double Click To Access Recieve Objects For Mapping ---------> p Hand_Position_Receive_Objects

Double Click For LM Data ---------> p LM DataCollection

Scale The Maximum and Minumum of Each Coordinate To Your Prefered Area of Interactivty. Observe the Raw Coordinates as a Guide.

LEFT HAND

Left Hand Xpos min -300	max 300	Left Hand Ypos min 30	max 500	Left Hand Zpos min -300	max 300
-200.00	0.00	50.00	500.00	150.00	-120.00

RIGHT HAND

Right Hand Xpos min -300	max 300	Right Hand Ypos min 30	max 500	Right Hand Zpos min -300	max 300
0.00	200.00	50.00	500.00	150.00	-120.00

Figure 5.14. Leap Motion Activity Area Dimensions presentation patch.

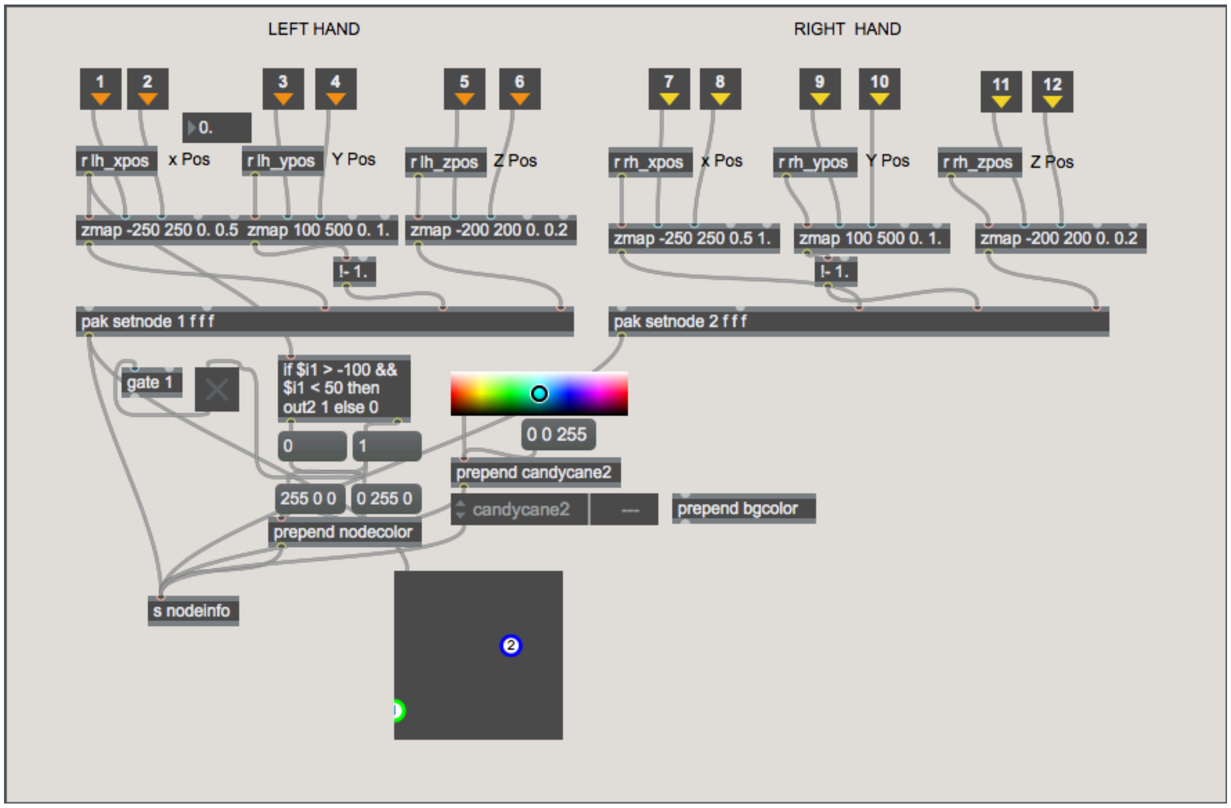


Figure 5.15. Node object programming.

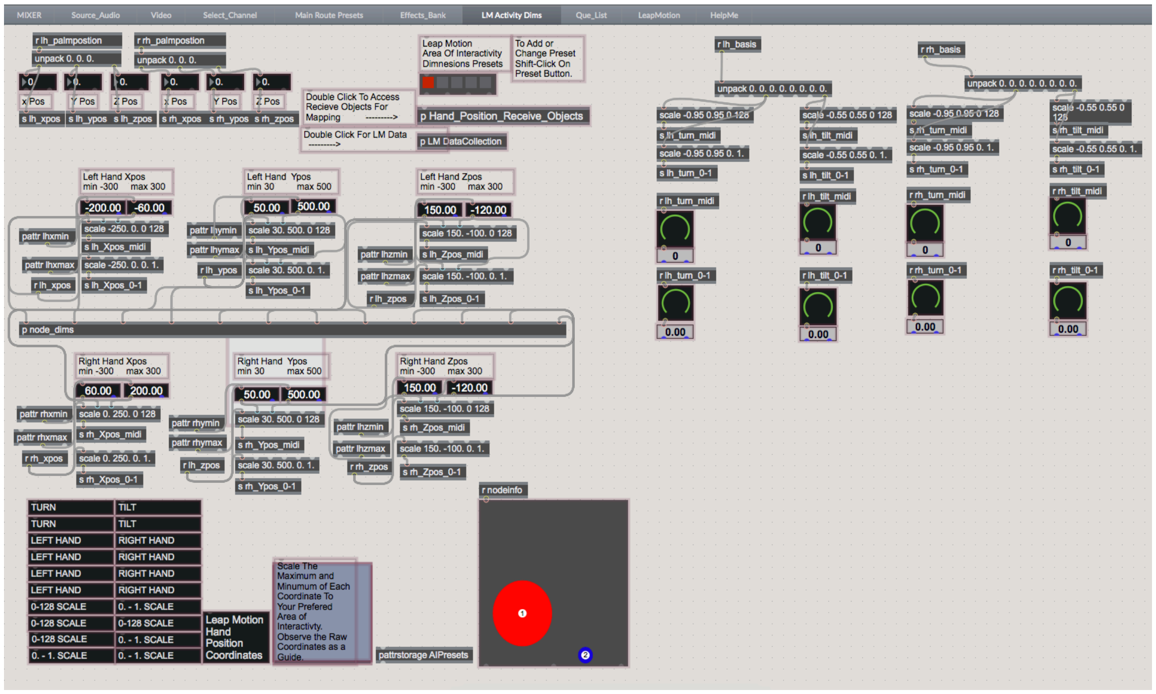

MIXER
Source_Audio
Video
Select_Channel
Main Route Presets
Effects_Bank
LM Activity Dims
Que_List
LeapMotion
HelpMe
r lh_palmpostion
r rh_palmpostion
unpack 0. 0. 0.
Leap Motion Area Of Interactivity Dimensions Presets
To Add or Change Preset Shift-Click On Preset Button.
Double Click To Access Recieve Objects For Mapping
Double Click For LM Data
p Hand_Position_Receive_Objects
p LM DataCollection
r lh_basis
r rh_basis
Left Hand Xpos min -300 max 300
Left Hand Ypos min 30 max 500
Left Hand Zpos min -300 max 300
-200.00
-60.00
50.00
500.00
150.00
-120.00
p node_dims
Right Hand Xpos min -300 max 300
Right Hand Ypos min 30 max 500
Right Hand Zpos min -300 max 300
60.00
200.00
50.00
500.00
150.00
-120.00
r nodeinfo
TURN
TILT
LEFT HAND
RIGHT HAND
0-128 SCALE
0. - 1. SCALE
Leap Motion Hand Position Coordinates
Scale The Maximum and Minumum of Each Coordinate To Your Prefered Area of Interactivty. Observe the Raw Coordinates as a Guide.
pattrstorage AIPresets

Figure 5.16. Leap Motion activity area patch programming.

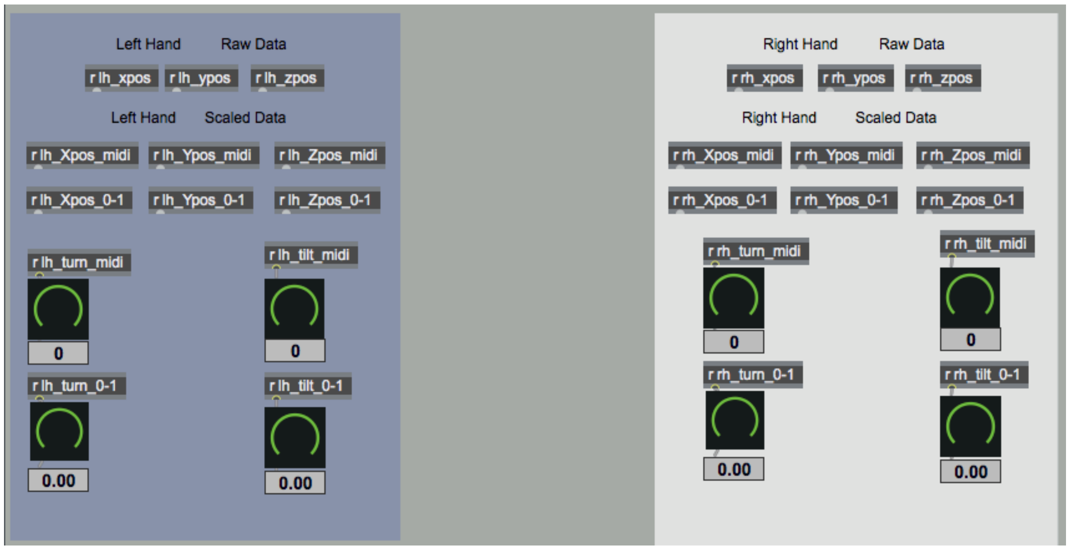
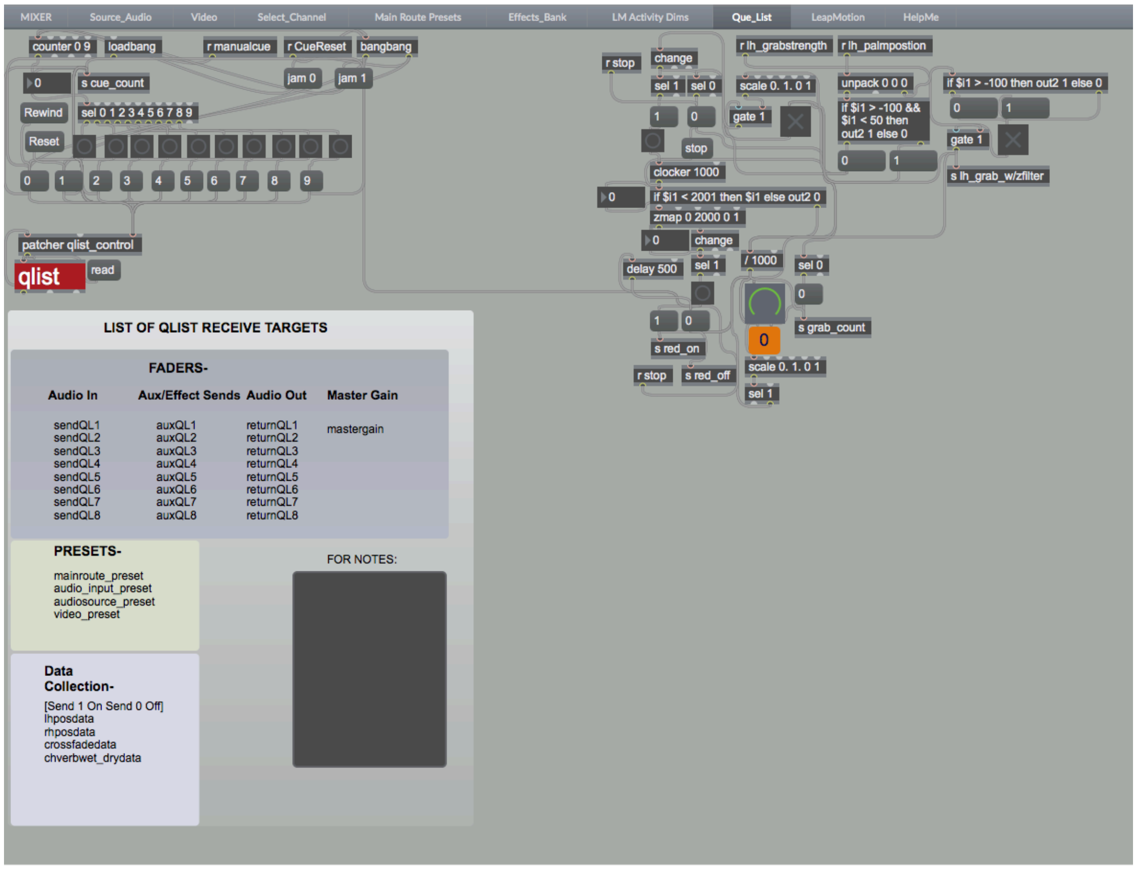


Figure 5.17. Leap Motion receive objects for scaled data.

Qlist Patch

The qlist tab is where all of the cues for the performance are programmed. The present patch contains nine cues, which are trigger by the grab strength data coming from the Leap Motion. The programming for the grab strength to cue the event in the qlist object occurs within this patch. There are also send objects that are referenced within the Mixer patch that displays the cue progress and the grab strength meter. Sends from the Mixer patch are received here to manually reset and cue the the events generated from the qlist object.

There is a list of all of the receive targets that can be cued for the qlist. These are necessary to properly assign the values to each parameter.



LIST OF QLIST RECEIVE TARGETS

FADERS-

Audio In	Aux/Effect Sends	Audio Out	Master Gain
sendQL1	auxQL1	returnQL1	mastergain
sendQL2	auxQL2	returnQL2	
sendQL3	auxQL3	returnQL3	
sendQL4	auxQL4	returnQL4	
sendQL5	auxQL5	returnQL5	
sendQL6	auxQL6	returnQL6	
sendQL7	auxQL7	returnQL7	
sendQL8	auxQL8	returnQL8	

PRESETS-

mainroute_preset
audio_input_preset
audiosource_preset
video_preset

FOR NOTES:

Data Collection-

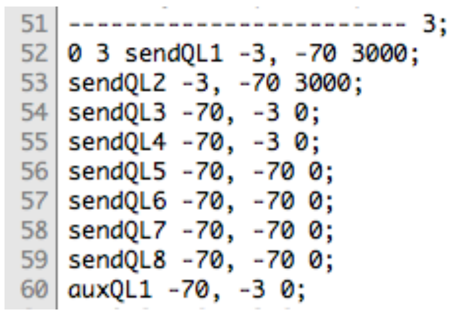
[Send 1 On Send 0 Off]
lhposdata
rhposdata
crossfadedata
chverbwet_drydata

Figure 5.18. Qlist patch.

qlist Object

For the purposes of performance and composition, it is advantageous to use the qlist object to send trigger messages to receive objects, which instigate a predetermined set of instructions. The qlist object is an object that stores a collection of cues that can be triggered remotely from anywhere in the patch. To remotely trigger an event or cue, one has to enter the address of the object that will receive message. The message must be formatted to the proper string of arguments to be interpreted by the specific object. One of the main cues for this project is a control for amplitude for each of the gain objects. To adjust amplitude smoothly, a line object is typically employed to direct ramp arguments, a set of beginning and ending variables over a specified duration.

Figure 5.19 shows all of the messages for the first cue. Line 51 consists of a series of dashed lines with the event number. The next line indicates that these are the first string of messages. Each event begins this way to visually separate each event in the queue. Line 52 affirms that this is event 3. The next piece of information in line 52 is the address of the receive object “sendQL1”. This is followed by the contents of the message for the line object. -3 is the beginning state, -70 is the ending state and 3000 is the duration. When this message is triggered by the appropriate hand gesture, a message is sent to the receive object “sendQL1”. This is accepted by the line object, which is then directed to the gain object. The gain slider moves from -3 dB to -70 dB over the course of 3000 milliseconds thereby fading out the audio associated with this channel.



```
------------------------ 3;
0 3 sendQL1 -3, -70 3000;
sendQL2 -3, -70 3000;
sendQL3 -70, -3 0;
sendQL4 -70, -3 0;
sendQL5 -70, -70 0;
sendQL6 -70, -70 0;
sendQL7 -70, -70 0;
sendQL8 -70, -70 0;
auxQL1 -70, -3 0;
```

Figure 5.19. Qlist messages.

Using the Leap Motion's Grab Strength Value to Cue Events

One of the primary uses for the Leap Motion in conducting a live performance is to assign certain gestures to trigger event changes. Events could be as simple as fading from one sample or instrument to another. Cueing a dramatic use of an effect or simply beginning or concluding the performance.

For this project the author decided upon the grab strength value to indicate when to trigger major changes during the performance. The gesture of forming a fist is very exact. Although the Leap Motion in finely attuned to tracking the fine motor gestures of all ten fingers the author wanted a solid and definite gesture to indicate an event change. When testing for tracking ability and latency issues the author found that there were moments where locating all fingers of both hands could prove to be problem. This eliminated the use of holding up a certain amount of fingers or pointing one finger in a particular zone of the interactivity area to cue events. Also during a performance a performer could potentially get lost in the moment and create a gesture that could be misidentified by the Leap Motion and an unintended cue could be triggered. The forming of a fist gesture takes a certain conscious effort and is more defined than the waving of fingers in view of the Leap Motion's sensors.

Grab strength is essentially a floating-point value between 0. and 1.0. The value 1.0 being a closed hand and the 0. value an open hand. In order to make use of this data steam a number of scaling paradigms and conditions had to be assigned to control the flow of data to initiate the trigger message as well as to provide a visual representation of the grab strength action.

The grab strength data stream as previously noted is conveniently represented by floating point values between 0. and 1.0. However, if the data is attempted to be used in its raw form and connected to either a bang object or any other object will result in the connected object receiving a multitude of input values. The reason for this is that the leapmotion object is being polled by a metro (short for metronome) object that issues a bang message every 10 milliseconds to scan for the latest tracking data received from the Leap Motion software. In order to control the data stream a scale object and change object were used. The scale object does what one might expect, scale a range of incoming values to that of the desired range. For instance a common scaling strategy in Max takes the range of 0 to 127 and scales it down to values between 0. and 1.0 or vis versa. As 0 and 1 messages are often used in Max to turn functions on and off scaling the Leap Motion's grab strength floating point values to integer values was necessary. The change object's purpose is to output a value only if the incoming value changes. Therefore when the grab strength value reaches 1.0 the change object will detect the change and allow the 1 value to be passed through it. The 1 can be directed to a select object (or sel object) with an argument of 1 that will output a bang message one time.

In preliminary tests this worked admirably to cue the events of the performance. Each time the grab strength value reached 1.0 a bang was sent to a counter object which stepped through a predetermined string of numbers; 1-9 for example. However, three issues arose during testing. The first being that at times the Leap Motion’s sensitivity often resulted in multiple triggered cues as a value could shift between 1.0 and .98 and back to 1.0 very quickly. Since the scale object output anything less than 1.0 as a 0 the change object could receive an unintended 1 message and trigger the counter object. The second issue, which occurred, was accidentally triggering the event while attempting to utilize other sets of data streaming from the Leap Motion with other hand gestures. Thirdly, when the open left hand was placed in front of and to the side of the area of interactivity projected by the Leap Motion a false grab strength indicator of 1.0 would be measured. This false grab strength measure appears to be due to a bug or possibly some other interference due to lighting conditions. This condition was also observed using other leap motion patches and OSC protocols. To counter act these false triggers the author undertook a more refined approach to controlling the data and implemented the use of visual feedback displays to confirm the Leap Motion’s tracking with the physical gesture.

The first step to eliminating the false triggers was to institute two conditional filters, which limited the area of activity in which the grab strength values could be received by the patch. The first condition excluded the area in front of and to the side of the Leap Motion where an open hand would give a full strength grab value. This was achieved by blocking the data flow when the left hand passed -100. on the z axis. Another condition opens the data stream when the left hand is directly over or slightly to the left of the Leap Motion. The opening and closing of the data stream is done with the gate object, which can open and close streams of data. When predetermined conditions are met the gate opens and when the conditions fall out of range the gate closes.

To gain a visual context for the data being received for the grab strength values the raw data was attached to a dial object that was preconfigured the to range of 0. to 1.0. As the left hand opens and closes a red line indicates the strength and increases or decreases in connection to the tracking data. The visual feedback from this grab strength meter aides the user by affirming the data tracking strength with the physical gesture.

The new strategy also required that a grab strength value of 1.0 be held for two seconds before a bang message be delivered to the counter object. To create the meter the raw Leap Motion data for grab strength is connected to a dial object that was configured to the Leap Motion’s grab strength values. As grab strength value increases the dial moves clockwise until the value reaches one. When the value 1.0 is reached a clocker object (essentially acting as a timer) is triggered and begins counting everyone 1000 milliseconds until it reaches 2 seconds. A display below the dial informs the performer of the progress of the timer. When that milestone is reached the dial blinks red to indicate completion and a bang message is sent to the counter and an event is triggered. If the grab strength value is not held for 2 seconds the clocker object resets and the process repeats. The clocker and the grab strength meter will also reset to 0 if both hands exit the area of activity of the Leap Motion.

This strategy has proven to be the most reliable and has significantly reduced the vulnerability of accidentally triggering an event. An analysis of left hand grab strength value confirms this. The line graph (Figure 5.21) details the occurrence of full strength grab values collected from the left hand during a test performance. There are ten occurrences of a 1.0 value, however only nine resulted in the cue of and event. With the exception of the starting and ending cues (where data collection started and ended abruptly) there are seven full strength grab values with perceivable hold durations reflecting the necessary hold time to cue an event.

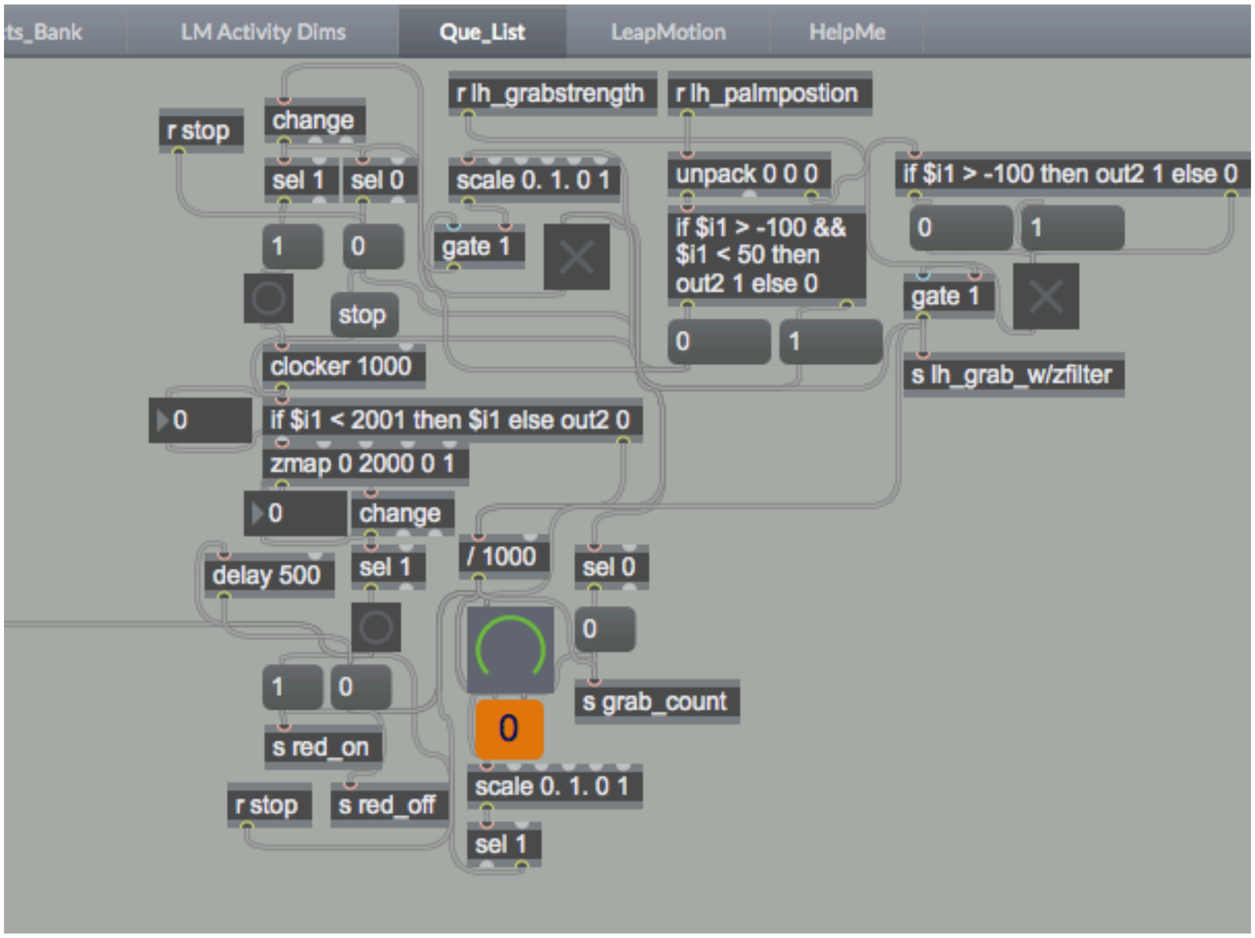


Figure 5.20. Grab strength cue programming.

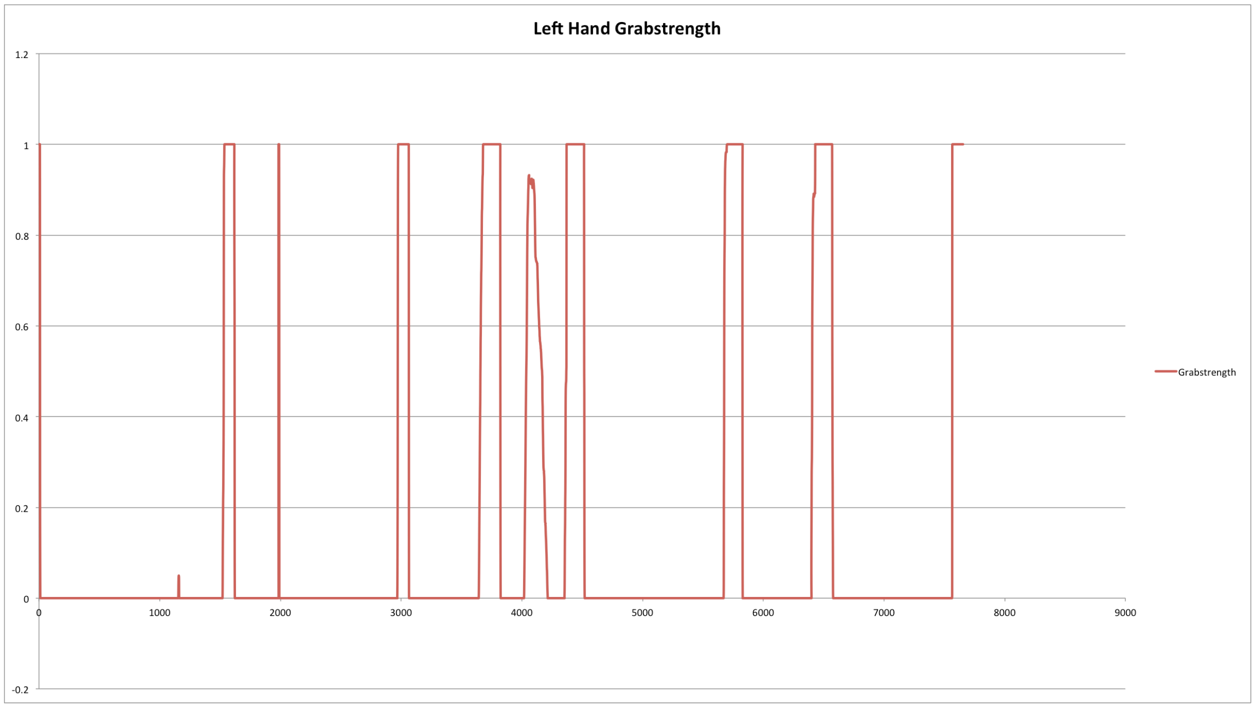


Figure 5.21. Grab strength line graph.

Leap Motion Patch

The leapmotion patch is where the leapmotion object is toggled on and off to start receiving data. A receive object is located here so that the object can be toggled on and off from the Mixer patch. The leapmotion object is the origination point of all the Leap Motion data. The hands patcher located is accessed here. From the hands patcher the desired parameters can be routed to the Leap Motion activity area dimensions patch to be scaled and sent through the patch. There is a small amount of added programming here that contains a stop message. The stop message is a value of zero that gets sent via the send and receive objects to any parameter it is connected to when both of the performer's hands leave the area of activity. This is an important feature used primarily by the left hand grab strength cueing programing to help prevent an undesired cue event. It can also be use to quickly turn off the dry/wet parameter of an effect such as reverb or delay for example.

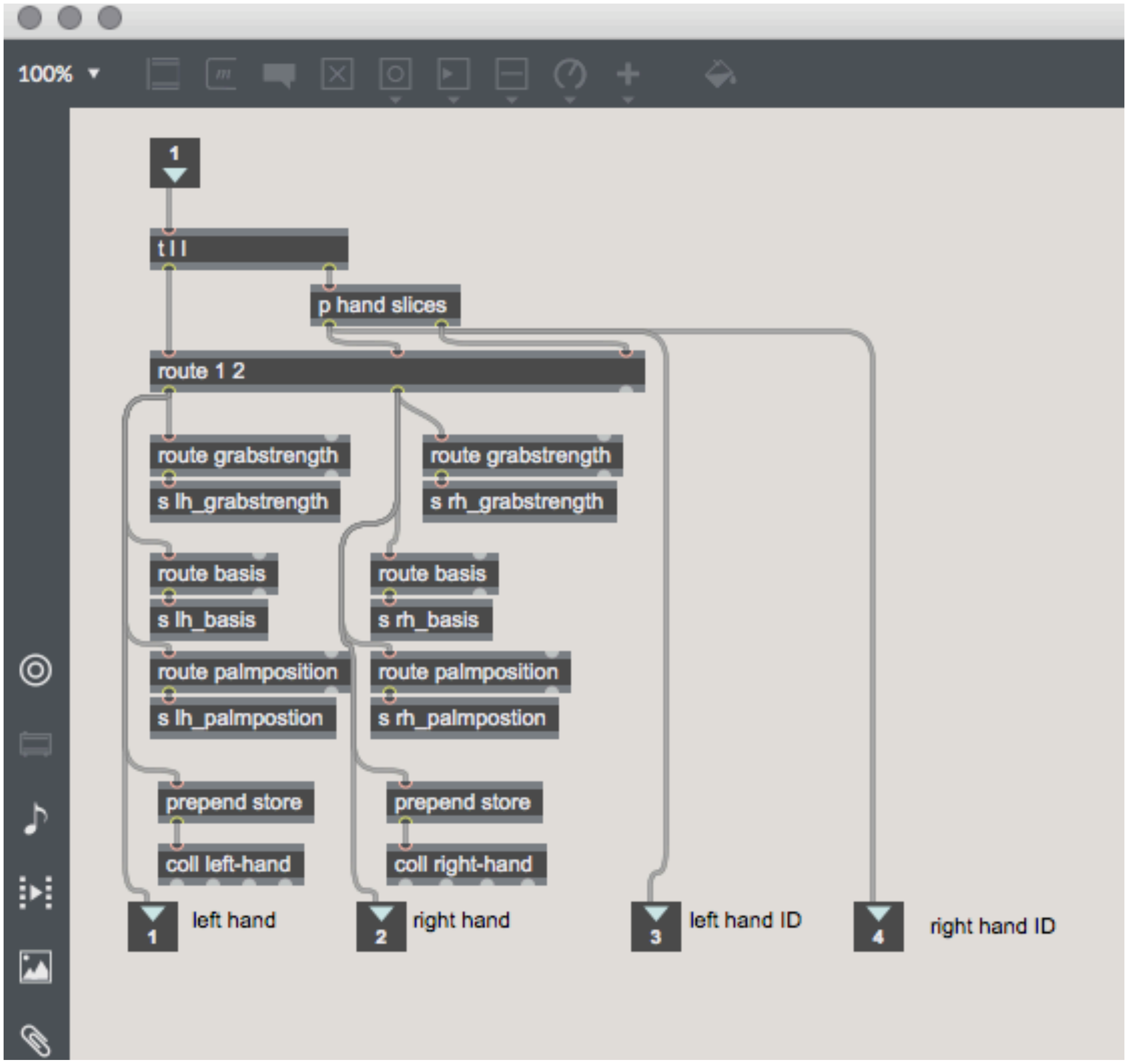


Figure 5.20. Leap Motion routing patch.

CHAPTER 6
GCPT Demonstration Performance

Hand Gesture As Performance Control

The proliferation of computer music and digital DJing has resulted in many performances where the artist is hovering over a laptop computer with other devices seemingly disengaged with not only the audience but with the music as well. Sadly, the activity of the performer and the energy of the music are often visually out of touch.

The GCPT has the ability to address this issue by incorporating an array of hand gestures into a performance in a similar way that a traditional conductor interacts with musicians. The GCPT could potentially comprise of traditional conducting gestures to control a performance though the freedom to experiment is left to the composer. The performer can then be perceived by the audience to be more physically engaged with the music through their gestural interaction with the Leap Motion and the GCPT.

In order to test the GCPT a demonstration performance piece was created. This was necessary to help connect the intentions of hand gestures with real-time audio and video manipulation. As noted in the previous section the GCPT was constructed to include many of the hand tracking data streams captured by the Leap Motion. To translate this data into an artistic expression required careful consideration and experimentation.

The GCPT was designed for cueing pre-established events and effecting audio and video. The gesture for cueing events had been determined to be the forming of a fist with the left hand. The intention of other hand gestures were left open ended to allow a composer the freedom to assign certain gestures or combinations of gestures to whichever parameter they desired.

Composing the Demonstration Piece for the GCPT

The author has an admittedly limited experience with live musical performance let alone performing with hand gestures. The acceptance of this fact guided the compositional strategy to keep effected parameters to a minimum and to assign audio manipulation duties to the left hand and control over the video stream to the right hand.

Cues

For the purposes of keeping the demonstration short only five cued events were programmed. The initial cue fades in the audio and video together. The next two cues introduce a new set of audio and video clips. The final cue slowly fades out all the audio and projections.

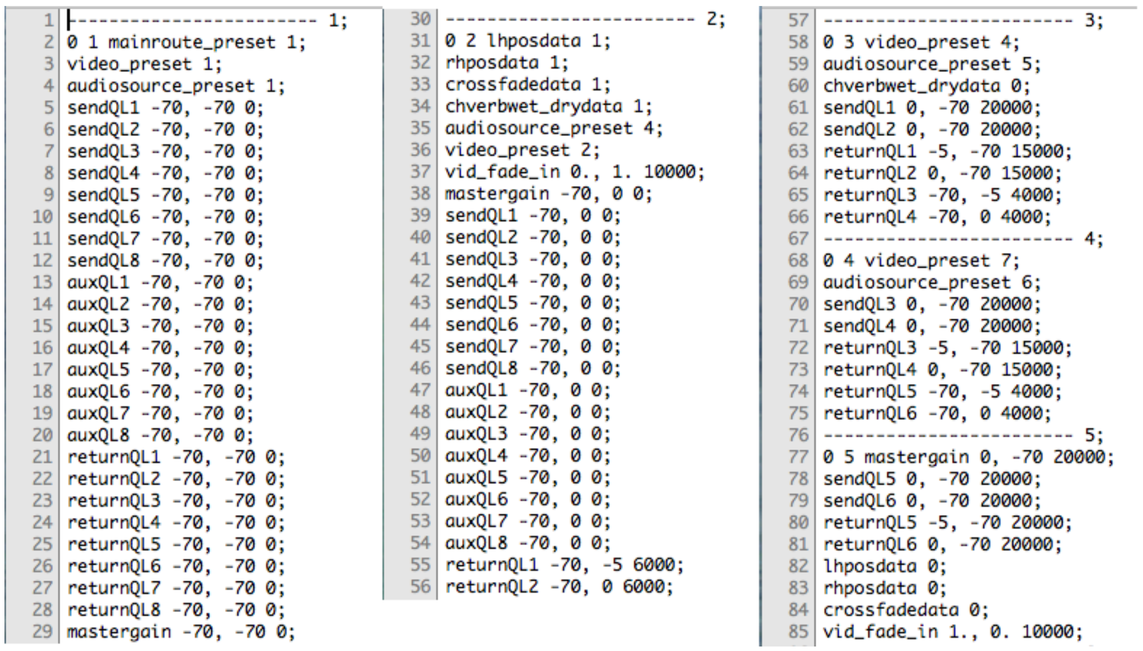
All of the cues were entered into the qlist object as described above.

The first cue selects the presets that were established for the main audio channels, videos, and audio clips. The first cue also sets all faders to -70 db.

The second cue triggers the coll objects to begin collecting data, changes the presets for audio and video, fades in the video and the selected audio channels.

The third cue again changes the audio and video presets, triggers the coll object collecting the reverb effect off, and fades out the previous audio and fades in the new. The fourth cue is activates a similar set of instructions minus the triggering of a coll object.

The fifth and final cue fades out audio and video and sends and off message to the remaining coll objects.



```
------------------------ 1;
0 1 mainroute_preset 1;
video_preset 1;
audiosource_preset 1;
sendQL1 -70, -70 0;
sendQL2 -70, -70 0;
sendQL3 -70, -70 0;
sendQL4 -70, -70 0;
sendQL5 -70, -70 0;
sendQL6 -70, -70 0;
sendQL7 -70, -70 0;
sendQL8 -70, -70 0;
auxQL1 -70, -70 0;
auxQL2 -70, -70 0;
auxQL3 -70, -70 0;
auxQL4 -70, -70 0;
auxQL5 -70, -70 0;
auxQL6 -70, -70 0;
auxQL7 -70, -70 0;
auxQL8 -70, -70 0;
returnQL1 -70, -70 0;
returnQL2 -70, -70 0;
returnQL3 -70, -70 0;
returnQL4 -70, -70 0;
returnQL5 -70, -70 0;
returnQL6 -70, -70 0;
returnQL7 -70, -70 0;
returnQL8 -70, -70 0;
mastergain -70, -70 0;
------------------------ 2;
0 2 lhposdata 1;
rhposdata 1;
crossfadedata 1;
chverbwet_drydata 1;
audiosource_preset 4;
video_preset 2;
vid_fade_in 0., 1. 10000;
mastergain -70, 0 0;
sendQL1 -70, 0 0;
sendQL2 -70, 0 0;
sendQL3 -70, 0 0;
sendQL4 -70, 0 0;
sendQL5 -70, 0 0;
sendQL6 -70, 0 0;
sendQL7 -70, 0 0;
sendQL8 -70, 0 0;
auxQL1 -70, 0 0;
auxQL2 -70, 0 0;
auxQL3 -70, 0 0;
auxQL4 -70, 0 0;
auxQL5 -70, 0 0;
auxQL6 -70, 0 0;
auxQL7 -70, 0 0;
auxQL8 -70, 0 0;
returnQL1 -70, -5 6000;
returnQL2 -70, 0 6000;
------------------------ 3;
0 3 video_preset 4;
audiosource_preset 5;
chverbwet_drydata 0;
sendQL1 0, -70 20000;
sendQL2 0, -70 20000;
returnQL1 -5, -70 15000;
returnQL2 0, -70 15000;
returnQL3 -70, -5 4000;
returnQL4 -70, 0 4000;
------------------------ 4;
0 4 video_preset 7;
audiosource_preset 6;
sendQL3 0, -70 20000;
sendQL4 0, -70 20000;
returnQL3 -5, -70 15000;
returnQL4 0, -70 15000;
returnQL5 -70, -5 4000;
returnQL6 -70, 0 4000;
------------------------ 5;
0 5 mastergain 0, -70 20000;
sendQL5 0, -70 20000;
sendQL6 0, -70 20000;
returnQL5 -5, -70 20000;
returnQL6 0, -70 20000;
lhposdata 0;
rhposdata 0;
crossfadedata 0;
vid_fade_in 1., 0. 10000;
```

Figure 6.1. Performance qlist

Audio Component

The author's primary compositional interests reside within the realm of field recording. The author recorded the audio for the demonstration at various locations using a hydrophone (underwater microphone), an x/y stereo microphone, and other recording equipment. Some of the audio clips received filter treatments prior to being imported for use by GCPT to help save on processing during the live demonstration. Selected audio clips were sent through the Effects Bank patch in the GCPT to receive treatments dictated by hand gestures. The strategy for the demonstration performance was to create a soundscape that subtlety alternates sound textures but maintain a similar tone and feel.

Audio Effects and Leap Motion Parameters

All three audio events in the demonstration shared two common features. Each event was composed of two audio clips, and one clip from each event was sent through a spectral delay effect that was not controlled by the Leap Motion. Spectral delay is a filtering effect that delays the various frequencies of an audio signal by different amounts (Abel, Pekonen, Smith, Vesa, 2009). The second clip of each event was sent through an effect that was controlled by the Leap Motion.

The first sound event introduced two audio clips. Clip one was a processed sample of wind to create a drone texture, while the other clip is a hydrophone recording of ocean waves. The wave audio was channeled through a reverb effect whose parameters are controlled by the Leap Motion. The delay time is controlled by the Y position of the left hand; delay time increases as the hand is raised. The X position controls a left to right channel panning effect. Turning the left hand determines the wet/dry ratio (a 0.0value keeps the signal unaffected, a value of 1.0 has the signal completely effected, and 0.5 is a mix) of the effect.

The second sound event introduces two new clips. Clip one was a recording of a water-powered mill, which passed through the spectral delay.

A Max for Live granulation type effect called the Fragulator, which was controlled by the Leap Motion, treated the second clip of water lapping against the pylons of a dock. The Fragulator essentially slices an audio signal into fragments into a size determined by the buffer size input, which is controlled by the Z position of the left hand. When the left hand moves away from the body the buffer size increases. The Fragulator then loops the fragments at varying speeds either forwards or backwards. Turning the left hand controls the wet/dry parameter, and like the previous event the X position controls a panning effect.

The third sound event cued two new audio clips whose effects were the same as the previous event. Clip one is the wave audio being passed through the spectral delay. Clip two, passed through the Fragulator, is a field recording of cicadas and ambient creek side sounds.

Video Effects and Leap Motion Parameters

The three video events were comprised of two videos sent through a pixelating effect and mixed together in real time. Each of the cued events introduced a new set of videos. The cross fading mix was controlled by the turning of the right hand. The pixilation effect, an effect breaks up the image into smaller particles, was controlled by the Z position of the right hand.

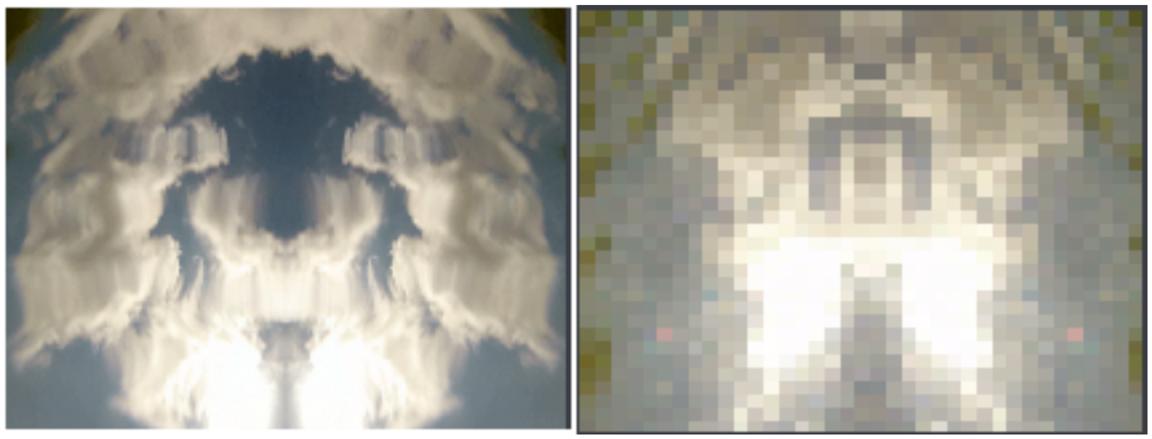


Figure 6.2. Left image shows raw video. Right image shows video with pixelated effect.

The field recordings all share water based sounds so the videos, recorded by the author, that were chosen for the demonstration depict the actions of water in various states of disturbance and activity. Each video received a small amount of processing prior to being imported into the GCPT.

Reflection on The GCPT Demonstration Performance

On November 20, 2015 the author performed a composition made for the GCPT for the defense of his project in lieu thesis. This took place in REVE theatre located at the Digital Worlds Institute of the University of Florida. The sound was played over the surround system and the video was projected on the center screen for the author's thesis committee. The following is a post mortem reflection of the compositional process using the GCPT and an offering of potential improvements with additional insights into using the Leap Motion as a gestural performance controller.

The composition and performance of the soundscape proved to be successful overall. Using the framework of the GCPT the author was able to easily import both the audio sources and the selected videos and create a series of presets called upon by the qlist. Chosen video effects were also easily established using Max's library of Vizzie bpatchers. The parameters of the Vizzie effects were easily manipulated by using the appropriate Leap Motion receive objects that were created to the scale values of 0.0 to 1.0.

Audio effects required some additional programing due to the multitude of possible effects and parameters to control. To fully exploit this area of the GCPT a higher degree of Max knowledge would be useful to a new user, though there are a number of plugin effects that could be used easily with minimal Leap Motion control.

After many practice runs through the demonstration all of the cue events proved to be on target and seamless. The grab strength meter was invaluable for triggering the cues at the determined times. For future consideration other meaningful hand gestures could be conjured to cue events besides the forming of a fist. Potential gestures could be inspired by sign language, street traffic control, or hands held together in prayer to name a few.

The Leap Motion controls over the audio effects also were a success. Each parameter responded as expected to the assigned Leap Motion control. This was also true with the video portion. Negotiating hand gestures with the desired effects during a performance required practice. The author foresees that choreographing a particular gesture to a desired effect should be considered in future compositions to articulate a more meaningful and aesthetic hand gesture performance. The beauty of being able to work within the Max environment with the GCPT framework allows for this kind of creative freedom.

The Leap Motion Controller was able to maintain a robust stream of data throughout many of the practice runs of the demonstration performance. The author found some lag time of the data stream and also occasional times where the Leap Motion ceased to track the hands altogether. These lapses were attributed to the performers hands moving outside of the area of interactivity and the delayed response of the Leap Motion to re-engage with hand tracking. Though much more accessible the Leap Motion shares some of the limited range with Max Mathew's radio baton. Gestures can feel constrained and limiting.

The author's findings were similar to, but less harsh than those found in the 2014 paper *An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking*;

“The Leap Motion Controller undoubtedly represents a revolutionary input device for gesture-based human-computer interaction. In this study, we evaluated the controller as a possible replacement for a fast and high-precision optical motion capture system in a limited space and with a limited number of objects. Based on the current results and the overall experience, we conclude that the controller in its current state could not be used as a professional tracking system, primarily due to its rather limited sensory space and inconsistent sampling frequency. “(Guna et al. 2014)

For future performances with the GCPT ample practice with the Leap Motion area of interactivity is advisable. The author would also recommend that any cue or effected parameter not be dependent on a precisely timed cue or parameter adjustment due to the potential of Leap Motion data being delayed.

Final Performance Data

The data collected during the final performance at the author’s thesis defense can be viewed in the figure below. A line graph was generated to show the grab strength values or cue event (blue), the wet/dry values of the audio effects (red), and the cross fading between videos (green). The x-axis of the graph represents the duration of the performance. The graph shows off the stability of the grab strength value to cue the events.

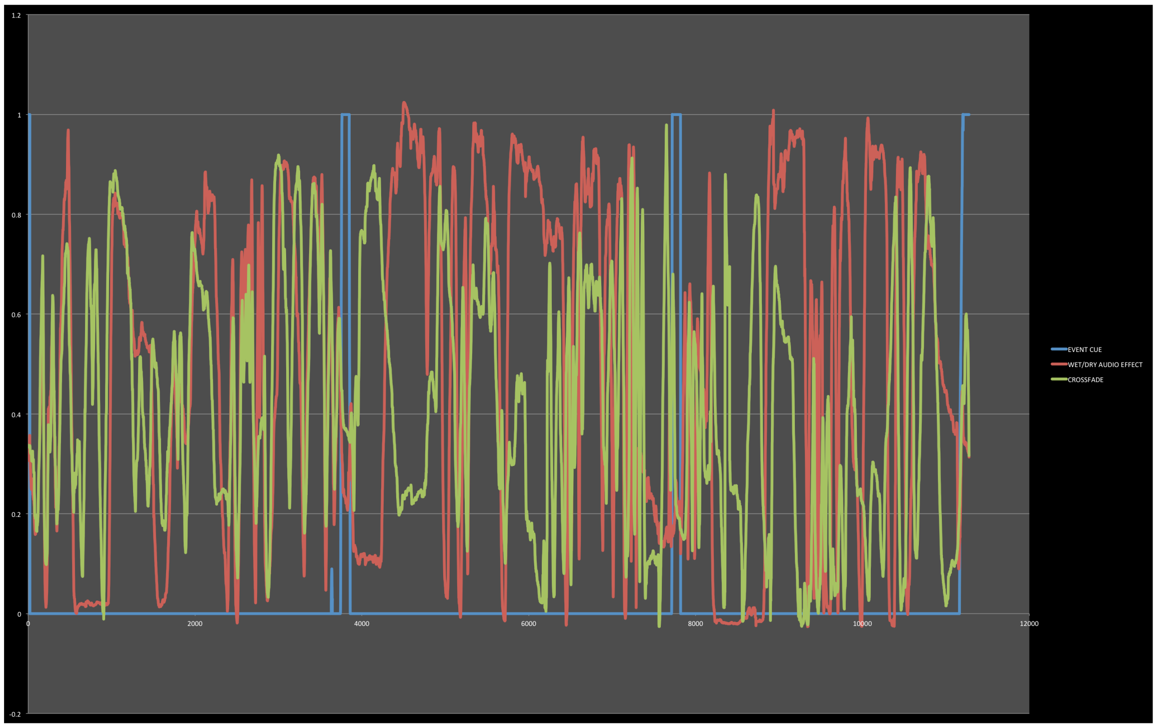


Figure 6.3. Performance data graph.

Conclusion

The GCPT, paired with the Leap Motion Controller, is a viable alternative to performing digitally presented music and soundscapes with enhanced visual content and encourages composers and artists to integrate expressive hand gestures to engage with both the media content and the audience. As the digital performance industry moves further into the twenty-first century artists will be looking to move away from the conventional keyboard, mouse, and touchscreen input devices and seek to control performance parameters with more natural expressive physical gestures. The GCPT represents one of the many steps towards that goal.

Appendix

A. Primary Max Objects Used In The GCPT

The definitions presented here come directly from the reference material provided within the Max software help files.

bang - The bang message is among the most powerful messages in the Max environment; it causes other objects to trigger their output, and can be used to create matrices of connections that produce complex output.

Bpatcher - Abstracts the contents of a patcher or subpatcher for use in other patchers, displaying only those visual elements which are desired.

button - Blinks when you send it any message, and it sends out a bang when you click on it.

change - Used to filter out repetitions of a number

clocker - The clocker object is a metronome that reports the time elapsed since it was started at regular intervals.

counter - Outputs the current count of bang message constrained to a specified range. Can be set to count up, down, or up-then-down.

coll - Allows for the storage, organization, editing, and retrieval of different messages.

comment - displays text which is typed into it in order to serve as a label or explanatory text.

ezdac - Audio output object which can be clicked with the mouse to turn audio on or off.

gate - This object is used to pass or restrict the flow of input to an outlet.

if - Evaluates input according to a conditional statement specified in an if-then-else form.

leapmotion - Ircam’s Leap Motion object used to interface to the Leapmotion.
Based on Leap Motion SDK V2.1.5 Tracking.

line - Generate ramps and line segments from one value to another within a specified amount of time.

message - Displays and sends any given message with the capability to handle specified arguments.

metro - Acts as a metronome which outputs bang s at a regular, specified interval.

nodes - Displays nodes in a 2-dimensional space, and calculates the distance from a node points.

pack - Combine items into an output list. The arguments determine the list format and types of the list elements.

patcher - This object creates patches within patches.

pattr - Stores its own data, or binds to another object to share its contents with other pattr-based objects (such as pattrstorage). Can be used for data routing or preset creation.

playlist - Used to organize sets of soundfiles and play them back. Each sound is given a visual representation in a clip where a selection from the entire sound file may be choosen.

qlist - Stores a collection of timed or untimed "cues" in the form of messages which can be sent either out its outlet or remotely to various receive objects in a patch.

receive - Receives and outputs messages from send objects sharing the same name. This allows the user to send any kind of message between Patcher windows or within a window without using patch cords.

route - Tries to match a message's first argument to the route object's own arguments.

scale - The scale object maps an input range of float or integer values to an output range.

sel - Selectively outputs a bang in response to any input which matches its arguments and will output non-matching messages out its right-most outlet.

send - Used to send messages without patch cords to receive objects with the same name.

toggle - toggle sends a 0 as output when it is turned off and a 1 as output when it is turned on (when giving input, a non-zero number will turn it on, a 0 will turn it off, and a bang will alternate the state of the toggle).

udpreceive - Receives messages transmitted over a network using the User Datagram Protocol (UDP).

zmap - Maps an input range of values and to an output range of values. Similar to scale, but clips values to the ranges, and does not allow inverted scaling.

LIST OF REFERENCES

Abel, Jonathan. Pekonen, Jussi. Smith, Julius. Välimäki, Vesa. "Spectral Delay Filters With Feedback and Time-Varying Coefficients*". Proceedings of the 12th International Conference on Digital Audio Effects(DAFx-90)*, Como, Italy, September 1-4, 2009.

Austin, Michel. Tayeb, Monir. 2011. "Berlioz in Paris."
<http://www.hberlioz.com/Paris/BPIndustrie.html. >
Accessed Nov. 08 2015.

Bellona, Jon."jpbellona/simpleKinect".
<http://jpbellona.com/kinect/> Accessed Nov. 10, 2015.

Bellona, Jon. "Human Chimes".Web.
<https://vimeo.com/32891916> Accessed Nov. 10, 2015.

Elsea, Peter. "Max and Programming".
< http://peterelsea.com/Maxtuts_advanced/Max&Programming.pdf> Sept. 29 2007.

Guna, Jože et al. "An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking". *Sensors* 2014, *14*, 3702-3720

Hantrakul, Lamtharn. Kaczmarek, Konrad. "Implementations of the Leap Motion in Sound Synthesis Modulation and Assistive Performance Tools". *Proceedings of ICMC,SMC*. Athens, Greece, 2014

IRCAM/ Sound Music Movement Interaction. "Leap Motion Skeletal Tracking In Max", <http://ismm.ircam.fr/leapmotion/> Novemeber, 7 2014.

Johannsen, Gunnar and Nakra, Teresa. "Conductor's Gesture's and Their Mapping to Sound Synthesis". *Musical Gestures: Sound, Movement, and Meaning.* Godøy, Rolf and Leman, Marc Editors. New York, NY. Rutledge, 2010. Print.

Leap Motion. "How Does the Leap Motion Controller Work?", <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/> August 9, 2014.

Leap Motion " Coordinate Systems"
<https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html> Accessed Nov 1, 2015.

Lehrman, Paul. "Nintendo's Will Remote as Midi Controller". *Sound On Sound.* Oct. 2008.
<https://www.soundonsound.com/sos/oct08/articles/wiimote.htm>

Ratcliffe, Jarrod. “Hand Motion-Controlled Audio Mixing Interface”. *Proceedings of the International Conference for Musical Expression*. 2014 <http://www.nime.org/proceedings/2014/nime2014_518.pdf>

Ritter, Martin. Aska, Alyssa. “Leap Motion As Expressive Gestural Interface”. *Proceedings of ICMC,SMC*. Athens, Greece, 2014.

Roads, Curtis. *The Computer Music Tutorial*. Cambridge, MA: MIT Press, 1996. Print.

Winkler, Todd. *Composing Interactive Music: Techniques and Ideas Using Max.* Cambridge, MA: MIT Press. 1998. Print.

Wikipedia Sources

“IRCAM.” *Wikipedia, the Free Encyclopedia.* Web. 7 Nov. 2015 <https://en.wikipedia.org/wiki/IRCAM>

“Kinect.” *Wikipedia, the Free Encyclopedia.* Web. 11 Oct. 2015. <https://en.wikipedia.org/wiki/Kinect>

“Max (Software)” *Wikipedia, the Free Encyclopedia.* Web. 18 Oct. 2015. < https://en.wikipedia.org/wiki/Max_(software)>

“Theremin.” *Wikipedia, the Free Encyclopedia.* Web. 03 Nov. 2015. <https://en.wikipedia.org/wiki/Theremin>

Software Developers

Ableton - Homepage. Web. 2015. <http://www.ableton.com>.

Cycling 74. Web. 2015. <http://cycling74.com/>.

Leap Motion. Web. 2015. <*https://www.leapmotion.com*>

Bibliography

Bayle, Julian. *Max For Live Ultimate Zen Guide: Become a Max for Live Master and discover a new way of Using Abelton Live.* Lean Publishing, 2013. eBook

Cipriani, Alessandro. Giri Maurizio. *Electronic Music and Sound Design: Theory and Practice with Max and MSP, Vol. 1.* Rome, Italy: ContempoNet s.a.s., 2009. Print.

Cipriani, Alessandro. Giri Maurizio. *Electronic Music and Sound Design: Theory and Practice with Max and MSP, Vol. 2.* Rome, Italy: ContempoNet s.a.s., 2013. Print.

Cycling ’74. “Leap MoDULAtion”.
<https://cycling74.com/toolbox/leap-modulation/> Oct. 3 1013.

Farnell, Andy. *Designing Sound.* Cambridge, MA: MIT Press, 2010. Print.

Kim-Cohen, Kim. *In the Blink of an Ear:Toward a Non-Cochlear Sonic Art.* New York, NY. Bloomsbury, 2009. Print

Manzo, J. V. *Max/MSP/Jitter for Music: A Practical Guide to Developing Interactive Music Systems for Education and More.* New York, NY: Oxford University Press, 2011. Print.

Roads, Curtis. *Composing Electronic Music: A New Aesthetic.* New York, NY: Oxford University Press, 2015. Print

BIOGRAPHICAL SKETCH

Thomas Doughty is a Master's Degree student at the University of Florida's Digital Worlds Institute and holds a BA in Anthropology from San Francisco State University. His focus is in digital projection, sound design and electroacoustic music composition. He has worked with UF's College of Education to assist in the development of a brain training lab; a series of computer games to foster self-regulation and improve social/emotional outcomes for middle school students with emotional & behavioral disorders (EBD). Thomas was the lead programmer for the Digital Worlds produced interactive eBook *How The Heart Works* for Shands Hospital Arts and Medicine. He presented his paper, *Lost (and Found) In Your Head: The Significance of Binaural Diegetic Audio in The Nightjar*, at the 2015 North American Conference On Video Game Music at Texas Christian University. Tom is a photographer, videographer, and field recordist. He has recently worked for New York State Parks photographing over 50 parks for their virtual tour website.