# *Interactive Projections: The Mechanics of Interactive Video Design*

An Honors Thesis (HONR 499)

by

Selena Webb

Thesis Advisor

Mr. Brian Moore

Ball State University
Muncie, Indiana

May 2019

Expected Date of Graduation
May 2019

## Abstract:

As an incoming Freshman at Ball State, I knew two things for certain: I had a passion for theatre, and I wanted to pursue a career in video. For all I knew, these were separate worlds. Theatre is based on live performance, and is meant to be beautiful in the moment. Often times video recording is prohibited in theatrical performances not just for legal reasons, it is simply not how the art form is meant to be consumed. Theatre is meant to be seen and experienced live and in person. Video on the other hand exists in a world of re-takes and post enhancements that allow it to be fine-tuned into a perfect and deliverable product. Combining the worlds of live and raw with edited and specific was something that I only imagined was possible, that is, until I found projections.

Projection design is the art of creating and integrating assets such as video clips and motion graphics, with technology through projectors, cameras, monitors, even LED walls. This can be used to enhance a performance for theatre, dance, concerts, etc. However, projections can be taken one step further. Instead of simply playing content and projecting it into a space, I wanted to explore interacting with the content as well. When it comes to the interactive world, there are an infinite amount of possibilities, and from all of my projection design experience from designing theatrical shows, I have never been asked, or challenged, to do an interactive piece. For my Honors Thesis Project, I would like to combine both research and creative components to create an interactive projections piece.

## Acknowledgments:

I would like to thank Mr. Brian Moore for advising me through this project. His technical guidance and unwavering moral support and belief in my abilities was essential to the completion of this project.

I would also like to thank Joe Court and Kris Kirkwood for their continual advice and support in encouraging me to pursue this daunting task.

**Process Analysis Statement**

As projects go, I began with research. What are all of the things I need? I boiled it down to three specific areas: camera, projections program, and media server. The following is how I came to choose the devices that I used in my final thesis, and how I implemented them into a working interactive system.

## Camera

A camera was integral to my set-up because I needed a way to capture moving data in order to have an interactive component to the project. Just like you might see in a movie theatre, anything that a projector outputs is better seen when there are no additional lights. For projections to show up well, there needs to be little light within the space it is being projected in. This is the reason lights are dimmed when a movie begins in a theatre: it enhances the experience, as well as allows for a clearer image. Therefore, I concluded that an infrared camera was the perfect solution for this particular project. Infrared allows a camera to function in low light because instead of being concerned with data extracted from the amount of light in the room, it can collect data based off of heat signatures.

As for the kind of infrared camera, I knew I needed something that would actively talk to my projections mapping program, Isadora, which I will further explained in the next section. I found that the Xbox Kinect was an inexpensive solution and suited all of my needs. Kinect cameras were a line of cameras produced by Microsoft. At first, the Kinect was made and used as a gaming accessory for the Xbox 360 and other video game consoles. Kinect's have a motion sensing input. Through a camera, depth sensor, and an internal infrared laser projector, it is able to capture video in 3D. Although this device was originally created for the gaming world, several developers and creatives quickly started using it for their own projects due to its' stellar motion capture ability.

One of the first roadblocks that I encountered was that the Kinect camera is formatted to work on a Windows computer. My laptop is a Mac, so naturally there were some issues with connectivity. To problem solve this I looked for drives that I could install onto my personal computer and used those to allow the two devices to speak to each other. There was a project that I found on GitHub, a code sharing website, that had software written specifically for connecting the Kinect device with Mac OS software. I then downloaded a software called Processing. Processing is a flexible software sketchbook and language tool for learning how to code within the context of the visual arts. By using this program, and implementing the Kinect language coding from the Github project, I was able to establish a working connection between my Microsoft Kinect camera and Mac computer.

## Projections Program

Next, I needed to find a projections program that would let me create the interactive commands. Essentially, a program to take in the data from the Kinect camera and turn it into usable information that I could create effects for. For this, I chose Isadora. Isadora is a projections program that artists, designers, and performers who want to add video and interactive media to their performance projects use. The software is an interactive media playback platform that allows for the control of display for video and media assets. It can be used for programming of cues and visual effects and for the control of how media appears once it is projected. Is it playing only in the upright corner? Are there 12 clips playing over the screen at once? Is it a circle, rectangle, triangle? This is all customizable within the program.

To start, I needed to beta test that I could get a graphic to appear based off of movement, any movement. At this point in the process I had yet to select my camera. I made a test program that created content based off of my mouse movement. Photo (1) below is no movement, and Photo (2) shows a dotted line graphic that followed my mouse. Next, all I needed to do was have Isadora recognize my hands and produce a graphic from human movement, instead of mouse movement. It sounded easy in theory, but this was actually one of the longest portions of the project.

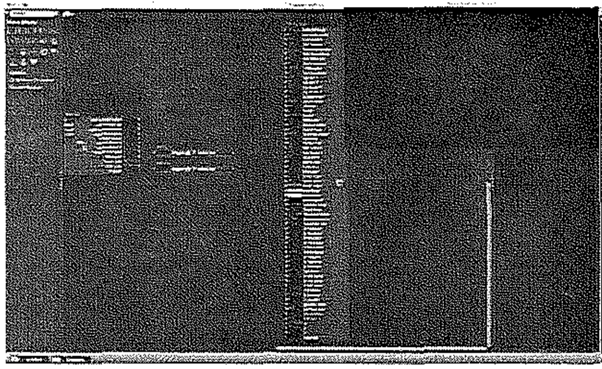


Photo 1

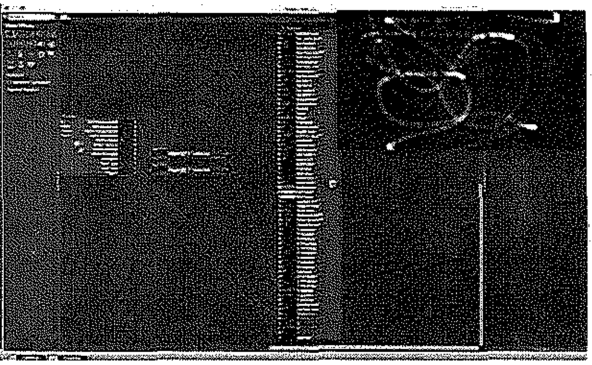
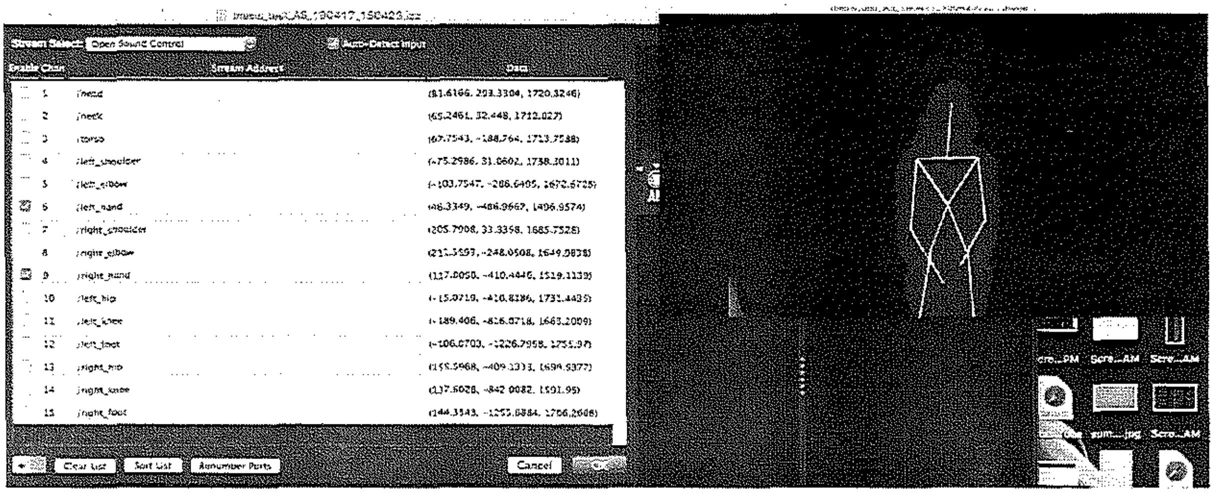


Photo 2

Once I got the Kinect to talk to Isadora I worked with a preview window that would actively show me what was being seen by the Kinect device. This can be seen below in Photo (3). Using the code that I implemented with Processing, I was able to see skeleton data tracking major points on the body: torso, hands, legs, feet, etc. By viewing the streamline set up, (Photo 3), I was able to label all of these for clarification. However, since I was only interested in "drawing" for the interactive part of it, I disabled all of the channels that were not the right and left hand. This proved to be a good choice due to the overwhelming amount of data that the Kinect camera processed later on in the project.

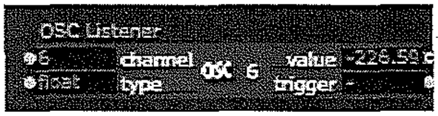


Stream Select: Open Sound Control ☒ Auto-Detect Input

| Enable Chan | | Stream Address | Data |
|---|---|---|---|
| □ | 1 | /head | (81.6166, 293.3304, 1720.3246) |
| □ | 2 | /neck | (65.2461, 32.448, 1712.027) |
| □ | 3 | /torso | (67.7543, -188.764, 1713.7588) |
| □ | 4 | /left_shoulder | (-75.2986, 31.0602, 1738.3011) |
| □ | 5 | /left_elbow | (-103.7547, -286.6405, 1672.6725) |
| ☒ | 6 | /left_hand | (46.3349, -486.9667, 1406.9574) |
| □ | 7 | /right_shoulder | (205.7908, 33.3358, 1685.7528) |
| □ | 8 | /right_elbow | (211.5597, -248.0508, 1649.0838) |
| ☒ | 9 | /right_hand | (117.0050, -410.4446, 1519.1139) |
| □ | 10 | /left_hip | (-15.0719, -410.8186, 1731.4435) |
| □ | 11 | /left_knee | (-189.406, -816.0718, 1663.2009) |
| □ | 12 | /left_foot | (-106.0703, -1226.7958, 1755.97) |
| □ | 13 | /right_hip | (155.5968, -409.3313, 1694.9377) |
| □ | 14 | /right_knee | (137.6028, -842.0082, 1591.95) |
| □ | 15 | /right_foot | (144.3543, -1255.6884, 1706.2608) |

Photo 3: Streamline set up (Left), Skeletal Data though preview window (Right)

Through research, I discovered the way that the Kinect would talk to Isadora is through Open Sound Control, or OSC. Open Sound Control (OSC) is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. I implemented an OSC actor into my workspace. An actor (which is an Isadora program term) is simply a tool that I can drag into my workspace and begin manipulating. An example of what an actor looks like in the program, specifically an OSC actor, can be found in Photo (4) below.



OSC Listener

6 channel

float type

value -226.59

trigger -

Photo 4: OSC Listener Actor

With the OSC actor I was able to customize what channel, or body part, I wanted to receive data from. For my project purpose, channel 6 related to the left hand, and 9, the right. If you notice, in Photo (4) above, the value has a specific number attached to it. Originally, I was unsure which value the actor was outputting. In a 3D space there are three values to be concerned with: x, y, and z. The x coordinate is right to left, the y coordinate is top to bottom, and the z coordinate (which is what makes it 3D) is forward and backward within the space. The fact that only one value was outputting became a problem because all three needed to be active for the movement to react correctly. This was an issue I struggled with for a while in this project, so I took to the internet to research some possible answers.

Fortunately, I found a forum on the Troikatronix (the company that makes Isadora) website that pointed me in the right direction. Due to the fact that I had assigned numbers to the channels, and disabled all but two (for the hands), meant there were several numbers unused. I routed two new OSC actors, and gave them the value of the next two corresponding channels. In the case of the left hand, 7 and 8, the value output was no longer just one coordinate. It was now able to output x, y, and z values by using all three actors. When cross referenced with my original streamline set-up (photo 3), it proved that the 7 and 8 channels were now outputting the y and z axis of the left hand. I repeated this step for the right hand as well. An example of the added OSC actors can be seen below in Photo (5).

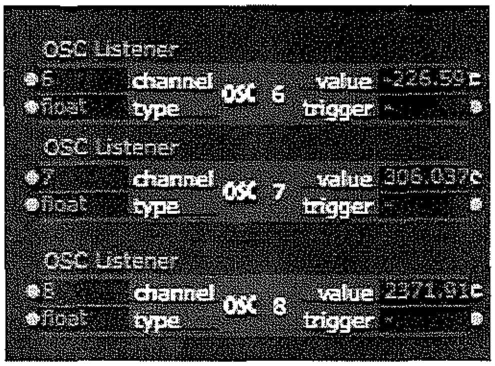

OSC Listener
6 channel OSC 6 value -226.59
float type trigger
OSC Listener
7 channel OSC 7 value 306.037
float type trigger
OSC Listener
8 channel OSC 8 value 2371.91
float type trigger

Photo 5

Unfortunately, even though the values were updating based off of movement from my subject, no graphics appeared on screen, as they had when the graphics were connected to my mouse movements. I hypothesized that this could be because the values were too extreme for the computer to handle, to fix this, I added a calculator actor which divided the numbers into a readable scale for the program. This brought down the values from three digits to two. This can be seen below in Photo (6).

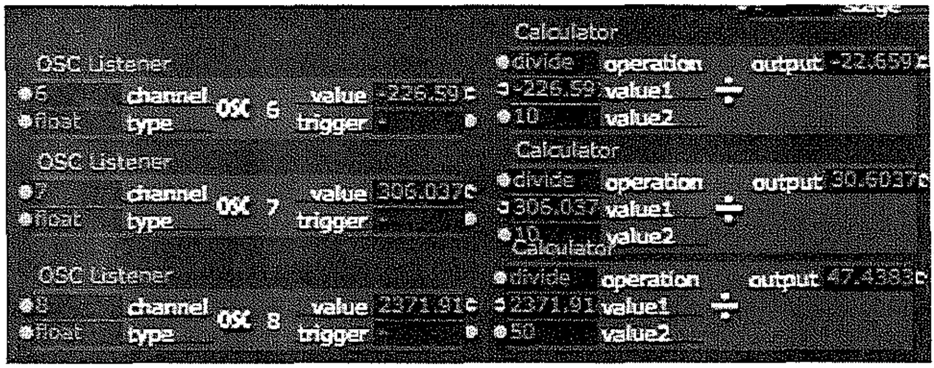

OSC Listener
6 channel OSC 6 value -226.59
float type trigger
Calculator
divide operation output -22.659
-226.59 value1 ÷
10 value2
OSC Listener
7 channel OSC 7 value 306.037
float type trigger
Calculator
divide operation output 30.6037
306.037 value1 ÷
10 value2
OSC Listener
8 channel OSC 8 value 2371.91
float type trigger
Calculator
divide operation output 47.4383
2371.91 value1 ÷
50 value2

Photo 6

Now that the values were all being communicated, I ran into another issue. Some values, based on where my subject was, were appearing as negatives. This was confusing the system, and rendering the interactive graphics to be glitchy, or unresponsive all together. To fix this, I added an absolute value coordinator. This allowed all outputs to be read as positive figures, finally allowing the interaction to work properly. This can be seen below in Photo (7).

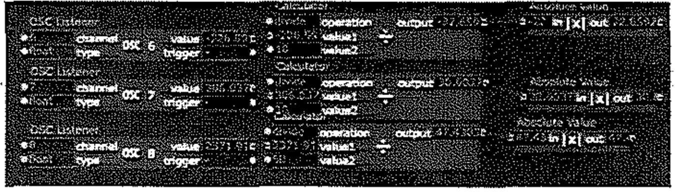


Photo 7

## Media Server

As an additional challenge, I wanted to not only make a working interactive projection model, I also wanted to learn how to use some of the more advanced equipment that the theatre department had, and was not being utilized. Green Hippo is a company that creates a media server device called the Hippotizer. A media server is a computer appliance dedicated to storing various digital media files, in this case, video and graphic files. These servers are specialized for media streaming, and can help compute large files, at high resolution, without overworking your personal computer. A Hippotizer processes video feed and creates a high resolution playback. It is an industry standard, and although I was only able to work with a version 3, (somewhat outdated now), it allowed me to learn valuable transferable information. I knew that by challenging myself to learn this machine as well, it would prepare me for working with these machines in the industry.

To get my Isadora program to connect to the Green Hippo media server, I needed a way for my two devices to talk to each other. To do this, there is a tool called Art Net, which is a communication protocol. A communication protocol is a system of rules that allow two or more entities of a communications system to transmit information to one another. In this case, that would be my Isadora program, sending information to the Hippotizer using Art-Net. Art-Net is a simple implementation of DMX512; a protocol in which lighting control information is conveyed in IP packets, on a private network such as ethernet. Supported functions include transmitting and receiving lighting data, and in this case, video data. First, I started with a wired hookup, directly connecting my laptop to the Hippotizer machine through ethernet, Photo (8).

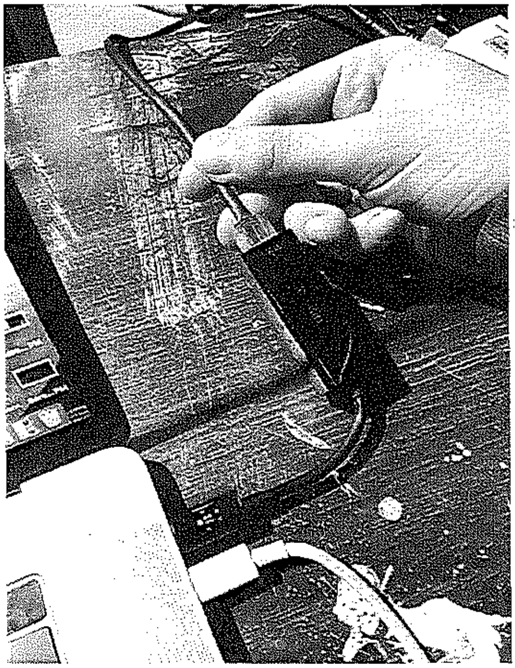
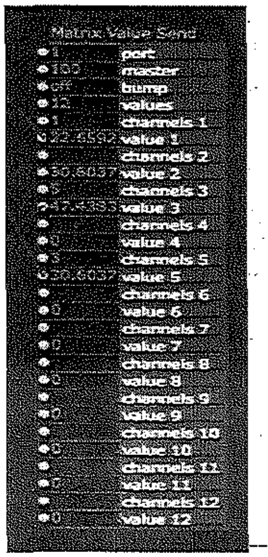


Photo 8

To implement Art Net, I then added some Art Net and Matrix Value Send and Receive actors. This set up a communication line between the devices. See below for signal flow in Photo (9) and (10). Essentially, the Matrix values were all assigned channels from the x, y, and z outputs of the OSC actors. It was then converted into 512 hexadecimal codes, that were then sent out through ethernet, and inputted into the Hippotizer program.



```
Matrix Value Send
1         port
100       master
off       bump
12        values
1         channels 1
22.4592   value 1
          channels 2
30.6037   value 2
5         channels 3
47.4383   value 3
          channels 4
0         value 4
3         channels 5
30.6037   value 5
          channels 6
0         value 6
          channels 7
0         value 7
          channels 8
0         value 8
          channels 9
0         value 9
          channels 10
0         value 10
          channels 11
0         value 11
          channels 12
0         value 12
```

--- >

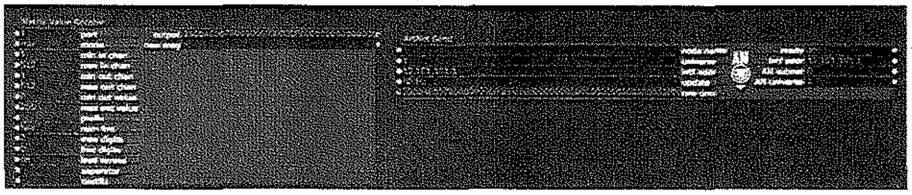


Photo 9 Photo 10

Next, I programmed my computers' Network IP (Internet Protocol) address into the actor, so the information could be sent from my computer, into the media server's machine.

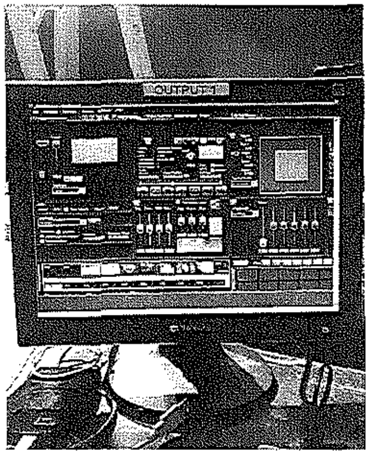


Photo 11

Above, you will see the main set up of the Green Hippo Hippotizer software in Photo (11). The bottom portion shows the folders where media files can be stored. This is where I was able to implement my own files. For this project, I created a base layer to look like traditional Japanese Sumi paper, Photo (12), as well as an inkbrush like stoke to drag across the screen, Photo (13).

The rest of the controls seen in the window above is filled with various values, colors, and scales that I can use to edit the video content how I'd like.

Photo 12

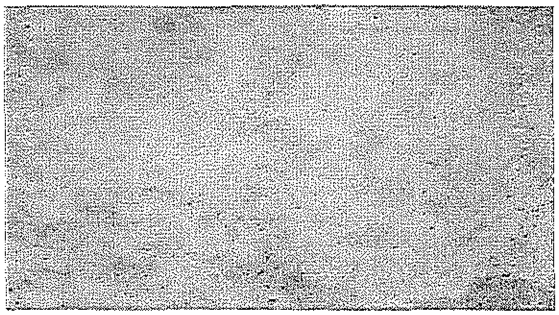


Photo 13

Although I was able to load my own content into a video folder on the device, the interactive component, or the brush stroke, was not yet working through this machine. After researching, I realized that just as I had to connect the Xbox Kinect to Isadora by using an OSC actor, I had to do something similar with the Hippotizer. So, I used something called Pin Bridge, a feature within the Hippotizer program that helps connect the x, y and z nodes, or data points, from the Kinect camera to the media server.

To do this, I went to the User Menu, and opened up the "Pin Bridge" function. From there, I had to manually add all of the point values that I wanted the media server to recognize. For example, in my Isadora program, each hand corresponded to 3 points of an x, y, and z value. So I created 6 pin bridges in order to correspond, and match the data inputted from Isadora. Next, I had to go in and manually select where I wanted each pin to receive data from. By clicking the add button, I navigated through the servers' connections and chose my selected layer, in this case a mask, and connected the layer to the positional values outputted by Isadora on the respective x, y, and x coordinates. Documentation of the menu navigation is provided below in photos 12, 13, and 14.

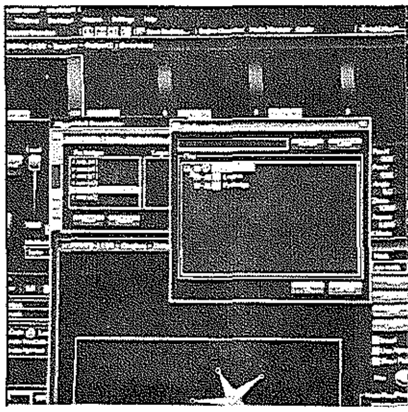


Photo 12

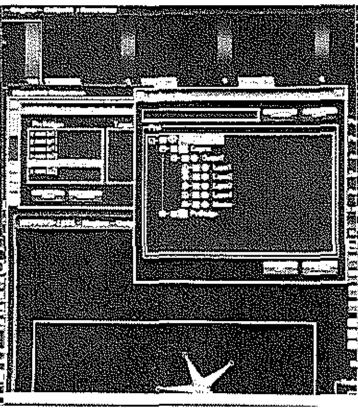


Photo 13

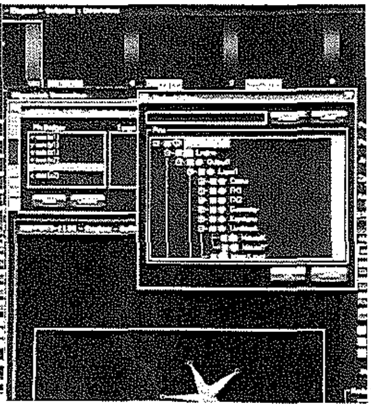


Photo 14

After hooking it up to a projector, now, I finally had a working model of an interactive projection using an infrared camera, a projection mapping program, a media server device. See photos 15-18 below for documentation.

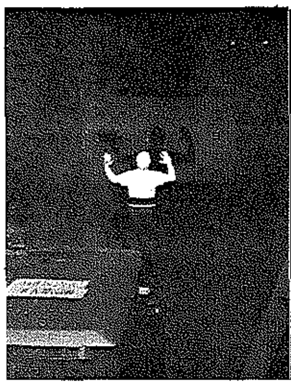


Photo 15

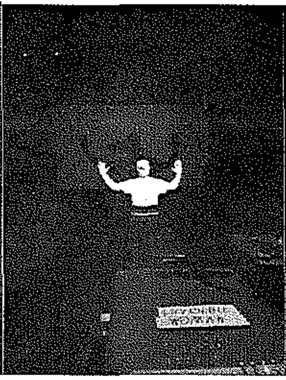


Photo 16

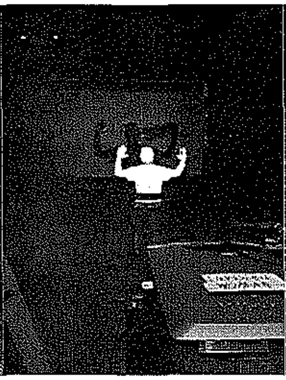


Photo 17

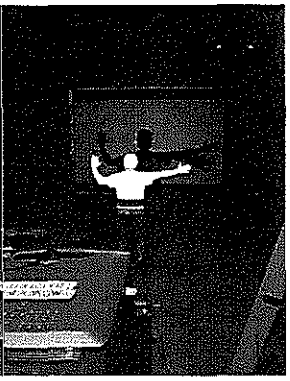


Photo 18

## Future Plans

While working with the Hippotizer, I ran into a lot of issues trying to create the graphic content that I wanted. Not only was this an older model of a media server, and therefore did not have as many features as I would have liked, a lot of times the system became overworked, and would continually crash. This turned five hour work sessions, into hours of rebooting and restarting equipment, hoping it wouldn't crash again. My thesis advisor was wonderful in helping me fix the mechanical issues I was having, and all in all, it helped me understand how to go about problem solving mechanical issues in the future.

Although the goal of the project was to learn how to create a working interactive projections model, and was not necessarily focused on the design aspect, I want to continue working on manipulating footage, and applying effects to it with the Hippotizer machine. Within my research, I also found several other ways to generate content that would continually interact with a moving subject. I want to further explore programs like Notch and grow in both my technical and design skills within the projections field.

When I set forth on this project, I wanted to understand, not just how to design an interactive piece, but instead, how to make one function. I wanted to take the time to understand all of the technical elements, collect the gear needed, build a program, and hook it all up myself to make a solid functioning signal line.

The theatre department itself also does not have an active program in teaching projections, as it does with lighting, costumes, make-up etc. I hoped that by showing such a great interest in using the gear and tech that the university already owned, that there would be interest built, and curiosity peaked by the other students, enough to explore this area in the future. 5

## Works Cited

Brown, Matt. "Kinect May Be Dead, but Still Has Its Place in 2018." *Windows Central*, Windows Central, 8 Jan. 2018, www.windowscentral.com/what-do-kinect-2018.

BUT. "The World's Largest Interactive Video Wall Using Kinect - Demo 2." *YouTube*, Better Use of Technology, 29 Aug. 2012, www.youtube.com/watch?v=2OzIssBt4J0.

Griffiths, Jamie. "Kinect into Isadora." *Jamie Griffiths*, 2018, www.jamiegriffiths.com/kinect-into-isadora/.

Griffiths, Jamie. "Kinect with Isadora." *Jamie Griffiths*, 2017, www.jamiegriffiths.com/kinect-with-isadora/.

Griffiths, Jamie. "New Kinect into Syphon." *Jamie Griffith*, 2017, www.jamiegriffiths.com/new-kinect-into-syphon/.

Hippo, Green. "Hippotizer V4 - Component Structure." *YouTube*, YouTube, 13 Mar. 2017, www.youtube.com/watch?v=szN39NltguE&list=PLxUm-kumwNQSASoxIYiIwWCSZmuWfCp_8&index=2.

Hippo, Green. "Hippotizer V4 - Component Structure." *YouTube*, YouTube, 13 Mar. 2017, www.youtube.com/watch?v=szN39NltguE.

Hippo, Green. "Training." *Green Hippo*, www.green-hippo.com/support/training/.

"Isadora Kinect Tracking Tutorial - Part 2." *TroikaTronix*, 2016, troikatronix.freshdesk.com/support/solutions/articles/13000014929-isadora-kinect-tracking-tutorial-part-2.

Learn Stage Lighting. "What Are Art-Net and SACN?" *YouTube*, YouTube, 1 May 2017, www.youtube.com/watch?v=OwzVgKEMs0Q.

Mawad, Tarak. "Bioluminescent Forrest." *3hund*, www.3hund.com/. Accessed 3 Jan. 2019.

Microsoft. "Download Kinect for Windows." *Download Kinect for Windows Runtime 2.0 from Official Microsoft Download Center*, 2018, www.microsoft.com/en-us/download/confirmation.aspx?id=44559.

"Using TouchOSC with Isadora." *TroikaTronix*, support.troikatronix.com/support/solutions/articles/5000692145-using-touchosc-with-isadora.