Gestural Music

Andrew Hollenbach, Rochester Institute of Technology

Matthew Cox, Eastman School of Muisc

Independent Study Final Report

May 21, 2015

1. Introduction

Gestural Music is a project aimed at leveraging the Kinect's capabilities to create a musical experience. Through the system we have developed, people of all experience levels are able to create music through movement of their body – in particular, their arms, legs, and hands.

The project began in the fall semester through Professor Joe Geigel's Applications in Virtual Reality course. The original team was comprised of myself, Matthew Cox, Vishwanath Raman, Emily Abele, and Ameya Lonkar. The project has been continued into the current spring semester under the direction of myself and the musician, Matthew Cox.

This paper builds off of our previous report, which defines the system, our analysis of the hardware and software, and more. This paper assumes previous knowledge of our system as defined in this paper. The original report can be found on GitHub.

In this paper, we describe our work developing a demo mode, a projection mode, and introduce several improvements made to the original instrument.

2. Analyzing Our Previous Work

At the beginning of the semester, Matthew and I spent a long time thinking about potential ways to improve the system. One of Matthew's primary concerns regarding the system is that it can be mentally and physically exhausting to play for long durations of time (greater than roughly 30 minutes). We believe there are a few contributing factors to this effect.

First off, the system has a diverse set of controls, which can be hard to track. For a power user who has experience with the system, it becomes easier over time, but for a person just starting out, the system can be very confusing and frustrating. Interestingly, the general consensus from people trying the system has been that even though the system wasn't inherently doing what they wanted, they were still really enjoying it. From this, Matthew and I developed a demo mode, and allow users to switch in between the demo and advanced modes. This will be further described in Section 4.

Additionally, we aimed to further reduce mental strain, as well as potentially improve overall system tactility by adding a projection element. By providing additionally information to the user directly below them, they can focus more on what their hands are doing and are not straining to see what the small monitor is trying to convey. This will be further described in Section 5.

The lack of physical feedback and support can also cause strain. Users have to maintain their left arm in a lifted position for playing accurate notes. Unfortunately, we could not think of a solution for this, aside from the fact that the demo mode encourages users to move their body more, which is easier on the muscles.

Lastly, our previous releases had a few bugs and user experience issues with the instrument itself. These issues have been addressed and changes will be highlighted in Section 3.

3. Improvements to the Instrument

The system has received a visual overhaul. Background flashing has been removed, as it was a very non-intuitive way to alert the user of information. A small box has been added to the top of the screen to show the current partition. A color scheme has been developed, and colors are gentler and more user-friendly and kid-friendly.

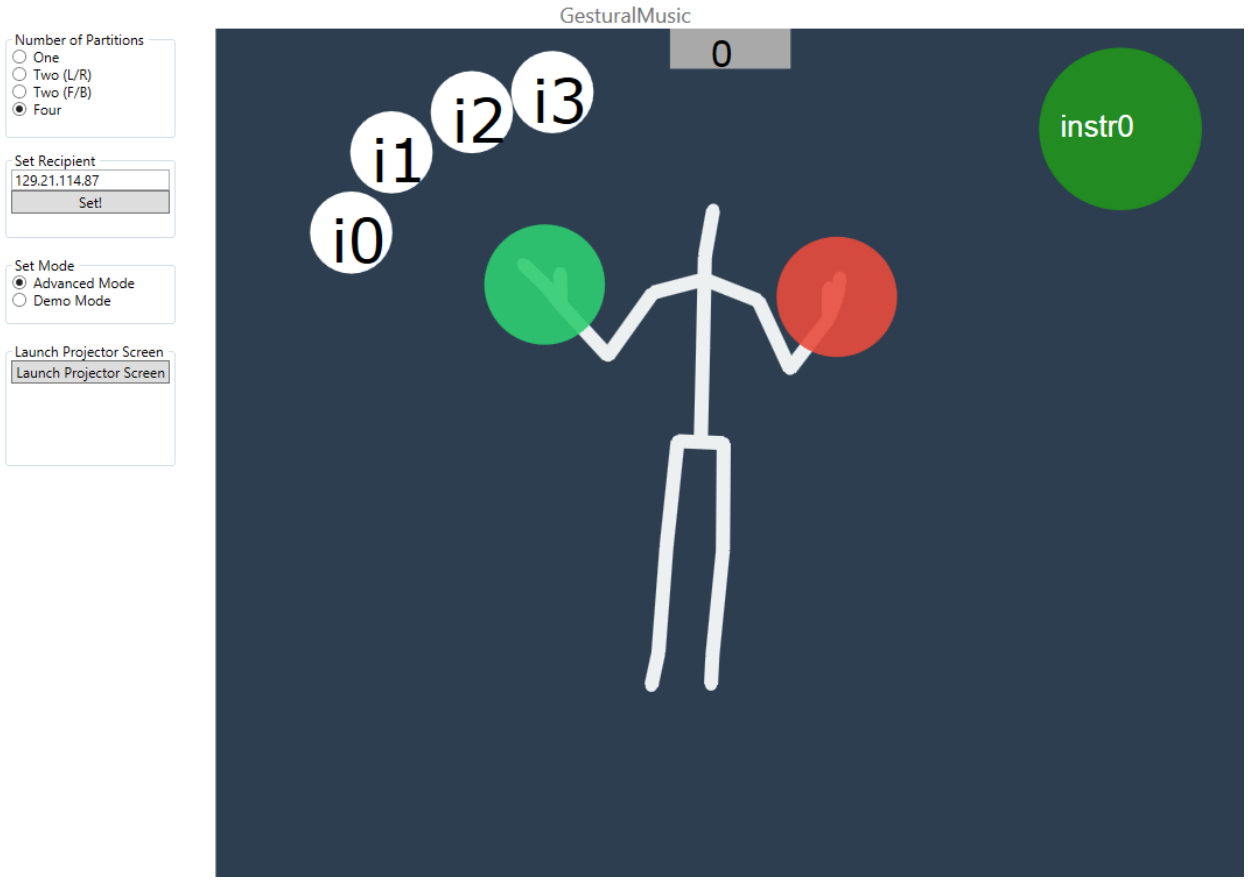

Figure 0: The interface

Another usability issue was the initiation step of extending your arms and making a double-lasso. A new user entering the system would have to restart the application to reset the arm length. This initiation step did not provide a significant accuracy increase, and therefore has been removed.

A new system of filtering has also been introduced. This was a goal of the original project, but ultimately, time ran short and it was not completed. The right hand/wrist is used for filtering. The system uses the right wrist's location relatively to the body shoulder to produce values from 0 to 1 to feed into Ableton Live. Filters send on the OSC signal `/instrName/filter<X|Y|Z>` and are guaranteed to be a value between 0 and 1. For example, the OSC signal might be `/instr0/filterX 0.78`. The Y value is marginally bended with a scaling function, since the human arm rotates circularly, yet values are read in a cube relative to the shoulder. Because the wrist will never reach a value of one under normal circumstances, we boost the value non-linearly such that it does.

Additionally, we incorporated a volume control based on proximity to the Kinect sensor. The closer you are to the sensor, the louder it becomes.

Lastly, the previous release of the system only supported selection of three instruments. This has been expanded to support four, given that there are four partitions.

4. Demo Mode

One major improvement to our system is the introduction of a demo mode. Users are able to seamlessly switch between the modes via the buttons on the left-hand side.

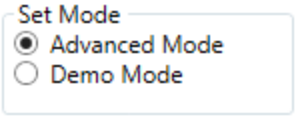

Figure 1: Toggle to switch between modes

Demo mode, at its core, borrows a lot of functionality from its advanced mode parent, but does things a little differently. The ability to partition the stage is removed and if there are any partitions, your stage is reduced to a single partition. The concept of partitions reduces mobility and increases complexity and therefore is not accessible.

Secondly, the way notes are actually sent is changed. Your right hand no longer controls when a note is played. Rather, the system will play whenever your left hand is open. You must have set up a track that receives OSC signals on `instr0`, but instead of the system using the

MaxOSCInstrument patch I created for the advanced mode, it uses the Max NoteDial from the Synapse library. Map as seen in Figure 2.

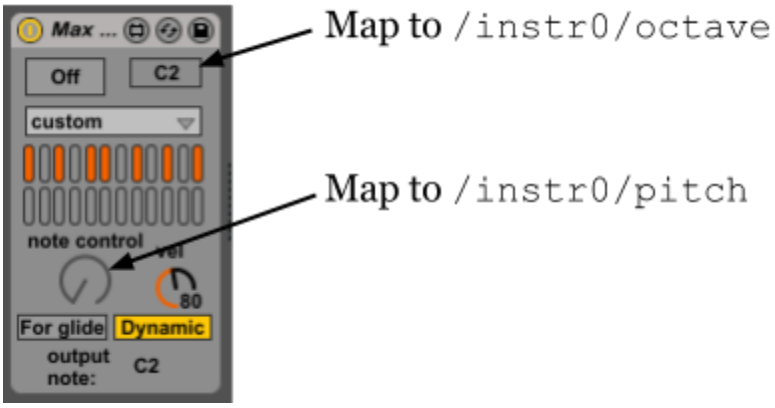

Figure 2: Max NoteDial

Additionally, because of the way it's mapped, we're going to need to use Max API OSC plugin (linked in the repository).

At any time, you can switch back to advanced mode by selecting that option in the menu.

5. Projection Mode

Another improvement to our system is the addition of a projection mode. To experience projection mode, there a few steps that you must take. Connect your projector and set your screens to extend, rather than mirror. Projector mode provides a second window (in addition to our already-existing monitoring and control window), so it is best if the projection window is full-screen on your projector, while still having the monitoring window on your monitor.

The first task is to align your projector. Unfortunately, you will need to already have a working knowledge of your approximate projection area - the reach of the Kinect sensor. This can be tuned later in the process, which will be described momentarily. The projector needs to be pointed at the floor. Note that a projector with a high number of lumens and wide spread needs to be selected, as the orientation you set it up in will probably be such that the image is extremely stretched.

To activate the projector window, click the “Launch Projector Screen” button.

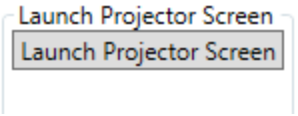

Figure 3: Launch button for projector screen

The window should appear similar to Figure 4. This is what the system looks like on a flat monitor. However, once dragged onto a projected floor, it will make more sense.

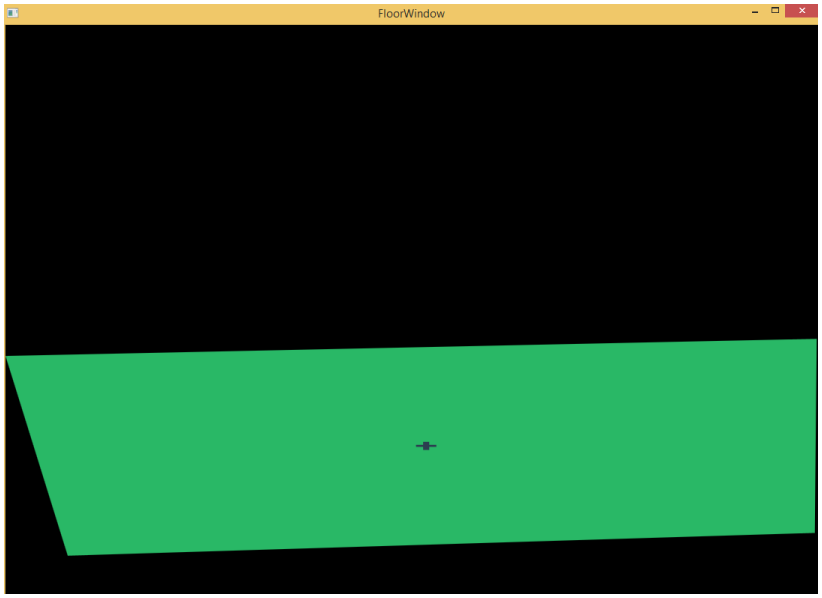

Figure 4: The projection screen

Drag the window that pops up over to your projector and set it to full screen.

The original idea of the system was that from this point on, the Kinect and the projection window would calibrate, such that the green area automatically aligns to the vision area of the Kinect. Unfortunately, I was unable to determine a way to do this, and ultimately had to go for a system similar to that of SurfaceMapper for Processing, where you manually drag the corners of a surface to the points you want them. Instead, the process described below is best done either in a space where the bounds are already known, or in a team of two, with one person walking around the stage to test boundaries and quadrants, while the other adjusts the stage.

The program should now show a green box that represents your stage. You can adjust the corner points of the stage, as well as the centroid of the stage. To adjust the points, press one of the 1-4 numeric keys (not on the numpad) and left-click and drag. When you release the click, the corner will be dropped. The numbers 1-4 represent the four corners, starting at the back-left corner and moving counter-clockwise. To adjust the centroid, right-click on the stage

and drag to the desired location. You will see the + symbol being moved around the stage. When this is released, the + symbol is set as the centroid.

To test, have your partner walk around and verify that the quadrant is aligned with when you are walking around in the physical space.

The system is ready for you! It only supports a four-partition configuration at the moment. The projector will let you know which partition you are in by highlighting it red, and display the instrument for each partition.

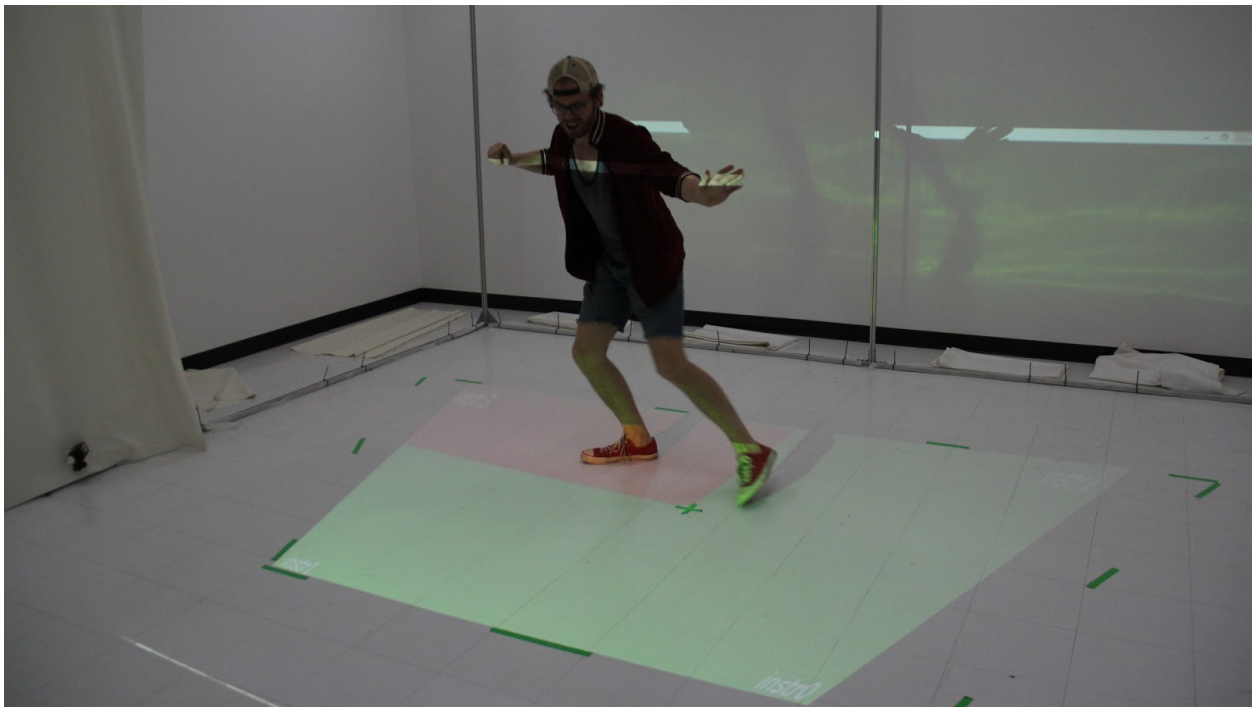

Figure 5: The stage mapped

The projection mode has both positives and negatives associated with it. For one, the tactility of the floor responding to you is extremely intuitive. However, setting up the project is not a simple task, as seen in the setup below:

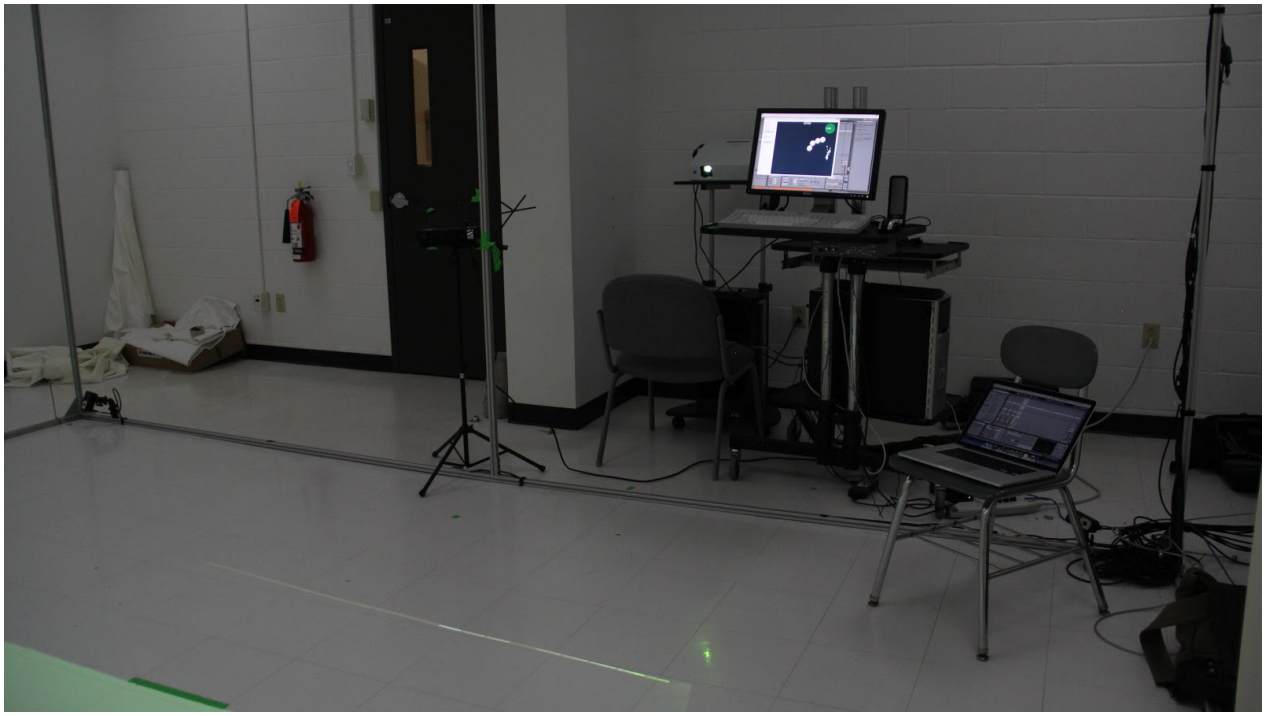

Figure 6: The setup

Overall, this is a fascinating concept and has an extreme amount of potential.

6. Results and Deliverables

In addition to the feedback provided throughout the paper, you can view the results for yourself online. The code can be viewed online at www.github.com/ahollenbach/GesturalMusic. Links to descriptive videos are included in the repository's README.md.

This document, combined with the previous report, are the most comprehensive form of documentation on the project, and walk the user through setup of the entire system. These are both available on the GitHub repository.

7. Potential Future Work

In projection mode, you'll notice that things like the text are skewed. C# provides a 3D transform class, which can be applied to 3D shapes. Unfortunately, most of my system was already written in 2D, and I was not sure how to convert it over without losing too much time.

In the future, I would like to re-implement this system to use a 3D-based system, such that the C# view transforms and 3D libraries can be leveraged. This also eliminates the need to set a centroid.

Another vision I had for projection mode is to project interactive controls on the ground. For example, loop controls would be a series of stompable buttons in front of the musician, more closely mirroring the current standard of looping pedals. This would further reduce control memorization stress.

Another goal of ours has been to develop a way to process multiple people in the instrument space. The current system supports the first body to enter, and just draws the other bodies if they are detected (but does not process them). This would really help augment demo mode, where each user could represent a different instrument. However, this is probably not practical above two or three total users, given the confinement of the Kinect stage.

Lastly, I think the codebase could use some refactoring. Unfortunately, due to time constraints and wanting to push the limits of our current system, the code has ballooned and could be reduced into much more concise, modular code.

8. Conclusion

Overall, I am very pleased with our progress on the system this semester. The demo mode opens up the accessibility of the instrument and allows even the newest of users to create music without having to worry about the complex features. The UI and UX improvements also dramatically increase accessibility. The addition of filters further develops the instrument's potential for more advanced users. The projection mode provides tactile feedback to a formerly non-tactile instrument.