Austin Dunn

260517470

MUMT 303

Final Project Report: The Basic Kinect Synth

The Basic Kinect Synth consists of a Max/MSP patch that turns movements and gestures read from an XBOX 360 Kinect and XBOX 360 Controller into synthesized music. The idea for this project came from some simple experiments I did using an XBOX 360 Controller with Max last summer and my recent growing interest in videogames. I thought it would be interesting to create an instrument that required two players to collaborate, much in the spirit of gaming. One person stands in front of the Kinect and uses their right arm to control the timbre of melodic/harmonic sounds that are triggered by the controller's X, A, and B buttons, and their left arm to change the timbre and speed of a repeating pop-like percussive sound. The person holding the controller, along with being able to trigger single notes or chords using the buttons, can use the triggers to set the decay time on the tones and to apply vibrato.

Chris Kalani's SYNAPSE for Kinect is the application that allowed me to collect data from the Kinect and send it to Max/MSP over OSC (His website is located here: http://synapsekinect.tumblr.com/). Exactly how I did that can be viewed in the [kinectRead] subpatch within my main KinectSynth patch. [udpsend] objects notify the Kinect of which joints to track. A [udpreceive] then gathers the x, y, and z values that indicate the joint positions in 3D space. These values are then routed to various [unpack] objects so that they can be extracted and scaled to modify the state of the melodic and percussive synthesizers.

The left hand in relation to the left elbow dictates the speed of a [metro] that triggers popping sounds in the percussive synthesizer, while the left elbow's height (y-value) dictates the frequency at which that pop is filtered. The right elbow's height determines the pitch of the sound produced by the melodic synthesizer, and the proximity of the right hand to the right shoulder changes the mix of a set of three oscillators of various waveforms (sinusoidal, square, and sawtooth) that emit the same frequency. The hand-to-elbow proximity is measured in 3D space, hence all the math at the bottom-right of the [kinectRead] subpatch.

The XBOX 360 controller, as a USB-HID (human interface device) friendly controller, is simple to set up with Max/MSP – although it does require a driver to work with Macs. I used the freely available driver by Tattiebogle, located at http://tattiebogle.net/index.php/ProjectRoot/Xbox360Controller/OsxDriver. The [hi] object reads inputs from HID devices. After having some trouble getting sensible results from the controller myself, I found Max user Steven Miller's patch for reading and visualizing XBOX 360 controller data on the Cycling '74 forums (post here: https://cycling74.com/forums/topic/xbox-controller-with-max/). A slightly modified version of Miller's patch is in the [controllerRead] subpatch, from which values are read and scaled for use with the synths. The left trigger is used to change delay time on sounds produced, the right trigger to apply vibrato (pressing the trigger deeper increases both the FM frequency and depth). The X, A, and B buttons are used to trigger single notes, major chords, and minor chords from the root note determined by the Kinect user's right elbow height.

The melodic synthesizer is actually a set of three [poly~] objects – one for single notes, one for major chords, and the last one for minor chords. I tried to implement a single [poly~] that would take bangs from different inlets to trigger these various events, but after seeing that that didn't work I split up the [poly~]s. The synthesizers take a root midi note value, oscillator mix number, vibrato level value, and decay time value. Values representing the third and fifth are added to the root note (in the case of the chord synthesizers) and those frequencies are sent to a set of oscillators. These oscillators are mixed according to the proximity of the Kinect user's right hand to right shoulder, applied vibrato if the right trigger is depressed, and applied an envelope of a set 20 ms attack followed by a decay time to 0 set by the left trigger.

[pops], the percussive synthesizer, simply outputs a recording of a "pop" sound created by freesound.org user unfa (http://www.freesound.org/people/unfa/sounds/245645/) repetitively at a rate set by the Kinect user's left hand position. This pop is bandpass filtered two

separate times, the resulting signals are output: once at 75 Hz, and again at a frequency between 0 and 1000 Hz set by the Kinect user's left elbow position.

The patch is capable of making some cool sounds that match the cartoon/videogame aesthetic that I was going for. However, I feel that the project may have suffered from my lack of specific ideas when I started my work, as well as my lack of experience in creating synthesizers in Max/MSP. I did achieve what I set out to, only to realize once I was finished how many aspects left room for improvement. The control parameters that go into the synthesizers, as well as the sounds that come out, aren't particularly interesting. A major improvement on the current patch would be to have a synthesizer, or perhaps even multiple synthesizers to choose from, that would take a broader set of parameters to control oscillators, LFOs, envelopes, filters, and mixers. If this was the case, perhaps the controller could select the parameters that are affected by the Kinect user in real time. Alternatively, as Ian suggested but I failed to implement, there could be a set of easily modifiable presets that could be interpolated between depending on movements. It would be interesting to take the groundwork I've completed to a dance choreographer for suggestions towards making the part of the Kinect user more intuitive and graceful. It would also be nice to set up the synth to only play pitches within a certain key decided by the users – one of the most noticeable issues with the patch is its incapability of making harmonically coherent sounds. The percussive part of the synthesizer also leaves much to be desired, both in terms of sound and how the sounds are affected by the Kinect user. I initially attempted to create a patch that would respond to sudden gestures, but was not able to find a way to do this effectively. I also imagined that having one type of percussive sound would match the sounds generated by the rest of the patch, but after hearing the final product, I must admit the necessity for a larger range of percussive options.

The Basic Kinect Synth works great doing what it is currently intended to do. I'm happy to have been able to make a synthesizer, even one that's a bit rough around the edges, using these two widely used controllers. The patch does feature a slew of issues, all aesthetic, and could be improved upon with more time and perhaps some input from people with more experience in choreography and synth design. I hope to come back to this in the future with more experience and new, solid ideas.