**Final Report**

**A.**

**Name:** Trevaughn McGregor  **ID**: 15073063

**Address**:79 E 32nd Street, apt.11 **Phone**: (347)-993-7276

Brooklyn, New York 11226 **Date**: 12/16 /2017

**B.** CISC 4900

**C. Supervisor**:Douglas Geers Address: Brooklyn College

Phone: 646-628-2699

**D.** The title of my project is **Music w/ Numbers,** built entirely using Max/MSP. My initial vision was to create a program that analyzed the emotive qualities of human speech and use that information to create a musical composition. However, upon numerous discussions with my supervisor, I was wisely forced to scale back my aspirations to an idea that was more feasible. Eventually after a series of brainstorming and revisions, I decided that my focus would be on how to utilize audio data, specifically the **pitch**, **loudness**, **brightness**, **noisiness**, and **attack** to help assist a user create a dynamic musical piece. I chose to keep my analysis within these parameters as I felt these are fundamental aspects of an audio signal.

To achieve my goal, I first built a patch (Max/MSP name for a program) that handled all my analysis. It first reads in an audio signal to a buffer that is then processed within the parameters I specified earlier, and stored in storage for later use. After collecting the data, I had to figure out what to do with them. I created a second patch, where I first created a moving range of random choices, which without much input from the user, could help create a series of dynamic randomizations of the data for interesting results.

Secondly, I mapped each parameter to different input and output ranges to control the flow of information. Thirdly, the data is sent through several audio processing effects that are controlled entirely by the user. The audio is split into 3 different signals and routed through a gate—which acts like a switch, that is also controlled by the user. Each signal is effected differently by effects processing. This decision was made to maximize the different outcomes the user could create. Lastly, for the user to export their creation, I added recording functionalities for the user to save and edit creations whenever needed. (Include building interface)

**E.** In the infancy of my program I contemplated utilizing MEAPsoft, (a program that rearranges and segments audio recordings) to help with my patch. Unfortunately, I had to abandon its implementation within the patch because of the lack of documentation available online. It would have been too demanding to learn both MEAPsoft and Max/MSP simultaneously. I eventually found a way around this issue, and found objects in Max/MSP that helped with executing my vision without compromising the quality of my patch.

**F.** My final patch is combination of an analytical patch and composition patch. I had to include both into one patch to properly build the UI. The program will be placed on a USB drive and marked **Music w/ Numbers\_v12.** In addition, there will be a Max program you must first install (offered in Windows OS and macOS) before you will be able view the patch.

Here’s a list of all the important objects that drive my program

**buffer~:** reads in an audio file

**groove~:** converts an audio file into an audio signal

**analyzer (pitch, loudness, brightness, attack) ~:** analyzes the audio signal and gathers the pitch, note attack, loudness, and brightness. These values are shown as floats.

**scale~:** maps an input range of float or integer values to an output range

**gate~:** route a signal to one of several outlets

**sig~:** convert regular numbers into audio signals.

**ezdac~:** Digital-To-Analog-Converter through which you will route all signals

**coll~:** stores a collection of data

**clocker~:** a metronome that reports the time elapsed

**random~:** outputs random numbers within the range between 0 and 1 less than the argument specified**.  
pitchshift~:** perform pitch shifting on an input signal

**retune~:** perform pitch detection and pitch shifting

**delay~:** delay a signal

**reverb~:** produces an echo effect

**filtergraph~:** generate filter coefficients

**freqshift~:** a time-domain frequency shifter

**HOW TO USE**

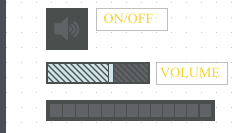
**First you will need to read in an audio file into the buffer. Click on "INSERT" and wait for waveform to appear.**

****

**Then you will go to the PLAYBACK tab and press “PLAY” to begin analysis.**



**In the event, you are not hearing sound, check the SETTINGS tab and make sure your audio button is enabled and volume is turned up**



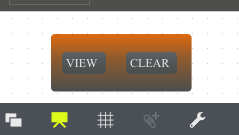
**You could view your audio being analyzed in real time in the analyzer window**



*analyzer window*

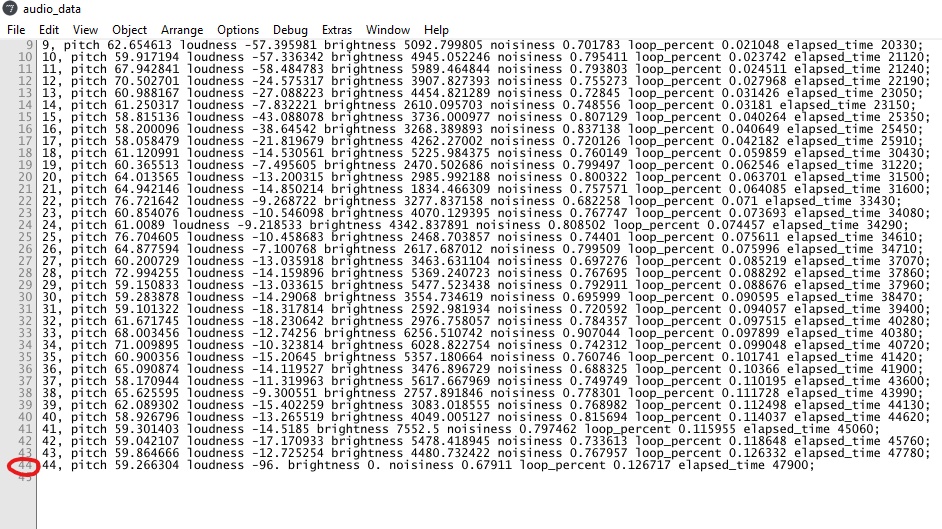
**Once you’re finish gathering enough data go back to the PLAYBACK tab and click “PAUSE” to stop audio file.**

**You can now view or clear your data that was being stored in the coll object.**



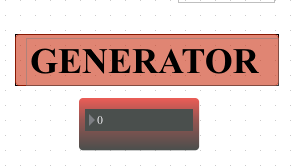
***coll object***

**Click on “VIEW” and scroll down to the end of file. On the far-left column make note of the last number. This number represents the number of data collected**



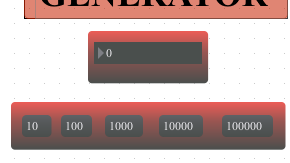
**This number is important for later when you decide to generate a new audio file**

**Proceed to the GENERATOR tab, where you’ll find the number object. Enter the number you’ve collected from the coll object**



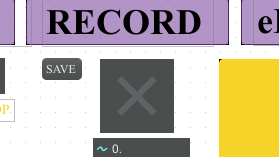
*number object*

**Below the number object you find a bar representing intervals (expressed in milliseconds) between events in your collection. This bar could be used creatively or as a shorthand way of sending numbers to the number object**



*interval bar*

Proceed to the RECORD tab where you will click on "SAVE" to save your audio to a disk. Next click the record button to start/stop the recording process. You can use the short keys 'R'-record and 'S'-Stop to make this process faster.



*record*

**Now proceed back to the GENERATOR tab and turn on the randomizer**



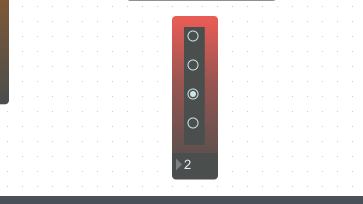
*randomizer*

**Your sound should now be moving through randomize ranges in the audio signal**

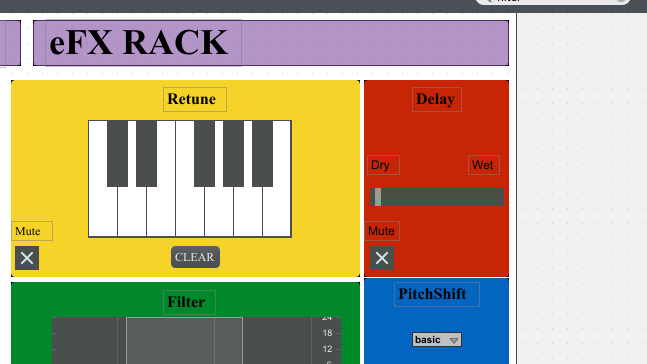
**To make things interesting try the gate object under the GENERATOR tab to output 3 different signals of your audio to various effects processing located in the eFX RACK.**

**The options for the gate object are as follows:**

**0 - no output 1 - output one 2 – output two 3 - output three**



*gate object*



*eFX RACK*

**You can manipulate the different parameters of the audio effects and see what interesting results you come up with. If you do not want an effect on your sound now just click the mute button and the effect will be bypassed. (NOTE: all effects by default are set to mute, so to add processing to your sound, you must first unmute your effect)**

**Once you’ve decided that you’ve created enough go back to the RECORD tab to stop the recording.**

**There you have it! You have just created a dynamic composition, which you can further manipulate in third party programs.**

**Credits**

*Samuel Pearce-Davies \*delay and reverb object\*. You sir are an absolute genius.*

*The numerous Max forums and YouTube videos that provided an equal amount of inspiration and frustration*

*Special thanks to Professor Douglas Geers, where without him none of this would be possible.*

**H. Final Thoughts**

I had a mostly enjoyable experience working on this project. I gained a lot of intangible and technical experience that could be applied to future endeavors. I left with a better understanding and appreciation of Max/MSP that when I started. At times the tasks seem daunting, but I learn how to properly maximize my time in deciding what is important to the success of the project. My critiques are as followed.

More internship options listed in the future: I would have loved to gain real work experience in the industry, especially with this being my last semester.

Professors listed were unresponsive: I emailed most of the professors listed as Supervisors and all were unresponsive. Professors who assigned their name to being supervisors should be more available