Spring 2020

The Impact Of Live Coding Within An Educational and Performance Setting

Alexus Renee Foster
Bard College

The Impact Of Live Coding Within An Educational and Performance Setting

A Senior Project submitted to the
Divisions of Arts and Science, Mathematics and Computing
of
Bard College

by Alexus Foster

May 3, 2020

Abstract

For the past three semesters at Bard, live coding has become my newest form of creative expression and performance. This method of coding involves either creating on the spot from scratch or editing pre-existing code in a real time manner. There is no real set structure or steps that must be followed, however, one rule must always be followed: show your code. In the classroom, live coding occurs when the professor demonstrates some algorithm by displaying the program on a projector for the entire class to observe. During musical performances, the same improvisational nature of live coding is also present, and the methods by which programmers use to practice live-coding are similar to that of musicians. In my two final concerts at Bard, I performed original music and live coded. Primarily using the program MaxMSP, I was able to create audio reactive visuals and vocal talkbox effect projects. The two concerts featured completely different setups; while one was in- person, the other was live streamed. In this paper, we aim to explore the impact of live-coding in both environments, educational and performance based.

Acknowledgements

I want to start by thanking my beautiful mother for instilling in me what it means to be a strong, independent, black woman. Thank you for your kindness, for being gentle with me, for showing me the right way. Without you, I would not be the woman I am today. I would also like to thank all of my advisors on my board: Keith O'Hara, John Esposito, Pamela Pentony, Sven Anderson, Thurman Barker, and Whitney Slaten. To Keith, thank you for your support in all of my years at Bard, and for your patience. To John, thank you for your sense of humor and encouraging words, and for helping me not be shy to take a solo. To Pam, my dear vocal teacher, thank you for opening up my voice and for showing me that it's ok to be nervous about performing, and to always remember: when it's genuine, it shows. Thank you Sven for all those times in office hours, when I didn't get the material you always showed me the way. Thank you Thurman for your wisdom and for always supporting me. Thank you Whitney for teaching me that there are limitless styles of music to enjoy, and for your empowering words. Thank you also to Matthew Sargent for showing me the beauty that is MaxMSP, and thank you Amber Billey, who taught me about databases. I hope you see this message!

Finally I want to thank my closest and best friends, Omar Ramirez, A'Zeyonna Hasty, Talaya Robinson-Dancy, Anabel Briu, Edwar Aviles-Mercedes, Adam Fallah, and Andrea Otey. You all truly made my life at Bard a million times better and I consider you all my family.

Table of Contents

Artist Statement

I am a musician, songwriter, and live coder from Atlanta, Georgia. As an aspiring singer-songwriter and live-coder, I produce art that illustrates the beauty of passion and love. I sang in the church choir from a young age, and thus my music features roots in RnB and Gospel, with the occasional experimental sound. The lyrical content of my music has heavy Motown and RnB influences, with a focus on having a conversation with the subject of the song. I released my first and only online project back in summer 2016, and plan on releasing more works this summer. Vocally, I cite Anita Baker as one of my vocal role-models, and fondly remember the emotions that Ms. Baker's music gave me when I was younger. Before diving head-first into the world of live-coding, I only improvised during a solo; now, I also improvise while programming. My performances at Bard were collections of both music I had studied in my classes and music I created on my own. In my final two concerts, I branched out and began to include live-coding as a part of the show. I love to travel, and want to explore the world, visiting other countries and neighborhoods to observe and play music with other musicians of the world. I believe strongly in the power of empathy, and my mission is to reach individuals from all walks of life through my art and vulnerability, and to live life unapologetically.

My concerts would usually open with a gospel piece as an ode to my roots and as a musical selection to get my mind and heart in the right place. For my first senior concert, I opened the show with a gospel song, "Calvary" by Richard Smallwood. I had 6 other original songs named I'm So In Love With You, Deeper Meaning, I'd Give My All To You, OOP, Eternal Mood, and Trashpop. Each song held it's own theme, from inner conflict, to the intense feelings that loving someone brings, to letting people know to stay in their lane. OOP in

particular stood out from the rest of the other songs because I played this song in the background while I live-coded. Live-coding during my performance was exhilarating and nerve-wrecking, as there is a big moment of silence while you write the code and run it. For my last concert, I chose to perform one of the most beautiful Anita Baker songs, “Just Because”, which speaks of love with no expectations or rules. This song makes me emotional and her delivery of the lyrics is a skill that I am working towards. My other four songs for this final concert are named Clear Intentions, Show Me, Dreams, and Body Shimmer and are all original compositions. This second concert was live streamed due to the global pandemic, and some of these songs had themes either musically or lyrically surrounding isolation and longing for human interaction.

I am proud of how my final concert turned out, even if it was performed remotely, because despite the circumstances I was still able to share my work with my friends and family. After Bard, I look forward to branching out and working with other artists, specifically in writing songs or producing. If my music can uplift at least one person, then I know that I have done what I’ve set out to accomplish.

Chapter 1: Introduction and Background

Live-Coding is the practice of writing, debugging, compiling, and running code in a real time manner. “Live Coding has established itself as an alternative to traditional laptop performances. Offering a form of improvisation at an algorithmic level, it gives an insight into the used algorithms to the audience by making the source code visible, while at the same time focusing on the (physical) presence of the performers.” (Zmölnig and Eckel, 2007) Programmers live-code for a variety of reasons ranging from educational opportunities to creative endeavors. In this practice, there are many parallels to improv in music, and being both a programmer and a musician, these similarities were striking. In my project, I explore both of these worlds, embracing the challenges. These challenges come in the form of learning the language inside and out, so that live coding on stage is autonomous, or letting the creative side of your work be shown, without limiting yourself as an artist and programmer. In the live-coding environment, on stage, the set-up and delivery isn’t very conventional; “While it is often not so interesting for the audience to watch pale faces illuminated by computer screens, it is at the same times often not very interesting for the performers to play their (however complex) systems and algorithms with the limited interface of keyboard, mouse and fader-boxes.” (Zmölnig and Eckel, 2007) In my artistic practice, I choose my live-coding decisions based on the story that my music is telling. For upbeat, sassy and beat driven records, there is some audio-reactive image going on in my immediate background, because I need the image to do the moving for me. For softer songs, I include calmer pictures. In the classroom, the professor writes out a working program from scratch, in demonstration, to their students. Sometimes, the students follow along on their

computers, and other times they are simply taking it in, simply observing everything that is happening. This is the Manifesto of live-coding.

When looking at coding as a process in the live-coding performance setting, there is more space and forgiveness given to the programmer. In fact, the environment is entirely experimental at times, with bugginess and errors being not only rare, but expected! Premade code, which I mentioned above, is utilized as well during live-coding performances, however there are some negative opinions towards this. Another part of the live coding process is diagramming or creating a mark-up, which is a method I employed personally! In Figure 1, notice my notes on which aspects and attributes of the code I could change. I remember figuring these scribbles out through a trial and error process, and even admittedly feeling like I was diluting the experience in a way, because I brought the notes with me on stage. Although live coding is essentially expected to be completely live, I wrote these notes down for my own performance anxieties that I sometimes experience. Ironically, I ended up not using them much during the show.

This thesis initially aimed to investigate the impact that live-coding has within different settings, such as an educational classroom or a performance setting. Within the classroom, live coding has been utilized in many different outlets. For one example, it has been used as a learning tool to accompany students who may require a more hands on approach when it comes to learning how to become a programmer. In the performance setting, live-coding can add another visual or audio aspect to performances ranging from art installments to music performances. My project was initially going to conduct a course followed by a survey, but I decided to keep it simple as a demonstration of live coding during a concert. I describe this in more detail in Chapter 4.

In future concerts, I expect to become more intentional in what I live-code. Right now everything is still very experimental, but one goal I am working towards is intention with the creativity aspect still being evident, and not sacrificing it during performance. Further in this paper, there are code snippets from two concerts I conducted that featured original music along with live-coding audio reactive visuals, and an explanation on a talkbox effect I created using Max/MSP/Jitter. These concerts were vastly different, as the first concert was in person, and the second concert was streamed via `twitch.tv`. Both experiences were challenging, however they taught me to be resourceful and to work under pressure, and both experiences also taught me that people are very open to live-coding, and even think it is exciting to watch.

Chapter 2: Related Work

The Pros and Cons of Live-Coding in Performance

Advantages of live coding during performance include: extreme flexibility, the ability to hear or see your efforts instantly, and being able to practice your improvisational skills in a new way. In a similar way that the audience and performer receives instant gratification at seeing a live-code implemented correctly, students in the classroom receive the same rewarding feeling after following a successful live-coding implementation. In the article "Live Coding In Laptop Performance", the disadvantages were also discussed, and included reasons such as: no debugging or testing, preparation for the next line of code takes too long, sometimes there may be moments where the code produces an ugly result such as a sharp sound. One other strong criticism of live coding in music is that it can sometimes be considered lazy performing or not considered real music, however this has been a criticism of computer music even before the thought of live-coding. Before the implementation of live-coding in music, there was computer music that simply featured two parts: the program, and the score. "In computer music there used to be a strong distinction between the program (the instrument) and the score (the player)... In the case of an interpreted language, this mediation can be constructed, but essentially the distinction is blurred." (Collins, McLean,Rohrhuber, Ward 2003) In this way, live-coding during performance blurs the roles of the computer and the person. The computer is no longer doing all of the work, there is now a more human engagement to performance using computers.

What is Cognitive Apprenticeship?

Cognitive Apprenticeship is "a model that focuses on making thinking visible. This model is based on the ancient model of apprenticeship where a student learns a skill from a

master by working under the master's guidance." (Raj, Halverson R., Patel, Halverson E. 2018). This model focuses on the process rather than the final product, and is a very useful model to consider when dealing with computer science students. Speaking from personal experience, the very first time I was presented with code, I wondered how did the author write a complete program in one sitting. When I first started, what I didn't know is that writing code is a process, it is not an activity that can be completed in one sitting. Live coding helps to shed light on this process by showing students a few useful strategies that programmers implement. For example, incremental coding, which is where you write only a few lines of code and compile it, as opposed to trying to write many lines, is one of the strategies that live coding allows students to witness.

In the article, "Role of Live-Coding in Learning Introductory Programming" the Cognitive Apprenticeship model is broken down into three stages: Modeling, Scaffolding, and Fading. In the Modeling stage, the students are following along with the instructor, as the instructor demonstrates, via live coding on a projector. In this stage the students are encouraged to follow along on their own laptops or they can alternatively just observe. In the Scaffolding stage, the teacher gives the students an assignment and provides a skeleton or "scaffold" of the assignment so that the students can piece together the missing elements. Lastly, the Fading stage is essentially the students working on an assignment by themselves, for example a homework assignment or a project. In this final stage, the instructor does not provide any scaffolds and leaves the process completely up to the student, who must demonstrate their mastery of the concept.

Looking At Programming As A Process

As mentioned earlier, it is important for computer science students to understand that programming is a process. In a traditional classroom, this concept may not be as clear. When students are presented with blocks of already made and debugged working code, or readings that feature this type of pre-made code, there are no disclaimers discussing the process behind this work. This can lead to feelings of frustration or inability, when a student attempts to write their own code, only for it to never work properly. With live-coding, the student is allowed to see that programming isn't linear. They see how a good computer scientist writes code, which is line by line. Debugging, which is a process of going through every line in your code and determining the cause of errors, is demonstrated not as an afterthought, but as an integral part of the programming process. One other positive is allowing students to personalize their own process by being able to see the different methods that each computer science professor utilizes.

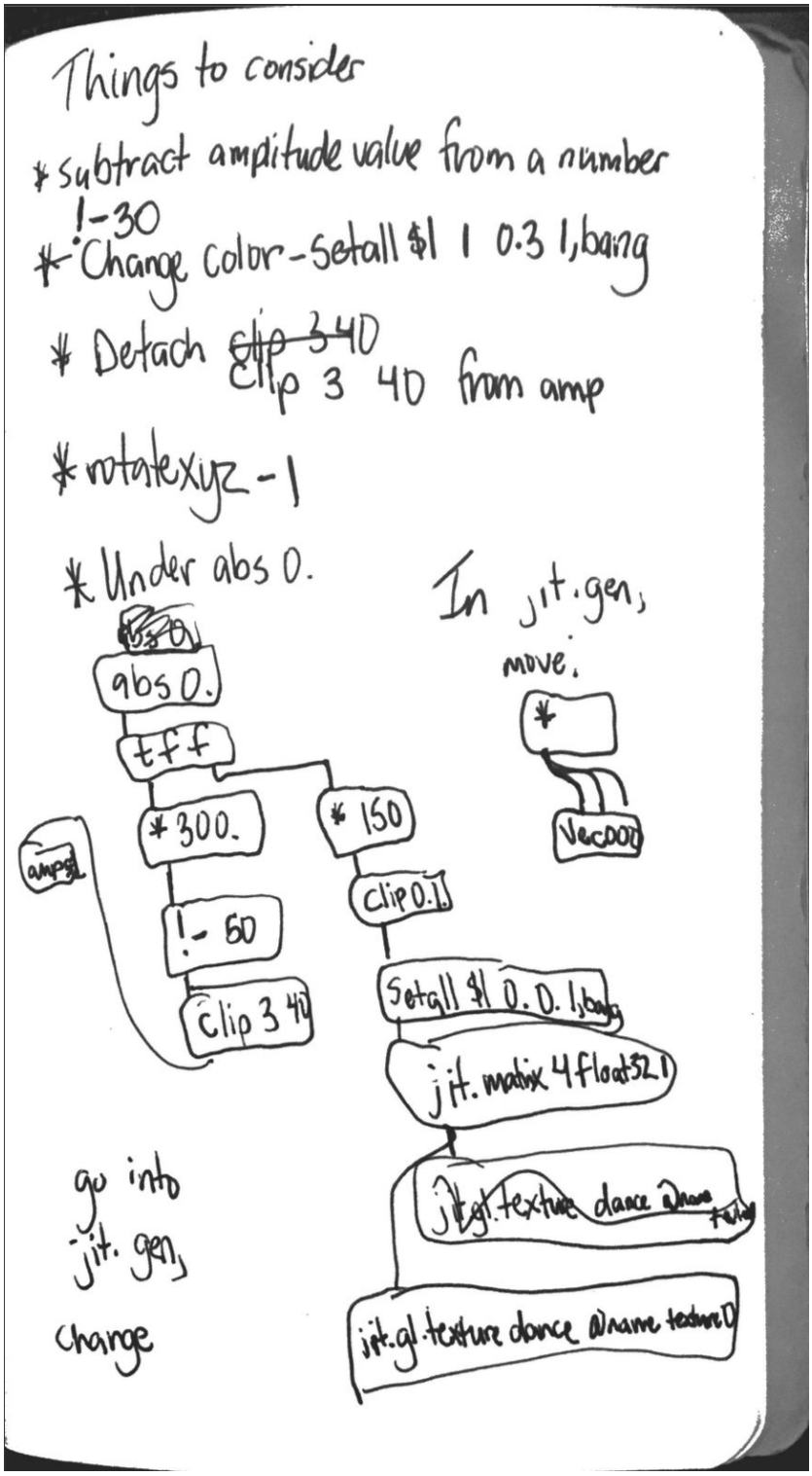

Figure 1- Alexus' Programmer's Notes

Other practices have been utilized as well, and these practices were similar to the discipline that a musician employs when preparing music! For example, isolation practices are one method that live-coders use when practicing. This technique is the practice of focusing on small parts, instead of the project as a whole. An example would be typing exercises, which work to increase the programmer's typing speed. Of the exercises I researched, I employed the typing, memory, and awareness practices. From the article, "Live Coding Practice" on memory practices- *"*Trying to keep all the processes and details in mind (especially without any graphical reminder in your live coding system) may be aided by memory practice exercises." (C. Nilson) While awareness practice works on being in the moment with the performance and even offers that "Meditation and reflection may also play a part in managing stress and concert anxiety." (C.Nilson)

Live-coding as a Collaborative Art Within The Classroom

The effects of live-coding on collaborative learning amongst students who are all learning the same concepts for the first time, is observed in using EarSketch. "EarSketch supports CSP (Computer Science Principles) through promoting creative practice and self-expression linked to computational learning; allowing for the use of computational concepts, such as layers of abstraction (e.g. the DAW timeline), loops, variables, different data types, and procedural algorithms; providing the Internet medium as both an individual and a collaborative space in which songs and scripts can be shared; and allowing for social impact of computing practices." (Xambó and Freeman 2016) In this example, students are learning from each other in many ways ranging from sharing their various music-making techniques to sharing their original

compositions. Additionally, EarSketch provided students with the chance to share their scores amongst each other and input edits as needed. The best environment, in order for live-coding to be successful is mentioned in the article as meeting these requirements. “A suitable environment for live coding in the classroom needs to allow for the exploration of real-time manipulation of code, in which both musical and computational understanding are important. Furthermore, the environment needs to be easy to use including domain-specific languages that are easy to understand.” (Aaron and Blackwell 2013)

Chapter 3: My Own Concerts

My two concerts are experiments, as I hadn't live coded prior to these performances. Though I haven't found my "formula" yet for my personal process, I still had preparation, in a much similar fashion to practicing, in order to have these sections of the concert run smoothly. Before I get into my preparation, I want to briefly discuss the two aspects within my first concerts' live-coding segments. I'll begin by saying that, for my first concert at least, half of that code was developed in the Technologies of Music class I took with Matt Sargent in Spring 2019, and the software and language that was used was Max/MSP. This code was a part of a final project we did, and I figured I'd add a visual aspect to the code, since there was already an audio aspect. The audio aspect was in the form of a vocoder that transformed my voice in realtime to have a robotic texture, while also being done in a three part harmony. This effect was inspired by one of my favorite bands Zapp and Roger, who were known for their use of the talkbox. Unfortunately, this effect wasn't utilized fully. I initially wanted to sing during one of my songs with this vocoder, however during the dress rehearsal, we discovered two things. The first was that I would need a separate microphone to capture my voice, and since the code was within my laptop, at the time I didn't know the solution/connection needed for this setup. The second was getting the actual audio to come out of the speakers, while at the same time having the mic connection that I just mentioned. Needless to say, during the concert, I ended up having a light-hearted moment and telling the story of how I lost the second half of my code. This was my way of salvaging that part of my live coding, because I felt it would be a shame to have done all of that work and not be able to show off my vocoder. The second part of my live-coding segment was an audio-visual program. Essentially this code was on display while I played an instrumental

named OOP, which was one of my original songs. This code was very basic as I am still learning shapes on MaxMSP, and I wanted to play with layers and shading so as to give the room I was in, a nightclub vibe. It somewhat worked, and what was really great, was that I was able to turn the image I loaded on an axis, giving the audience a 3d experience as well. I'm really saddened that I didn't get to incorporate different colors into the imagery, as that would've taken those visuals to the next level.

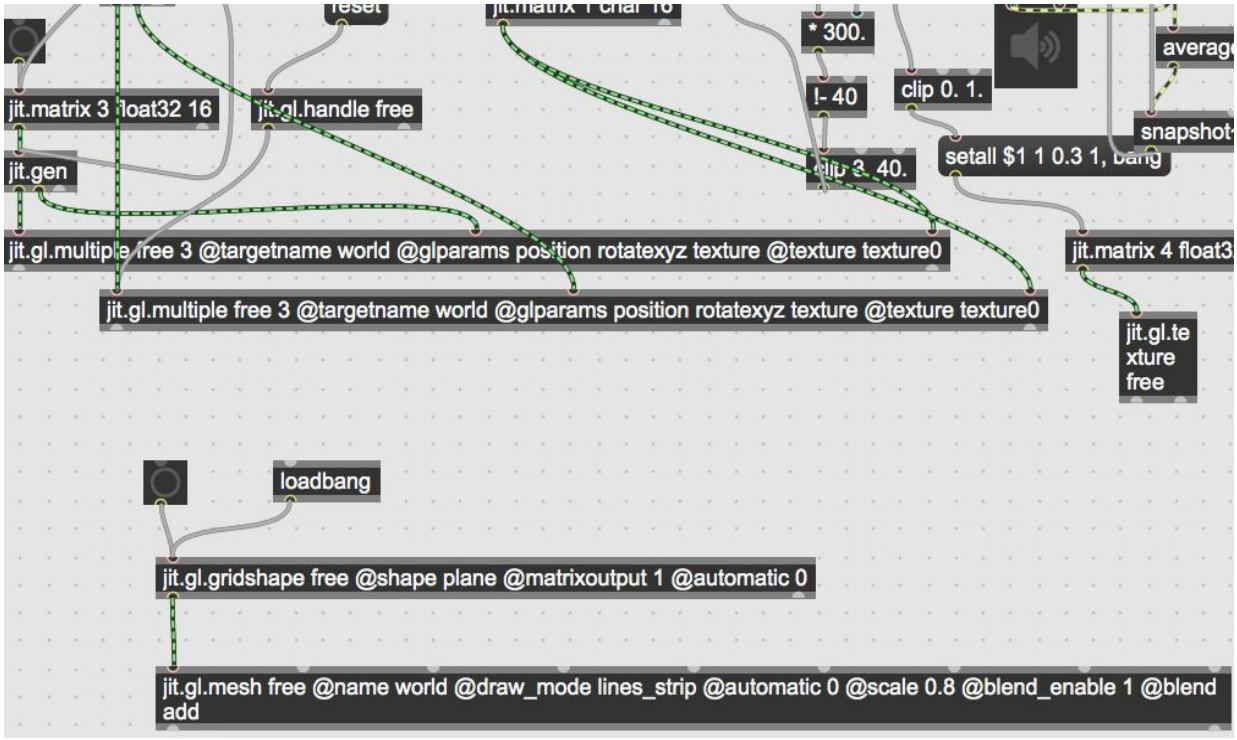

Figure 1.2- Generation of the Image begins at the `Loadbang` section

Max/MSP/Jitter

Max/MSP/Jitter is a visual programming language, which, for someone who is a visual learner, helps immensely. The program features three parts: Max is the MIDI section that handles mathematical operations, MSP deals with the audio, and Jitter offers graphical abilities, including videos. More on this software will be discussed in this chapter, as well as how I decided to utilize the software and why. In addition, this section will discuss the pros and cons of using this software. To begin, each Max project is called a Max Patch, and in contrast to the other languages, each piece of these patches are objects! "Max patching starts on a blank canvas, free from tracks, layers, or predetermined structure. This makes it natural to create interconnected processes and discover nonlinear approaches that would be hard to do elsewhere." (cycling74.com 2020) I will first discuss the interface.

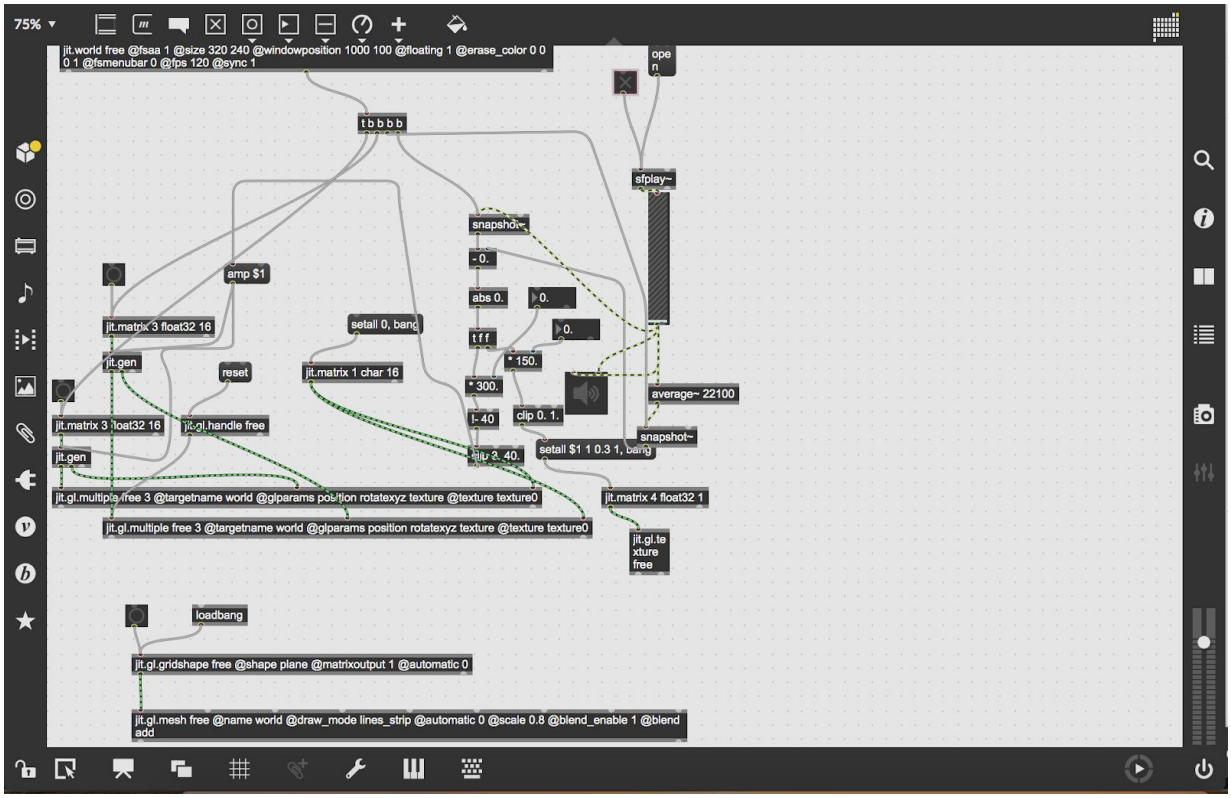

Figure 1.3- An Audio-reactive Jitter Visualization Patch, set to OFF

MaxMSP features a very clean, yet sometimes complex interface. One advantage of using this software is the visual aspect of connecting each piece of the code with another piece. These parts are called inlets and are at the two right and left corners of each object box. Depending on the type of object, the box may have both inlets at the top and outlets on the bottom. For programmers who like to keep their code organized, this software allows the user to resize object boxes, move boxes around and organize the "hooks", which are the wires that connect the object boxes. Depending on how the user organizes the hooks, there may be an effect that occurs if everything is connected correctly, or in other words, the code works. Max will not allow incorrect connections, for example outlets to outlets, to happen and will also show the user what each inlet and outlet connection will do. This interface overall is very user friendly, especially to someone who may have prior coding experience. Another extremely useful feature of the Max interface is the help library, which is built into the actual program. Unlike Java where the programmer has to search the Java documentation online for assistance, Max has a built in help library that allows the programmer to search for what different objects can do without taking themselves out of the environment. Additionally, the help library offers copy-and-paste options of an instance of the object being used, which can provide a quick solution to the question the programmer has. To someone who has never coded before, the interface can become overwhelming, and because patches can become dense, this software may be a little difficult at first to someone programming for the first time. Jitter is another graphical interface, but we will discuss this later in the chapter as it serves its own purpose independent of the Max interface.

Max is able to support six types, and types are essentially a kind of data that will let the program know how the user is using that data. The six types in Max are: `int`, `float`, `list`,

`symbol`, `bang`, and `signal`, with the last two being Max specific types that interact with the Max audio interface. With a bit of Max background, I can now talk about my programmer's notes and my code that I used during the first concert.

Looking at Figure 1, we can now explain what these notes mean. Within the big idea of this Max Patch, in order to display the audio reactive image as efficiently as possible, I came with a big portion of it created, and changed certain attributes with the objects. In Figure 1, to "subtract the amplitude value from a number", alters the image so that the space between each square shape either increases or decreases. Change color is self explanatory, it simply would change the images' color. To detach the clip, made it possible to keep the vertical axis stationary, while the horizontal axis rotated. And `rotatexyz` is the attribute that allowed the image to rotate initially. I then illustrated diagrams, as Max is very visual, to also provide extra support. Notice how they are illustrated with the object boxes and inlets and outlets to match.

These types of notes, as well as practice, and actual performance hours all contribute to the live-coding process. As a beginner and reflecting back on these concerts, I recognize now that it is ok to use notes while I am starting out. The goal is to eventually become automatic and intentional with the software, and with consistent practice, much like a musician, this can be achieved.

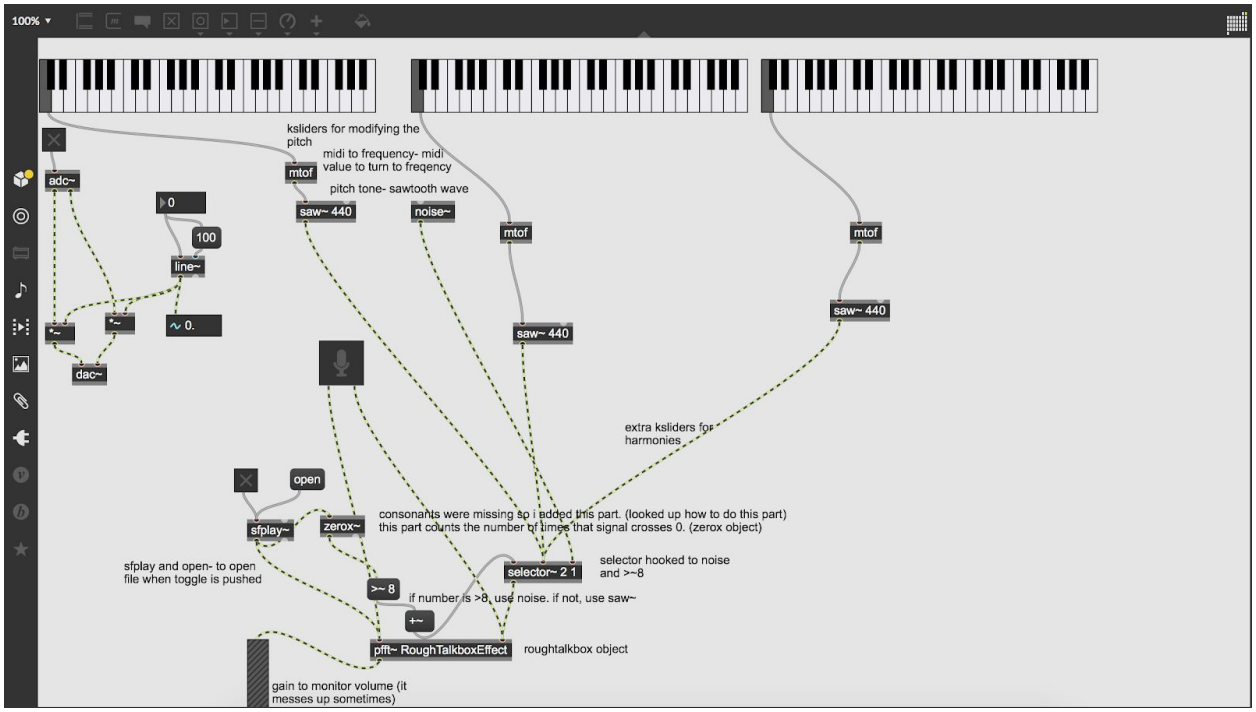

Figure 2 Max Vocoder, currently set to OFF

In Figure 2, I made a vocal talkbox as my final for Technologies in Music, inspired by my favorite band Zapp and Roger. This talkbox has several features including harmony capabilities with an option of making an infinite amount of harmonies by adding more `kslider` objects, manually changing the distance between the notes via mouse-click, and the option to load a pre-recorded song that performs in talkbox fashion.

I insisted on having the talkbox program be as simple and user friendly as possible. Also, this was one of those cases where I made the patch as visual as possible. Not only would this benefit me personally, but I also thought the audience would like the three keyboard visuals. Within the code I have three `kslider` keyboard objects all connected to a `midi-to-frequency` object, separately. `midi-to-frequency` takes midi note values and returns frequencies. From there, they each are connected to a sawtooth wave, which when

specified as saw~ produces nicer tones that don't experience aliasing, which makes tones sound cluttered and unintelligible. From here I connected the three objects to a `selector` object, which just takes any number of inputs. The most crucial and challenging part of this code was figuring out why my talkbox would unpredictably buzz out, and when I applied a visual oscillator(not pictured) to the selector object, I noticed that my tone signals would sometimes go above the x-axis, and that this was causing the buzz. Adding the `zerox` object helped solve this issue, and finally the last step was to add audio playback buttons. This is done with the `dac` and `adc` objects and these two objects send audio signals to each other. The `adc` object receives the audio, in this case my voice, and the `dac` object sends this audio out to whatever output is selected on the computer. When I used this patch for my concert, I alternated the actual keynotes on the keyboards to allow for harmonies to be made while I spoke into the talkbox.

Figure 2.1- Creation of the world; the window that displayed the Jitter Visualization

In Figure 2.1, I used Jitter which is a part of Max that deals with building graphical effects and editing movie clips. It allows the user to create a "world" where the user loads in pixels to create graphic animations. For my concert I made an audio reactive patch that was very simplified and featured square shapes that would dance to the rhythm of the song. For this code I had to create a separate window, called a world, that took the arguments `fsaa` (full screen anti-aliasing), `size`, `window position`, `floating`, `erase_color`, `fs menu bar`, `fps` (frames per second), and `sync`. This created the actual pop-up window that would hold the audio

reactive image. For the image, it held the attributes name, `draw_mode`, `automatic`, `scale`, `blend_enable`, and `blend`. This is the mesh effect's shapes and attributes. These two were the biggest builds of the patch, and to get the image to react to the audio, I connected them to a trigger object, named `tbbbb` that was then connected to a `snapshot` object. When the `snapshot` object receives a `bang`, or signal, it returns the value as a float message. Also note the float objects being received by the jitter image. From there the `snapshot` and trigger objects were connected to the `sfplay` object, which simply plays an mp3 file, and once a signal was received (here I refer to a signal as a rhythm), those float messages were returned and the image would move to the beat. From here it was possible to change the colors, but I did not figure out how to do this, and decided to do this effect in my final concert. One other effect that I enabled was the ability to rotate the image along an axis, and that is the axis with the `rotatexyz` attributes.

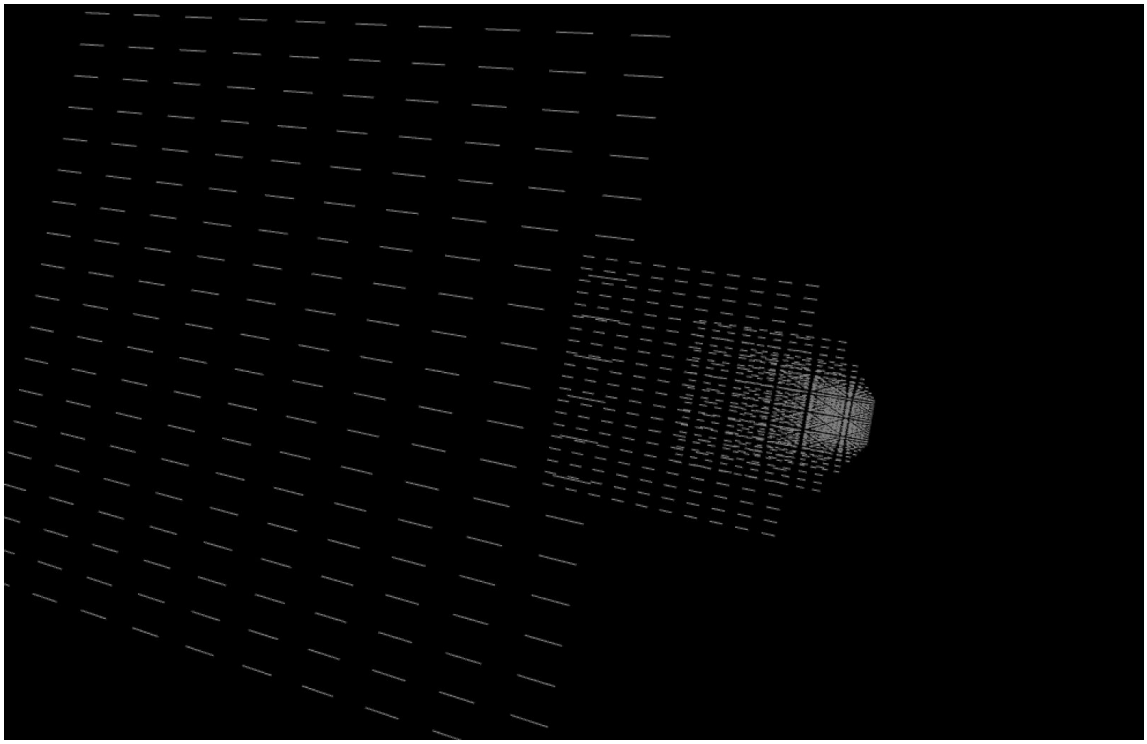

Figure 2.2- A side view of the Jitter Visualization

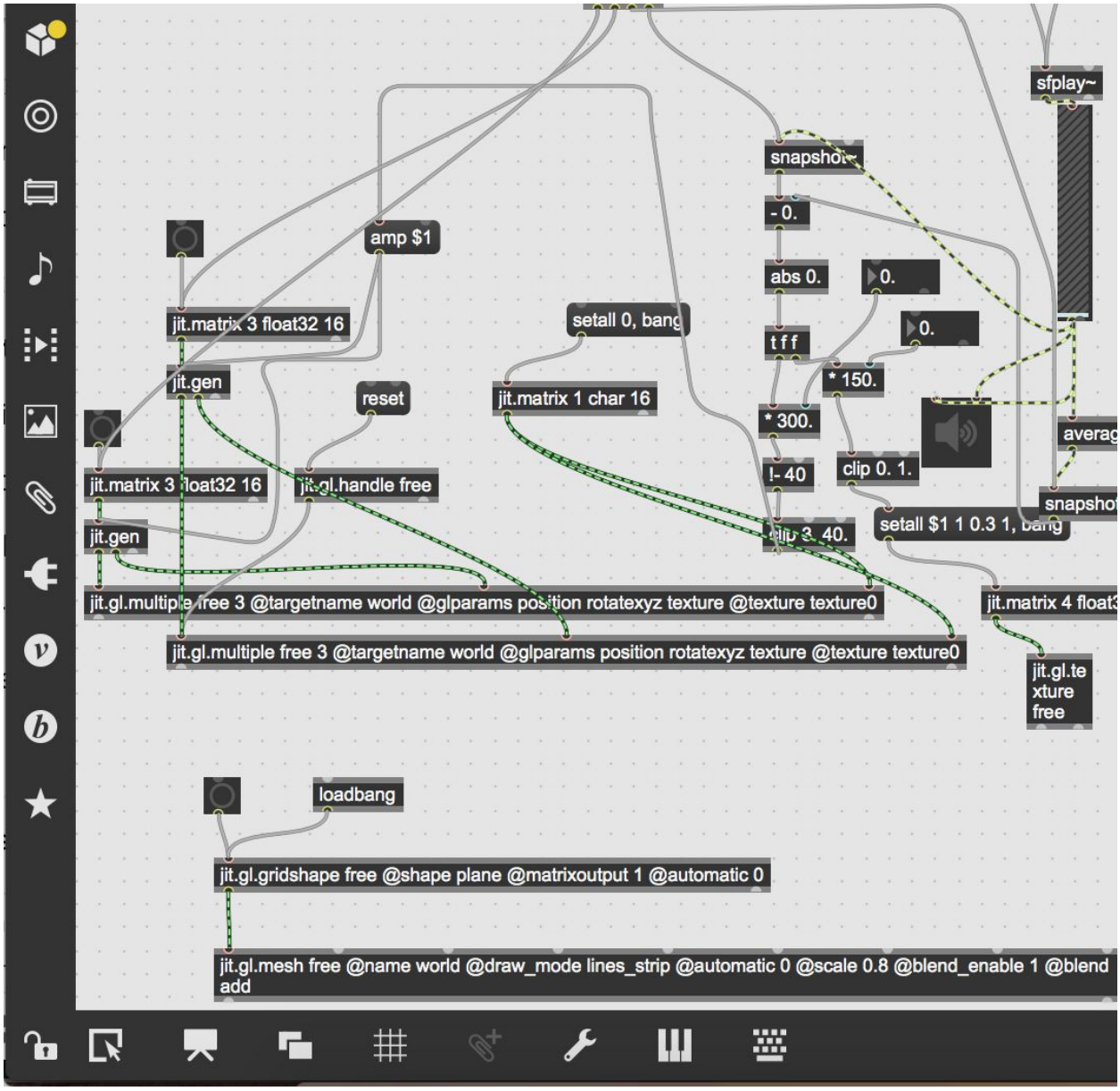

Figure 2.3- The section of code that generates Figure 2.2

Second Concert

My second concert will also incorporate visual aspects and audio reactive materials. I will be opening the concert with this visual camera capture software, named shaderbooth, that

incorporates live coding. In it you can choose several different visual “filters” and live code your alterations into your capture feed, giving the audience funny and unusual effects. My set-up was a bit tricky to figure out as an unexpected and large change occurred this semester. With school becoming remote and virtual, I had to find an alternative set-up in order to complete my concert in a way that I would still be able to give an actual concert. I went with an alternate option to present my concert via the live streaming service Twitch, and due to this fact, I needed to make sure the audio set-up was sufficient enough so that everyone could hear what I was saying and doing. The audio set-up couldn’t be the norm; the norm being the instruments and computer connected to an amp etc. Instead I settled on connecting a mic and my keyboard to my computer via an audio interface. Additionally, instead of utilizing my mac camera, I was able to borrow a computer webcam from my advisor. This concert was also obviously much shorter in length, given the fact that there were no longer musicians left on campus and that the buildings had all been closed for the rest of the semester, leaving limited resources. At the time of writing this paper, the concert has not happened yet, but I am looking forward to it!

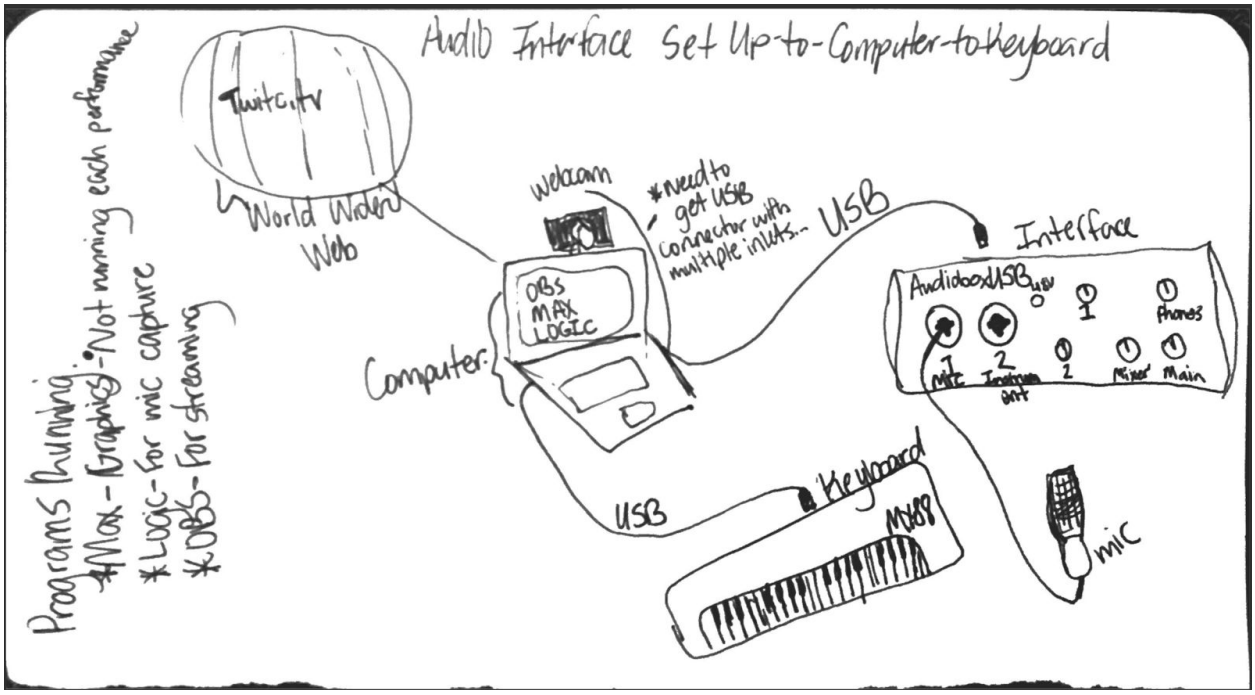

Figure 2.4- The Second Concert Set-up

A little more on my set-up, and a comparison of my set-up from the first concert to the last one. My final concert incorporated the use of video capture software named OBS, so that I may capture the video and audio feed in a real time manner. OBS or Open Broadcast System is a "free and open source software for video recording and live streaming that offers high performance real time video/audio capturing and mixing."(obsproject.com 2020) Open Broadcast System not only features streaming capabilities, but it also features recording capabilities as well, which is integral during this time of remote interaction. The audio setup features the use of an audio interface, which is a small box that holds multiple inputs for musical instruments and mics. This box is then plugged into the computer, and from there I select my input and outputs as needed. The mic I'm using is a condenser mic, Audio Technica at2020, which means it requires phantom power on the interface in order to operate. It will work with the help of the audio interface and Logic, which is a music making software. The audio interface is

one separate connection via usb. The other connection is my keyboard, which is a Yamaha MX88, and this is plugged in through a usb as well. Although my setup in this final concert is much more succinct, the technicalities and connection are more confusing. In my first concert, the setup was pretty standard for a concert: instruments and mics ran into amps, and my computer ran into a projector to be displayed to the audience. With my final concert set-up, even the idea of streaming music to a live audience from just my room seemed daunting, and this is the biggest difference personally for me. With a few concerts like this, I could see myself getting used to regularly hosting these types of concerts.

Chapter 4: Next Steps

For a large part of my time at Bard, I adamantly chose to keep my two majors completely separate. In my mind, music was my creative outlet and computer science was another outlet that I knew would support me. CompSci also provided me with a daily challenge that allowed me to set coding specific goals for myself each semester. I even moderated into my majors separately! It wasn't until my junior year when I took Technologies Of Music with Matt Sargent that I saw how I could possibly combine both music and computer science. In this class we used a program called MaxMSP that features endless possibilities, from vocal effects like delay or talkboxes, to musical effects. Since taking the class, I've also found that Max can be used to do almost anything, including visual graphics and video playback! I remember speaking about coding during my final concerts to my CS advisor and I thought they would be apprehensive, but they were the opposite; they were so excited! My advisor and I then began to speak about how coding is done in a live setting, and from that day on I began to plan for my project. I took Matt's class initially to have an elective that was in two subjects that I loved, but I didn't think it would play such a major part in my project. I made sure to design and utilize my final project for this class in a way that I would be able to use some of the code in my concert. As is to be expected, my project changed from its initial aim, and it changed even further when a global pandemic hit in my final semester of undergrad.

I initially planned to design a CS based lesson where I would teach a course to a group of non CS majors and a group of CS majors. These courses would be taught using the "traditional" method, and a method that employed live coding. The traditional method included sending students home with reading material that would explain the assignment. The live coding method

was an in class live demo that taught the same assignment, but it was more hands on. The aim was to take survey responses from these two groups of students to determine which method of delivering the lesson was more beneficial to the students. I hypothesized that the students would prefer the live-coding method as it presented students with a visual and hands-on real time experience. In contrast to the traditional method, I felt the live-coding method would appeal to more students and a more diverse learning style. This experiment wasn't carried out however because I found a substantial response from simply having a concert that featured live-coding.

For my second concert, I was interested in providing a survey or experiment. In one option I run an experiment in which I conduct a lesson using live-coding vs. traditional learning, and ask students which method they preferred. Alternatively, a survey given before and after my concert could also provide me with insight on how live-coding has impacted the general opinion of coding or my project as a whole. But as mentioned earlier, this didn't occur, and instead I gave an in bedroom streamed performance.

Now, the next steps I am looking at, is taking live-coding and music as a serious hobby that I will want to eventually make a living out of. Using live coding as a means of creativity, especially in the cross-section of the musical performance realm, is still a relatively new avenue. As I mentioned earlier, the pandemic hit and with an entire society reduced to their living rooms, it really showed the power and importance of remote performance. Concerts streaming on Instagram live and Facebook live have become the norm as of the creation of this paper, and I believe that we are witnessing the beginnings of true live stream performance.

As for myself, I want to continue live-coding and performing music. While at Bard, I was blessed with the opportunity to travel to Haiti and teach CS classes while I was over there. They

love Computer Science down in Haiti, and I was able to meet kids and young adults who truly valued education and creativity! I will be returning to Haiti to work with the students there again and show them some live-coding, as I think this is something they would like. In the very distant future, I will open a music school for children who cannot afford normal priced music lessons. This school would offer reduced or totally free lessons, as long as they maintained an excellent GPA.

Bibliography

Zmölnig, I., Eckel, G. (2007). Live Coding: An Overview

TOPLAP Manifesto. http://toplap.org/wiki/ManifestoDraft.Accessed May 2020.

Nilson, C., "Live Coding Practice", Accessed December 2019

OBS Open Broadcast System. https://obsproject.com/. Accessed May 2020

Cycling '74. https://cycling74.com/products/max/. Accessed May 2020

Raj, A. G. S., Patel, J. M., Halverson, R., & Halverson, E. R. (2018). Role of Live-coding in Learning Introductory Programming.

A. Xambo, J. Freeman, B. Magerko, and P. Shah, ´ "Challenges and new directions for collaborative live coding in the classroom," 2016

N. Collins, A. McLean, J. Rohrhuber, and A. Ward, "Live coding in laptop performance," Organised sound, vol. 8, no. 03, pp. 321–330, 2003.