Neo Khan

December 16, 2021

**Computing and the Arts Thesis: Spotify Web Visualizer**

*Implementation*

When I was first designing this project, I originally planned to use Processing to create an animated visualization of music in conjunction with SuperCollider3 and Spleeter to split songs into different instrumental tracks and analyze each track. My analysis program utilized built in musical analysis features of SuperCollider3 to gather data into a package and send it to Processing via OpenSoundControl (OSC) messaging. I ran into many problems and limitations with this implementation which I will detail in the next section. After discovering the Spotify Web API via another CPAR major in the Art Senior Seminar Class, I realized that I could skip making my own music analysis program since Spotify had their own analysis program that was more developed and had more information than I could have extracted from songs given the time frame. Therefore, I decided to change directions and create a FullStack Web App using React and NodeJS to request data from Spotify regarding certain songs and create my visuals using JavaScript. I was much more familiar with JavaScript and Web Development so the development process of the second implementation was much smoother and easier than the first implementation.

*Problems*

I attempted two different implementations of this project:

1. Visualization with Processing and Music Analysis with SuperCollider3 and Spleeter
2. Web App (Visualization with React, Music Analysis through Spotify API)

With the first implementation, SuperCollider3 was a difficult program to use and learn. Even though the documentation was well-written out, the nuances and syntax of SuperCollider3 were unusual compared to many of the other more popular programming languages such as Python or JavaScript. I was in communication with Scott Petersen, a Yale CS Professor who teaches CS 431: Computer Music. I am currently taking his course and asked him for guidance on using SuperCollider3. There was also an issue with connecting SuperCollider3 with Processing since there was a limit to how much information I could send between programs via OSC messaging. With the amount of information I could send, I was only allowed to send very limited and basic information regarding frequency and amplitude of songs for approximately 20 to 30 seconds of a song. The goal of my project was to visualize a song in its entirety, so this was a major barrier to my project for which I could not figure out a solution. Because of these issues, I decided to change the direction of the project implementation.

With the second implementation, the most difficult parts of the project were connecting to the Spotify Web API and working on the visuals on the website. The Spotify Web API isn’t too detailed or explicit in how one could connect to it or use it in their programs. I had to rely on YouTube tutorials and other online resources to help me figure out how to create the backend for my project. Once that issue was solved, the next issue was to figure out how the visuals should look on the web app. My weakest subject is Art so I was very lost in how I should make the visuals. I decided to keep things simple and only create simple shapes and figures such as squares, circles, and lines to go for a more geometrical and abstract approach to the visualization. I met with my Art advisor, [Anahita Vossoughi](mailto:anahita.vossoughi@yale.edu), who told me to keep the visuals how they are now and the experimentation with the visuals could come next semester. As of now, the project has all of the technical pieces necessary with the exception of a “pause” feature. Right now, the animation cannot be paused when the music is paused. I attempted to create an internal clock and make the animations all time-based, but my internal clock was not updating accurately, causing the animations to be out of sync with the music. This is due to limitations of the Interval feature of JavaScript. Hopefully I will be able to figure out a solution soon but it is not necessary as I currently have a workaround for it.