FINAL PROJECT REPORT

My final project is a sound/music visualizer.

++ You can use the enter key to pause the music/visualizer and enter it again to resume

+++ Press SHIFT key to save any composition you want.

After the final class I got really inspired by the potentiality of visualizing sound through programming so I thought about doing something related to sound as I have always been interested in music and sound.

\*\*\*

Another push for my decision was that I was also looking at experimentation of visualizing notations or musical scores around the time I started so I got inspired to make a visualizer similar to these sound representations.

A picture containing engineering drawing

Description automatically generated

A black and white drawing of a person's face

Description automatically generated with medium confidence

Włodzimierz Kotoński, *Aela: Muzyka Elektroniczna*

(*Aela: Electronic Music*), 1970. Published 1975.

"PR-I VIII", Boguslaw Schaeffer, 1972

#1

I chose the song Long Season by fishermans as a tester my song visualizer because it has been a staple song selection for me personally due to the fascinating complexity and layers within the song. And another bonus is that there is a sense of build-up and transitions between each parts throughout its 30-minute in length makes it even more fun to experiment and notice the changes in the visuals with times.

#2

My initial approach to the project was trying to find a way to translate the song into actual notations through javascript. It was quite challenging and time-consuming to understand all the music theory and how notes works in a composition so I had to give the idea up. I however did find some amazing resources on how to calculate the notes based on a formula from this series [[1]](https://www.youtube.com/watch?v=XCVY8eVwfvI), which I will have to look into it further for more experimentation.

After that, I learned about p5js’ “getEnergy()” method in which it will get the amount of volume at a specific frequency. They can generate a data of volume at bass", "lowMid", "mid", "highMid", "treble" frequency which I found are great datas that can help me to further visualize the sound based on different frequencies. I did learn a lot from our last class on how to analyze the sound and use that data for visualization and also watch [the p5js sound series of the Coding Train](https://youtube.com/playlist?list=PLRqwX-V7Uu6aFcVjlDAkkGIixw70s7jpW) to see more ways I can do it.

#3

After figuring out how I want to analyze the piece of music, I then went on to try and visualize 3 frequencies range: “lowMid”, “ highMid” and “mid”. I do want to add more visualization for other frequencies like “treble” also some datas from “getPeaks()” and “getOctaveBands()” but due to time limitation, I will save it for further experimentation.

In order to visualize the range, I map all the results from the fft.analyze on to a set range of position and using that mapped position (x,y) to draw all the graphic based on it and also based on the value of analyzed results in real-time.

*let* sizeLowMid = map(lowMidEnergy,minlowMid,maxlowMid, height + originOffset, (height / 2) + 100);

*let* sizemidEnergy= map(midEnergy, 5 , 100, height+ originOffset , (height / 2) +200);

*let* sizehighEnergy = map( highEnergy, 5 ,maxlowMid, height+ originOffset, (height / 2)+ 200);

A picture containing chart

Description automatically generated

* The white line: Low-mid frequency
* The hatched black line: High-mid frequency
* The bubbly line: Mid frequency
* The circle + words (random translated lyrics of the song): Beat changes found

#4

Having all the data visualized, my next approach was trying to further add some element of “random” into the visualizer without disrupting the flow of the visual and the sound. After trying out with getPeaks() and simply using certain frequency point (somewhere >200) if it show I felt that the visual can get too overwhelmed with dynamic songs.

After a long session of scrolling and wandering the internet, I figure out a very resourceful git repo on Visualizing sound made 8 years ago of [Jason Sigal](https://www.jasonsigal.cc/) who is happened to be the director of Free Music Archive. One of a few things I find interesting was his method of checking for beat changes by tracking the amplitude level of the sound [[2]](https://therewasaguy.github.io/p5-music-viz/demos/01d_beat_detect_amplitude/).

How the code works is that there is a baseline threshold of what can be considered as a beat – Beat Threshold. And every time the amplitude level cross that threshold it then can be counted as a beat with another condition that it must be higher than the beatCutoff. The beatCutoff level then decay at a decay rate. The new beatCutoff is then recalculated as the maximum value between the beatThreshold and this second version of the beatCutoff. During this cycle, the beatCutoff decays itself at a rate of 0.98 (this is I think I should be look further into how to reach to this rate or whatever number is appropriated). If the the sound meet two of the main criteria to be a beat then the function onBeat is signaled.

The framesSinceLastBeat is used to add an interval or waittime before checking another available beat in the song.

function detectBeat(level) {

if (level > beatCutoff && level > beatThreshold){

onBeat();

beatCutoff = level \*1.2;

framesSinceLastBeat = 0;

} else{

checkBeat = false;

if (framesSinceLastBeat <= beatHoldFrames){

framesSinceLastBeat ++;

}

else{

beatCutoff \*= beatDecayRate;

beatCutoff = Math.max(beatCutoff, beatThreshold);

}

}

}

I did try out with different value for all variables so that it fits the best with the complexity and tempo of the song.

*var* beatHoldFrames = 18;

*var* beatThreshold = 0.15;

*var* beatCutoff = 0;

*var* beatDecayRate = 0.98;

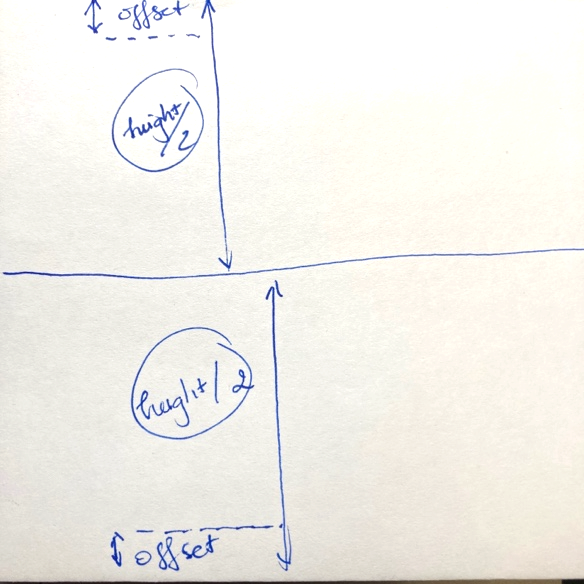
*var* framesSinceLastBeat = 0;

*var* checkBeat = false;

#5

The layout of the visualizer was the trickiest part to solve because what I wanted was to create something similar to the graphic notation. I created a single horizontal split on the screen in consideration for the complexity of the graphics but also a straightforward way to connotate a sense of structure that of graphic notation.

In order to do the split, I figured out a formula that basically subtract the offset or the bottom margin of the visualizer for the row and then use that bottom row position to detract the its height and top margin for the first row. It can be seen as a bottom-up approach to calculate the size each row.



ogX = - (height / 2) - 100;

originOffset = - 100;

*var* y = histLowMid[i] + ogX;

*var* x = i;

if(x \* 20 > width)

{

y = histLowMid[i] + originOffset;

x = (i \* 20 - width) / 20;

}

#6 Further notes

* A lot of of my inspiration comes from these visualizers [[3]](http://www.musanim.com/watch_mam.html) [[4]](http://modeleightysix.com/i-dont-wanna-go), [[5]](https://youtu.be/Xz-YnCbD_FE)
* I believe there is more to explore and solve with my program right especially trying to implement some of the structures I learned from previous weeks and also looking into ways to shorten and optimized the codes using either class object or even linked list.
* There is a lot direction that I could take on further with this project and I do thing I learned a lot more about sound visualization and the different layers of sound as data. I also think that through this project, I got to relearn how to solve small simple things within the codes from splitting the visualizer to how to separate different layers within a canvas and also how to analyze and make use of certain data for the program.

Reference

Fourier Frequency <https://www.youtube.com/watch?v=spUNpyF58BY>

The Coding Train p5j Sound Tutorial <https://youtube.com/playlist?list=PLRqwX-V7Uu6aFcVjlDAkkGIixw70s7jpW>

Music Animation Machine <http://www.musanim.com/watch_mam.html>

Audio Visualization 1600 Rainbow Balls <https://youtu.be/Xz-YnCbD_FE>

“I don’t wanna go” visualizer

<http://modeleightysix.com/i-dont-wanna-go>

Visualization art using p5.js based on the sound of waves at Yuigahama #1 of 2 <https://www.youtube.com/watch?v=mBPAtibRb1E>

Calculate Frequencies of Musical Notes with JavaScript - Part 1 <https://www.youtube.com/watch?v=XCVY8eVwfvI>

getEnergy()

<https://p5js.org/reference/#/p5.FFT/getEnergy>

P5-music-viz Repo

<https://github.com/therewasaguy/p5-music-viz/tree/master>

Sound intervals recorded by vibrating tuning forks at different frequencies. Sound and music. 1892. <https://www.are.na/block/1635343>