Music Visualization in OpenGL

CS 450: Introduction to Computer Graphics

Nicholas Skinner

[Skinnern@oregonstate.edu](mailto:Skinnern@oregonstate.edu)

Video Link: <https://media.oregonstate.edu/media/t/0_vn317y9b/>

**Project Proposal: Music Visualization in OpenGL**

For the Final project for this class, I would like to create some form of a music visualizer within the OpenGL environment. This project would consist of importing and analyzing audio files, as well as ‘distorting’ a 3d object depending on the frequencies currently playing within the audio. The inspiration comes from the musical visualizers built into windows media player, and similar applications that will display an animation of the music through reading the file.

The current concept I have for the project are: a shape, likely a sphere, in a centralized position within the viewport. This object will distort depending on the frequency and volume of the audio input, the frequency should determine the location on the object, and the volume should determine the intensity of the distortion. The actual distortion to the object should mostly look like the object is being pulled at from whatever location the frequency determines necessary.

I anticipate that because my project is focused on distortion, the object will be using textures that will be stretched around the sphere for the manipulation of the sphere. To help display the distortions, lights will shine at the object to help highlight the regions of the sphere where the distortions will happen.

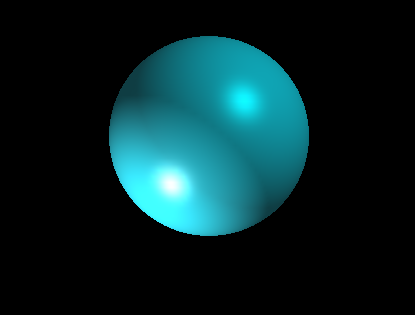
The hardest part of this concept for me sounds like it would be applying distortions to the object that are dependent on the current input of the object, this will need to be based on an algorithm and effectively will be a ‘dynamic texture’ as it will be constantly changing depending on the frequency and volume of the input audio. I have found some libraries and tools that will assist me in the project (Those tools being FMOD, and using FFT to translate the digital audio signal into a frequency domain.)

**Final Submission: Music Visualization in OpenGL**

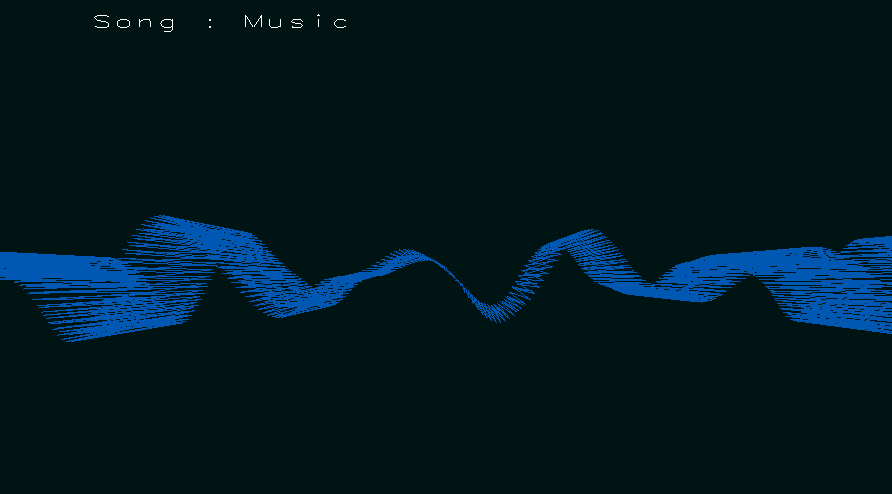
For my final project, I had aimed to make a music visualizer through the tools provided to us throughout the course. I managed to make something I’m happy with, and I thought it was a great project, as I got to decide what I wanted to do for it. I did take a significant turn in the direction I planned the go in though.

Video Link: <https://media.oregonstate.edu/media/t/0_vn317y9b/>

**Planned:**



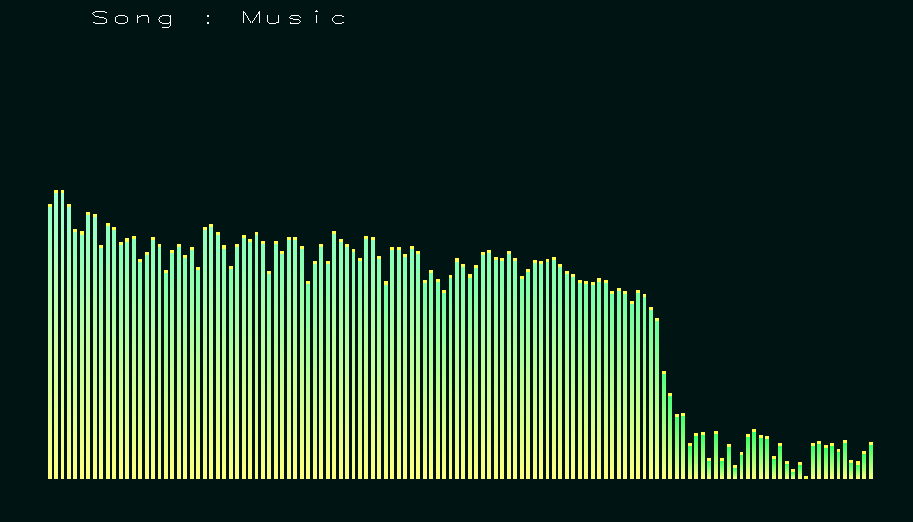
**Result:**



In my project proposal, I noted that I was interested in making a design where an object (possibly sphere) would bulge and distort depending on the volume and frequencies contained in the audio, This seemed like an easy enough concept at first, but I had some difficulties and walls very early on.

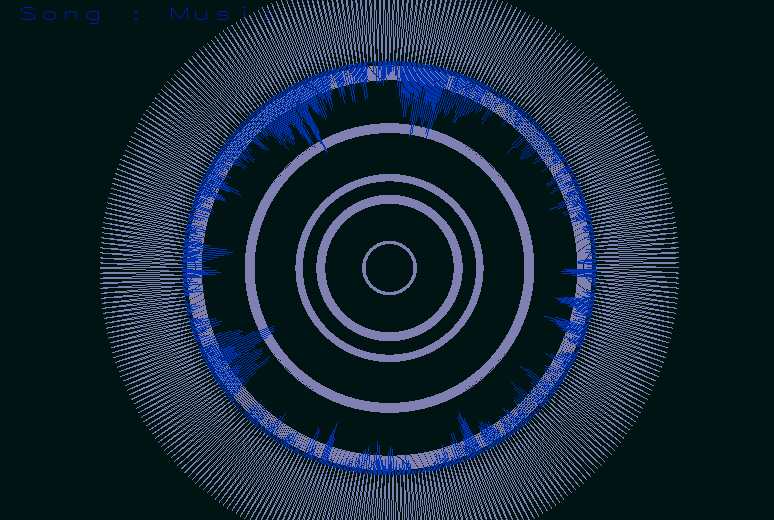
First things first, I browsed around for solutions to intake and play audio for my application, FMOD seemed like a promising tool, and it was a very good choice while I had been using it, I encountered some issues with the tool when trying to apply distortions to a sphere, and was ultimately unable to resolve the problems I was having. I looked for, and found an alternative within a utility called Portaudio. This tool worked somewhat similarly to FMOD, but I ended up with the same issue, which I determined to be user error. After both of these Issues, I made an executive decision to change my direction with the project into a general music visualizer, which would redraw vertexes to display appropriate information about the current state of the program.

To apply the Audio to the project, I had set up a digital signal processing filter using a library of fast fourier transforms, FFT is an efficient algorithm for translating signals from one domain to a frequency domain. Digital audio itself contains information about the sounds that should be played at designated time intervals, and the digital signal processing filter that was created is an approximation of the audio informations given to me at the time.



Now that I had imported data pertaining to the audio, I needed to apply it to a graphical interface. Trying and failing with the sphere, I realized that I didn’t understand what was happening where methods were not responding, or only gave null signals, the sphere was not distorting, and I didn’t know how to troubleshoot it.

After some research, I found some ways that I could go about visualizing audio in a more analytical and measurable way, I found some ideas within time domain, which is signal amplitude over time. As well as frequency domain, which would be the signal amplitude over the frequency of the given input. These were both great visualizations of the audio, and give measurable correctness to the transformations the shapes will go under. Each implementation had a few hurdles along the way, mostly limited to math graphical representations. One specific issue that I had encountered through the project was needing to refer back to FFT rules and documentation to resolve numerous issues about display, as well as audio troubles. After Implementing both systems, I found lighting to be an issue, the scene was too dark, and needed to be highlighted to display transformations. Lighting was fairly easy to implement, and fun to play with, though, I did need to refer to the handouts a number of times.



I feel like I learned a lot about OpenGL through this project, I learned where some of my limitations in understanding reside, and I wish I had a bit more time to resolve those issues. Though, through shifting to displaying items through another mean, I learned that there is no one correct way to implement a system. If I did have more time, I think I would have re-approached the sphere bulging idea and would have used shaders to try to display the distortions instead of trying to make a sphere play nice with stretching portions of itself. I am happy with what I ended up making, it is different that what I had initially planned, but I feel like its a better representation of what I understood from the class.