COMP371 - Winter 2023
Adrien Tremblay - 40108982

Project Report - 3D OpenGL Audio Visualizer

Description

The project I chose to undertake this class was to create an “Audio Spectrum Visualizer”. The application reads the data from an audio source and then generates a visualization in OpenGL. The raw data is not used to directly drive the graphic visualization. Instead, the audio data is analyzed/processed in order to extract the amplitude (loudness) of the individual frequencies (notes) that make up the audio.

Once the frequency data is extracted, there are an infinite ways to visualize this information. The typical visualization is as a series of bars, where each bar represents a range of different frequencies, and the height of the bar indicates the combined amplitude of the frequencies in the range. This type of 2D visualization is one of the three different types of visualizations I wanted to implement.

The second visualization is exactly the same as the first one but three dimensional. Instead of rendering flat bars, I wanted to render cuboids.

The third visualization was to render an imported model, and to move the vertices around depending on the frequency amplitudes.

To meet the requirement of the application being interactive, I planned to add in GUI controls to different application variables that would affect the render.

Objectives

- Audio Capture
- Audio Processing / Analysis
- 2D Visualization - Frequency Graph / Wave
- 3D Visualization - Frequency Soundscape
- 3D Model Visualization

See the Project Proposal for more detail on the objectives

Results

The final result was great. I managed to complete all five objects and all three different types of visualization. The only thing I was not able to do which I wanted was to have my application visualize audio from the computer’s line-out. It was a lot simpler to have the application read an audio file, because the entire audio recording data can be loaded in memory.

The visualization looks very cool in my opinion. It looks better than I expected. I was additionally also able to implement basic Phong shading which was not part of my intended project requirements. Additionally beside the functionality to change the number of bars rendered and switch between visualization types, I also added controls to spin the visualization, rotate the camera, and

change the spin and rotate angles. Another thing I implemented was a circular bar layout which looks very nice; Instead of positioning the bars in a straight line, they are positioned in a circle.

Discussion

Overall, the project went extremely well. I feel like my code's architecture is not the best (a lot of the code is directly in the main function of the main.cpp file). However, I wrote my code to be quite optimal performance wise. What surprised me was how hard it was to do the signal processing part. I had to spend a lot of time doing research into the theory of signal processing and playing around with the FFTW library to finally get the result I wanted. Another big time suck was playing around with different libraries. For example, my first choice of a GUI library was ImGui, but after a lot of work I was not able to get it to function with my project because of unresolved bugs with the library to integrate it with SFML.

I think the result I produced for the scope of a class project is amazing, but If I had more time I would do the following; Firstly, I would improve the code that does the signal processing to get the frequency amplitudes to give more conservative values so that the height of the bars didn't vary so much. Secondly I would add more controls in the GUI, a lot of variables are hard coded constants that can't change. I would also add an option for the user to select different audio files in the application. A final touch that would have been cool would be to add particle or lighting effects.