Audio Visualizer

Created by: Carlos Hernandez

This is the documentation for my audio visualizer 3D. This document will go into detail of what the project is about, what the goals were, what was accomplished, my struggles and what I have learned, and how to use the audio visualizer.

Setup

This project primarily utilizes FFTW3, and SFML. Just to get this project to work out of the box you will need to have SFML, FFTW3. To build this project:

```
-lfftw3 -lsfml-graphics -lsfml-system -lsfml-window -lsfml-audio
```

Usage

At the moment the usage of the program is simple. The program will automatically perform an FFT on the audio data from the file in samples of 1024. Then it will play the audio file, “excitable.wav” included with the package(kudos if you know what movie this is from). While the audio file is playing the program will keep track of the playing offset of the audio and display the correct visual representation of the audio. This is the time synchronization that caused me some issues but was handled using SFML clocks and time. The space bar pauses and plays the audio(with the audio still in sync with the visual representations).

Description:

The aim of this project was to be able to read an audio file and interpret the audio data into the frequency and magnitudes of the audio. Which would then be able to be output as a simple graphical representation. My main goals were:

1. To read in an audio file
2. Perform an FFT on the data
3. Interpret FFT data
4. Create a graphical representation of interpreted data
5. Sync the audio and graphical output of the data
6. Create a modular audio visualizer that would allow me to always be able to use this to improve upon later.

The stretch goals were:

1. Clean user interface (play, pause, forward, backward, next track, previous track)
2. Perform multiple types of data representation (normal, log, etc..)
3. Use mp3 files instead of wav files
4. Increase efficiency

The main goals were all achieved and I even put in some UI (playing and pausing the audio with the space bar) but improvements in each area can and should still be made. The visualizer still cannot play anything but wav files also the project does not go into multi channeled files. Although what the program can do it now does much better compared to the original method in 2D, so much so that I transferred the improved code over to the 2D project. This is the major difference between the 3D project and the 2D project. I rewrote the algorithm for performing an FFT and getting the data from the audio file. Now instead of trying to get in between the playing audio buffer through overloaded function calls, I now just perform an FFT on the whole data set before ever playing audio and the audio is then synchronized through clocks and time objects with the visual representation of the audio. This resulted in a crisper sound, better visualization, and much better performance compared to the 2D project. While the graphical representation is straight forward, it isn't the focus of the project. I wanted to improve upon the actual interpretation of the audio data gathered and make it a real audio visualizer. Graphics can always be improved with time but the audio data gathered is the complicated bit.

Conclusion:

The good thing about all the headaches that this project gave me was it helped me understand audio, wav files, libsdnfile, FFT's, FFTW, kFFT, SDL, and SFML-audio. This project is mainly about what is happening in the background and not so much about user interface or interaction which kinda restricts the documentation. While I could go into the performance of an FFT and how SFML audio works it isn't exactly useful in a manual on how to use the project. For all the problems I had, I am actually happy about the project and I do plan on working on it more as time goes on to make it something I can use or show off. I have always loved music and being able to see its representation how I may perceive it is something to me which is beautiful and worth the pain.