Aparnaa Senthilnathan 5/6/2025

Global Illumination

VisualDove: A CSCI 711 Final Project Report

Intro

For my Global Illumination final project, I created a music visualizer VisualDove, which incorporates a blend of computer graphics elements with audio tools such as frequencies to display visuals that accompany the selected song choice. My main motivation is to challenge myself to learn how to code and integrate music elements into computer graphics. In addition, I planned to create different music patterns and incorporate different illumination effects to diversify my music visualizer. Plus, I plan to use VisualDove as my new visualizer for my future remixes and songs that I'll produce under my music alias Illumidove

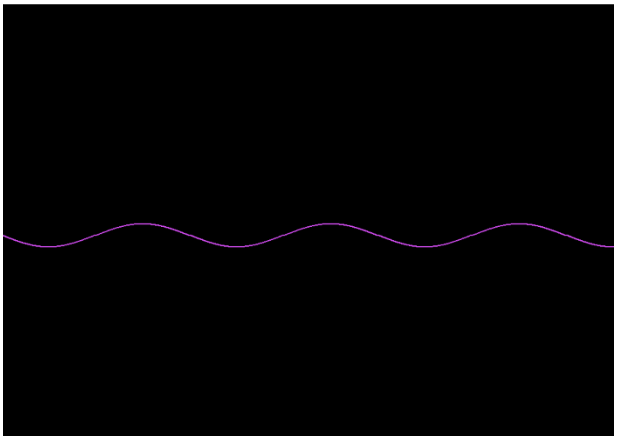
Architecture

To create the music visualizer, I used Python as my base object-oriented programming language. I used Python because it is the first programming language I learned overall, I was most familiar with it, and it contains many libraries that can help with audio. The two Python libraries I used for visualizing audio are PyGame and Librosa. While PyGame is used to code and display moving graphics, Librosa is used to handle audio tools including decibels and manage audio playback. During the project's early stages, I used PyAudio to create a simple audio visualizer using microphone input and sine waveforms to understand how audio is converted.

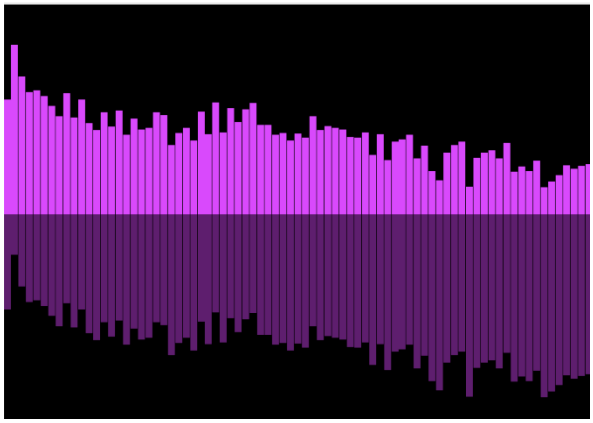
System

To create the music visualizer template, I created the AudioBar class, which uses PyGame to create and render rectangles that change height based on the noise level in decibels. To get the decibel values for each of the AudioBar instances, I used Librosa to load data from the song, which includes the sample rate. Then, I created a spectrogram using short-time fourier transform values from the aforementioned data collected with Librosa, get both times and frequencies arrays, and used them to calculate the index_ratio of both. They will then be used, alongside the given time and frequency at the moment, to get the decibel values from the spectrogram. I also created another rectangle object in a darker color compared to the regular AudioBar that is flipped through the x-axis. On the computer graphics side, I normally had plans to render the light effects on each audio bar. However, I ran into many programming challenges. I found out that it takes a while for a program to raytrace an image on Python, and I didn't know how to incorporate RGB values into each of the pixels in the rectangles drawn using PyGame. Instead, I decided to take a different approach, and create a separate file for my raytracer functions and have the program render two huge rectangles and save them onto a color grid of RGB values in bytes. There, I plan to use the color grid map for each of the rectangles given the x, y, height, and width coordinates.

Results:



The early stage of the visualizer



Current Outcome

The Ray-Traced Image to be used as the color-grid

Future Work:

Despite making some progress with my final assignment, there is still some work that I plan to do in the future. I plan on fixing my rendering for the ray-traced image because I found that the lighting and reflection aren't rendered properly. There were a couple of pixels with very different colors from the rest, making the image less consistent in colors. I had plans to implement a k-d tree to optimize my raytracer and reduce the time it takes for the program to render. With that, it'll be easier to utilize it in images with bigger resolution which will help me render the image with dimensions that fit those of the music visualizer. After fixing my rendering and my color coordinate grid, I plan to implement photon mapping to help make my music visualizer pop with light. I also plan to experiment with photons to create different wavelengths and different colors, and other polygons for my audio bars, including using a pile of spheres and particles. Finally, I plan to incorporate user functionality such as having the user choose a song from the music library folder or upload a file for the music visualizer to play. I also plan to incorporate functions for the user to play, pause, and stop the visualizer.