OpenGL Music Visualizer

CS 450: Intro to Computer Graphics

Kyler Stole

stolek@oregonstate.edu

T-Shirt
My size preferences are: L, M, XL, XXL, Any.

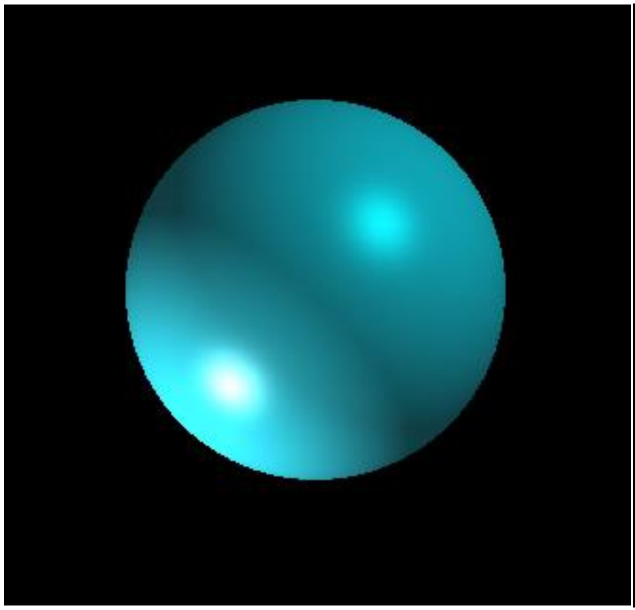
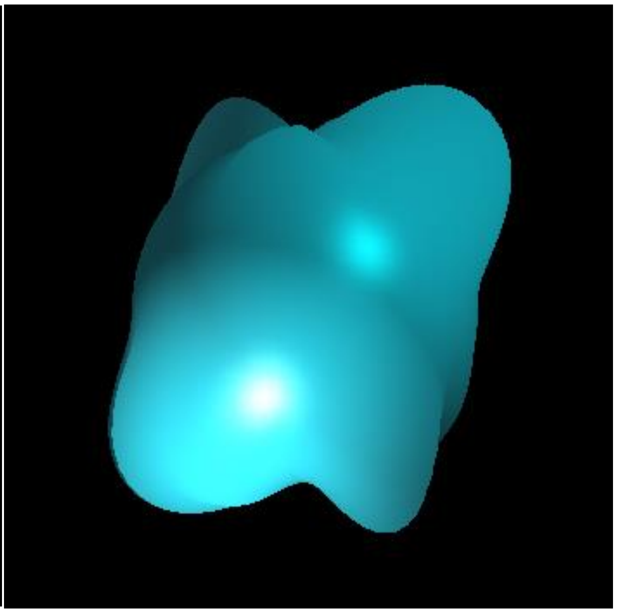
OpenGL Music Visualizer – Project Proposal

For my final project, I would like to create a 3D music visualizer using OpenGL. This will involve analyzing audio files, which I have not done before, but will also contain considerable 3D graphics.

Currently, I am envisioning a sphere somewhat centered in the scene. Based on varying frequencies, different portions of the sphere will bulge out. The volume will dictate how far the bulges go. The sphere will have a texture applied to it that slowly translates around it. Although, I currently have a sphere in mind, I may change that to some other object or make the object selectable. Regardless of the object, the same effects will be applied.

The scene will also have lighting that will move around the visualizer object. Besides the visualizer object, which will take up most of the center of the screen, there will be other elements in the scene. I would like to put some form of stage below the visualizer object, which will also move a bit with the music (somewhat reflecting the visualizer object's movements). Finally, there will also be some particles floating in the scene.

For my project, I aimed to create a 3D music visualizer in OpenGL, and I actually did it! This was a very fun project because I actually had something in mind that I wanted to do. I was able to apply a lot of creativity to get something that very closely resembled what I had originally imagined.

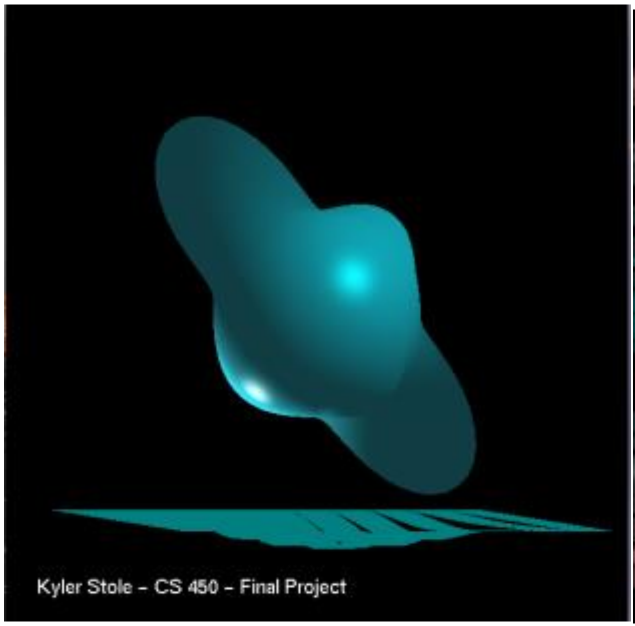
In my project proposal, I described a sphere that bulges out different sections depending on the music. This seemed like a simple concept, but getting it to fit what I was imagining was not easy. Even after I got the audio analysis part of the project working, connecting it to the sphere was a difficult journey.

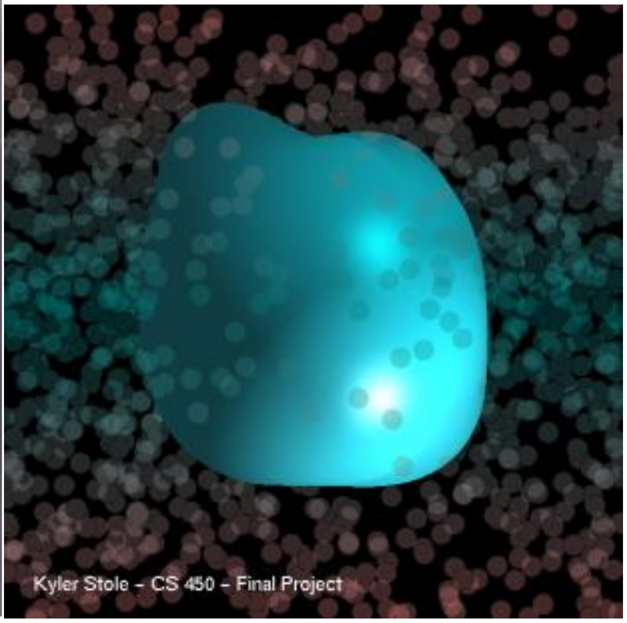
The first step in this project was the audio analysis. For this, I used a useful library called FMOD, which I found from searching about online. This library and the FMOD Studio application that can accompany it are used by game developers to add music and sound effects. I used the low level C++ API from the library. It helped me play songs and perform frequency analysis. I set up a digital signal processing (DSP) filter using a technique called fast Fourier transforms (FFT). The FFT is an efficient algorithm for translating a signal from one domain (in this case a digital audio signal) to a frequency domain. Essentially, digital audio contains information about the sound that should be played at different time intervals and what I got after the DSP filter was an approximation of the volume of each frequency at one point in time, based on averages for the last little bit of time. The frequencies are represented in discrete sample bins. In code, this is a large array of frequency bins with the volume of each discrete frequency range.

With the audio represented in discrete terms, I was able to apply it to the sphere. I used the MjbSphere function from a previous project to create the sphere. I then modified the points using the frequency spectrum data, having the radius for some points grow depending on the spectrum data. This was very choppy at first, so I had to add a step in between the frequency analysis and the sphere drawing that somewhat smoothed the spectrum by averaging ranges of frequencies. I messed around with several ways to represent the spectrum on the sphere. In the end, I split left and right speaker channels between the top and bottom hemispheres. Then, I wrapped the linear frequency data around each hemisphere, such that the lower frequencies would be in one area and rotating around the sphere would result in increasingly higher frequencies. Since the DSP acts on the whole range of frequencies in the audio, it produces a range of frequencies up to the sample rate of the song (this was 44,100 Hz for the songs I used), which is the highest measurable frequency. Typical songs, however, only used the lower frequencies. Even very sounds that we consider high pitch don't register very high on the full spectrum of possible frequencies. Due to this, I cut off most of the spectrum and only kept the lower part.

Besides the frequency analysis, the other very clever part of this project was representing the bulges on the sphere in an aesthetically-pleasing way. When I first tried hooking the sphere up with the music, it looked very strange. I applied the radius transformation to entire slices of the sphere, which bulged out equally. What I really wanted, though, was to have the center bulge out and gradually taper off near the extremes. First I tried using a Gaussian function. This helped but led to some pretty strange extremes. What I ended up doing was modeling the hump of a sine curve and then multiplying that by the amount of bulge for a particular slice based on the frequency data. This worked quite well and got me something that was very close to my original idea.

Another difficult part of the project was one of the additional pieces. In my proposal, I suggested that I wanted some particles floating around. I experimented with generating particles using the GL_POINTS topology. After a lot of work, I got the particle generator working, and the physics mostly working. The particles are created above the sphere and then bounce off the sphere when they hit it and off the stage below the sphere as well, before eventually dying. However, due to time restraints, I never quite polished the particles the way I wanted. Also, although they bounce off the sphere because they know the base radius of the sphere, they do not bounce off the extrusions on the sphere; instead, the go right through. Similarly, the stage under the sphere is also present but not quite up to par with what I imagined. Due to time constraints I had to limit it to a simple plane. It's also limited to react to only one frequency bin, so songs without a steady, low-frequency beat don't get any movement out of the stage. If I had more time, I would perform beat analysis on the spectrum data and use that to move the stage.





I learned a great deal from this project. All of the frequency analysis was new, and I appreciated learning how to break audio down into representable elements. I also learned about particle systems, and the many difficulties that come with trying to control them. Finally, I became more familiar with representing objects in OpenGL. If I had more time, I think shaders would have been a better way to implement certain aspects of the project, but I didn't think I could accomplish what I wanted to do with shaders in the time that I had.