Cornell University

CS 4621 Practicum Final Report

A♯ – Music Visualizer

Shane Moore
swm85

Zachary Zimmerman
ztz3

Emre Findik
ef343

Joseph Vinegrad
jav86

December 15, 2014

1 Summary

Our group worked on a music visualizer called A♯ (A Sharp). While typical visualizers have nice intricate displays, we feel that they do not meet the standards of a music visualizer in the true sense of the term. Our visualizer attemps to create a more accurate representation of the actual features of a song.

The visualizer models a song using a single sphere. We animate this sphere through sets of transformations based on data analysis from a song. In particular, our application analyzes features of a song, inlcluding beats and frequency amplitudes. Then, we use beats and overall amplitude to alter the sphere radius and frequency amplitudes to perform displacements on the surface of the sphere. We are also in the early stages of applying vertical translations based on changes in frequency, as well as choosing color based on the mood of a song.

2 Implementation

For our graphics pipeline, we wrote vertex and fragment shaders for our sphere to set positional and color data. After receiving data for each frame from the sound analysis, we set the radius of the sphere according to beat pulse and sound amplitude. Then, we use frequency data to compute vertical translations of the sphere. The idea behind this is to create an upward movement of the sphere when the pitch rises and a downward movement when pitch falls. We model this vertical motion of the sphere using a damped harmonic oscillator.

We also use frequency amplitude data to set displacement magnitudes. We dynamically create a texture map using this data at each frame. Essentially, the objective here is to displace higher points on the sphere according to the amplitudes of high frequencies, and lower points on the sphere according to the amplitudes of low frequencies. Finally we set shader uniforms based on the computed radius and texture map at each frame.

3 Software Design

Our codebase is divided into two main modules: (1) Sound analysis and (2) Graphics.

(1) Sound analysis takes place as a pre-processing stage, before we display any graphics. We obtain data for the sound analysis in `app/analysis.py`. We utilize an audio and music processing library called librosa to do the bulk of the work in obtaining the data. The data we receive from this includes separated harmonic and percussive sounds, a melspectrogram (frequency amplitudes), tempo, beats, frame times, and a chromagram (???). All of these data sets are compiled for each frame in the song. Finally we perform operations to condense this data in order to make use of it in our graphics module (probably want more info here or in implementation section).

(2) Once we receive the sound data from analysis.py, we pass them along to our graphics module in `app/graphics.py`. First we setup our screen and initialize constants and data structures based on the sound data. At each frame, we update features of the sphere, including radius, color, vertical positon, and displacement map. This information is passed into the shaders, `DispMapped.vert` and `DispMapped.frag`, which update the appearance of the sphere on screen.