Cornell University

CS 4621 Practicum Project Proposal

A♯ – Music Visualizer

Shane Moore
swm85

Zachary Zimmerman
ztz3

Emre Findik
ef343

Joseph Vinegrad
jav86

October 9, 2014

1 Summary

Our group proposes to create a music visualizer called A♯ (A Sharp) which goes beyond the interpretive scope of current visualization software. We find that the typical music visualization does fine at looking good alongside the music, but fails to go beyond and provide actual interpretation or insight into the song.

As Edward Tufte said in his book *The Visual Display of Quantitative Information*, "At their best, graphics are instruments for reasoning about quantitative information" (Introduction). The job of a music visualizer, therefore, is to provide the user with enough relevant and useful visual information that they may interpret, on a higher level, the characteristics of the sound which is being visualized. If possible, the visualization could be considered a summary of the song, or even a rudimentary alternative. We plan to work towards this standard in our music visualizer, A♯.

2 Software description

The software we are planning to create will output a video file, given an audio file input. The visuals in the video file will be built upon not only instantaneous properties of the audio but will also be dependent upon qualities such as tempo, rhythm progressions and the musical genre, which requires analyses of the audio as a whole or in greater parts. Each of these audio qualities will be linked to a specific visual quality in the video file, such as the color properties (hue, saturation etc.) and the output shape.

3 Application in Graphics

The graphics portion of our project will focus primarily on modeling and animation.

Given our goal of creating complex visual displays that accurately reflect different types of music, modeling will be a key component of our work. Determining what combination of objects, textures and shading models to use for different songs is an interesting challenge. Our plan is to begin with a single perfect sphere that we will transform as the song progresses. In addition to modeling the sphere, we will also keep track of some underlying mesh representing the transformed sphere. We will likely represent this mesh using an indexed triangle set, potentially with optimizations such as triangle strips and triangle fans. Since we are only using one object in our visualizer, we do not anticipate rendering to be a major challenge, giving us an opportunity to focus more on modeling.

Animation is another major area of focus for our project, as we aim to produce a dynamic moving display for our visualizer. To enhance the design and versatility of our visualizer, we will utilize different animation patterns corresponding to different types of music. Currently our basic idea is to perform transformations on the surface of a sphere with different speeds, shapes and magnitudes depending on the music. To implement this, we will apply techniques such as bump mapping along with splines to seamlessly

animate the sphere frame by frame. While we havent yet discussed animation in class, we plan to do our own research on animation topics before they are introduced in class.

4 Properties of Music for Quantification

- Song genre
- Amplitude at a given frequency
- Sound location (stereo)
- Sound quality (Hilbert scope)
- Tremolo
- Centroid, spread, skewness and kurtosis (of an amplitude envelope)
- Mel-frequency cepstrum
- Rhythm complexity
- Distinguish accompaniment from melody
- Melody: pitch, volume, position in chord

5 Software Architecture

Our code will be broken up into four modules, which each represent a specific part of the program. They are laid out as follows:

1. Data collection / statistical analysis (input $\rightarrow$ data)
    - Takes in sound file
    - Runs library calls for music analysis
    - Returns the results of all analyses in the formats of the respective libraries
2. Data formatting (data $\rightarrow$ models)
    - Takes in the raw analysis data
    - Formats it all in a sensible way
    - Returns the data in the cleaner protocol
3. Data interpretation (controller: models $\rightarrow$ views)
    - Takes in the clean data
    - Interprets it in new (i.e. unique to our project) ways
    - Returns a property-based representation of the visualization
4. Rendering (views $\rightarrow$ output)
    - Takes in the parameters of the visualization
    - Creates the scene based on these parameters given

- Animates based on the parameters with respect to time
- Returns the video

The first module will perform some primary analyzation of the sound file that's been passed to the software. This will include attempts to determine the key and tempo of the piece, variations (and base-levels) of volume, and spectral analysis for some understanding of the instrumentation being used. Data from the analysis will be stored in the next module; a series of models that will be designed for fast access and calculations necessary for our representations of the piece. A controller will act as an interface between the models and views, which will store data about the visual representation itself. A final module, the renderer, will take these software representations of the animation (the views), and produce a video to accompany the song.

We plan on using python for coding this project, with a few third-party libraries for the sound analysis module.

6 Work Distribution

Emre: Sound file analysis (1) and model for representation of sound (2)

Shane: Model design (2) and controllers (3)

Zachary: Sound data interpretation (3) and visualization design (4)

Joey: Graphical representation of the view and rendering (4)

7 Milestone

By the milestone, our group hopes to have the following as part of a proof of concept:

- Warmup: midi support, later on, wav support
- Working interfaces for each of the sound libraries we choose to use
- Integration of data from all of the libraries into one format
- A rudimentary visual representing every meaningful piece of insight we can glean from the music