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

A♯ (A Sharp) is a music visualizer designed for meaningful sound information conveyance. While other visualizers have flashy and intricate animations that pleasantly accompany the music, they do not successfully convey, interperet, or even begin to replace the music. This is an enormous shortcoming of music visualizers — current applications certainly do not meet the standards of a music visualizer in the true sense of the term. A♯ attemps to make strides toward filling that gap, with hopes that other visualizers may follow suit.

A♯ models a song using a single sphere mesh. The sphere is animated through sets of transformations based on comprehensive data analysis of the song's sound file. The application analyzes several important features of a song, including the overall key, beat event times, and the frequency amplitude spectrum at each instant of the song. The results of the song analysis determines the appearance of the sphere during the playback animation.

The source code for this project, as well as any extensions to it, can be found at `https://github.com/Oneman2feet/a-sharp/`.

# 2 Conceptual Mapping

In order to achieve an intuitive representation of the chosen song, the conceptual mapping of A♯'s visuals was designed with great care, as follows:

- Overall volume determines sphere size
- Beats are shown as pulses in sphere size
- Overall pitch controlls sphere elevation (higher up means higher pitch)
- Sound complexity (distribution of frequencies) determines sphere distortion
- Mood is conveyed by the color of the sphere

## 2.1 Sound Complexity

Arguably the most important component in song identification is the melodic contour (i.e. the shape of the melody). Rather than abstracting away the frequencies in a given musical piece, we found that the simplest and most relevant way to display melodic contour was to display the frequency distribution directly.

This was accomplished by mapping each latitudinal band of vertices on the sphere mesh to a "frequency bin", or a group of similar frequencies. Each band is then distended from the sphere based on the power (log amplitude) of its resective bin at that time frame. This gives the desired illusion of the bands following the melodic contour of the song as it's being played, allowing individual melodies to be picked out by the user.

### 2.2 Mood

The decision to correlate the mood of a song with the object's color is partly inspired by condition known as synesthesia, in which signals from a particular sense (smell, sight etc.) invoke perceptions in another sense. It has also been found that those without synesthesia may associate sounds with colors,[4] though they would not directly perceive them.

The greatest challenge is procedurally quantifying the mood and assigning it a color, for which there is no clear answer. Attempting to discover frequency patterns for particular moods did not seem conclusive, however, approximating the mood based on tempo and tonality brought reasonable results. The findings on the effect of tempo on mood are not consistent throughout (Husain[3] and Tsang[4]), thus a level of arbitration is needed to represent moods through tempo to achieve parallelism with the moods considered in Bresin.[1]

To simplify the mood detection algorithm for more reliable results, four moods were picked from among the ones mentioned in Bresin's table of colors: sadness, happiness, love and anger. None of the given moods had a significant correlation with color saturation and value, except for anger which had an observed negative correlation with saturation — this correlation was decidedly not implemented due to aesthetic concerns. However, both tempo and tonality were correlated with color value according to the conclusions reached by Tsang and Bresin respectively.

## 3 Sound Analysis

Sound analysis takes place as a pre-processing stage, before the visualization is run. The sound data is collected in the module `analysis.py`. Much of the waveform analysis is done with the help of the audio and music processing library LibROSA.

The data received from LibROSA includes:

- The uncompressed waveform
- Separated harmonic and percussive waveforms
- Beat frames (list of time frames for which a beat event occurs)
- Mel Spectrogram (amplitude of each frequency bin over time)
- Chromagram (amplitude of each pitch-class over time)

From this initial data collection, more information about the song is gleaned. The frequency spectrum is truncated to an audible range. The spectrum is also condensed, via a weighted average, into a measure of the overall pitch at each instant. The overall key of the song is statistically predicted based on the prevalences of each pitch-class found in the chromagram, as discussed in `http://ismir2004.ismir.net/proceedings/p018-page-92-paper164.pdf`. From this key and the tempo, a base color value is chosen, corresponding to the mood of the entire piece.

The final conclusions of the analysis are then formatted into a python dictionary for use in the graphics module.

# 4 Graphics Implementation

The results from `analysis.py` are recieved by the module `graphics.py`. This module controls playback of the visualization with the help of the windowing and multimedia library Pyglet.

The graphics module uses vertex and fragment shaders to set the sphere mesh's positional and color data. After initialization of the window and global variables and constants, the update function is scheduled to be called regularly by the Pyglet app. At each frame, the sphere's radius, position, color, and displacement map are calculated according to the current sound data. This information is passed into the shaders, `DispMapped.vert` and `DispMapped.frag`, which update the appearance of the sphere on screen.

## 4.1 Volume and Beats

The radius of the sphere is calculated as a sum of contributions

$$\text{radius} = \text{minimum radius} + \text{current volume} + \text{proximity to beat}$$

Where the proximity to the nearest beat is expressed as

$$\left(\frac{\text{time between beats}}{2} - \text{time since previous beat}\right)^2$$

in order to have the attack and decay of the beat appear as a pulse.

## 4.2 Frequency Analysis

At each frame, we programmatically generate a new displacement map for our sphere. The map distends each latitudinal band of the sphere based on the power of its respective frequency bin, as explained in section 2.1

## 4.3 Average Pitch

The vertical positioning of the sphere is affected by the overall pitch as determined in `analysis.py`. As the average pitch grows higher in frequency, the sphere tends to move upwards, while lower pitches cause it to descend. The vertical motion of the sphere is modeled after a damped harmonic oscillator, where the equilibrium location changes depending on the average pitch frequency.

## 4.4 Mood

The color of the sphere at a given moment is determined by the base color found in `analysis.py` as well as the position in the song. The rgb values of the sphere's color are shifted according to a sinusoidal function of the form

$$\cos 2\pi \left(\frac{1}{2} + \frac{\text{current time}}{32 \cdot \text{length of a beat}}\right)$$

in order to achieve one full oscillation every 32 beats.

# 5 Improvements

- The overall pitch for most songs stays in the middle of the range and rarely changes much, making the vertical movement of the sphere go largely unnoticed. A revised heuristic could ignore more frequency bands to focus more on following the melody, as well as emphasize changes more to allow for a larger range of motion.
- Currently there exists only detection of major and harmonic/melodic minor tonality, due to Gómez's[2] analyses being based on classical pieces. Since songs of other genres may vary significantly from the reference distribution of pitch-classes for classical music, the algorithm currently in use is by no means perfectly accurate, but it is always able to figure out the key of the song and the tonality with minimal deviation from the correct result along the circle of fifths, given that the input song does not feature modulations.
- LibROSA's beat tracking function often accidentally identifies weaker beats as strong, which leads to unwanted rapid changes in color due to the way the period of color oscillation is currently calculated.

# 6 Results

A♯, as an initial attempt at a meaningful music visualizer, can be considered a success. There is much to be improved, but A♯'s foundation of intuitive visuals is strong. Hopefully A♯ will lead the way for other music visualizers to begin emphasizing meaningfulness over surface level appearance, and that eventually the sophistication of the analysis may allow for incredibly intricate detail that remains intuitive.

# References


[1] Bresin, Roberto. "What Is the Color of That Music Performance?" KTH - Royal Institute of Technology, n.d. Web. 12 Dec. 2014.

[2] Gómez, Emilia, and Perfecto Herrera. "Estimating the Tonality of Polyphonic Audio Files: Cognitive Versus Machine Learning Modelling Strategies." (n.d.): n. pag. Web. 12 Dec. 2014.

[3] Husain, Gabriela, William F. Thompson, and Glenn Schellenberg. "Effects of Musical Tempo and Mode on Arousal, Mood, and Spatial Abilities." University of Toronto, n.d. Web. 13 Dec. 2014.

[4] Tsang, Tawny, and Karen B. Schloss. *Associations between Color and Music Are Mediated by Emotion and Influenced by Tempo.* (n.d.): n. pag. University of California, Berkeley. Web. 12 Dec. 2014.