Cos 323 Final Project Write-Up
Collaborators:
Christopher Crawford
Louis Guerra
Andrew Zhou

Code:

Fourier.java - uses the FFT algorithm to convert an input discrete signal (array of double) into its frequency domain representation. This program contains various methods for handling the complex numbers involved in the calculation. This program was adopted partially from StdAudio.java by Robert Sedgewick and Kevin Wayne
Process.java – Contains 2 methods, *notes* and *normalizeVector. Notes* takes as input an array of doubles that represents the results from the FFT. The indices are changed into frequency values, which we then convert to semitones above or below middle A (440 hertz). This information is used to sum up all the amplitudes that represent each note of the chromatic scale. A 12-vector is returned; it represents the relative appearances of each note along the length of the song. Note that before the FFT information is analyzed, we effectively bandpass the song by omitting frequencies below 130hz (about C3) and above 2100 (around C7). This improves performance time and effectiveness by focusing on the frequencies which tend to carry the melody of the song.

NormalizeVector is a simple function that normalizes any given vector such that its magnitude will be 1.
Profilematch.java – Contains *getKey* which inputs the 12-vector from *notes* and compares it to the 24 standard vectors that were found to best represent the profiles of the 24 different keys according to the research paper, "What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered" published by Temperley. *GetKey* uses least squares matching to best match the input vector with one of the key profiles, and outputs the key of the song, in string format.
Keyfinder.java – Keyfinder is the primary program that combines everything else. It inputs a .wav or .midi file, and converts it into an array of doubles, a representation of all the samples in the signal. Keyfinder than uses Fourier.java to convert the song into a different array of doubles, which represent the frequency (of appearance) of all acoustic frequencies. This array is passed into *notes*, and then *getKey,* and prints the key of the input song.

Results:
After testing a few test runs where we tried our code on .wav files of pure tones produced from digital synthesizers, we ran our finished code on various songs with mixed results.

Song	Key Found	Actual Key
460 - Loops Soup Too	F# Major	F# Major
Demicheli Geminiani – Alegro in F Major	B Major	F Major
Bach - Partita in A Minor	C Major	A Minor*
Get Lucky – Daft Punk	A Major	A Major
Stevie Wonder - Superstition	A# Major	D# Major*

Martin Garrix – Animals	F Major	F Major
Skrillex – My Name is Skrillex (Remix)	F Major	F Minor*
Kill The Noise – Black Magic	G Minor	G Major
Michael Jackson – Beat it	A# Major	D# Minor
Rhye – Open	A Minor	G Major

We found that the complexity of the arrangement mattered little in the effectiveness of the program, as evidenced by the success of the program in handling songs with very noisy or distorted sonic aesthetics (Skrillex and Martin Garrix, and the near success with the Kill The Noise song). The complexity of the composition seemed to matter most, as we were completely off *only* with songs with more sophisticated melodic elements, such as *Open* or *Alegro*. Interestingly though, in the cases above marked with an asterisk, the program returned a key that was either the relative, parallel, dominant, or subdominant of the correct key. This means, that while our program may have been incorrect, it was certainly very close.

Looking Forward

If we were to continue to look for ways to improve on our code, we might start with looking into alternate algorithms (such as those involving wavelets) for figuring out the frequency distributions of songs. We could also add compatibility with other types of common audio files, such as .mp3 or .flac. Also, while our program was meant to be very general, we could potentially increase performance by optimizing our program for specific genres of music; if we have a good idea of the melodic characteristics that we expect to input, we could alter the algorithms to consider the most important features of the input song.