Key Detector Audio Plugin

We wanted to learn how to make audio processing software for our Senior Design. We all have a background in music, so this was our way of combining a passion for music with writing software.

Problem

Musicians and producers frequently communicate musical ideas through keys. A key is a shorthand for the series of notes in use. If a musician does not have the experience, key detection software can assist this process. Existing solutions either only work on files (Tunebat.com), or are expensive and too bloated for real-time use (Melodyne).

Solution

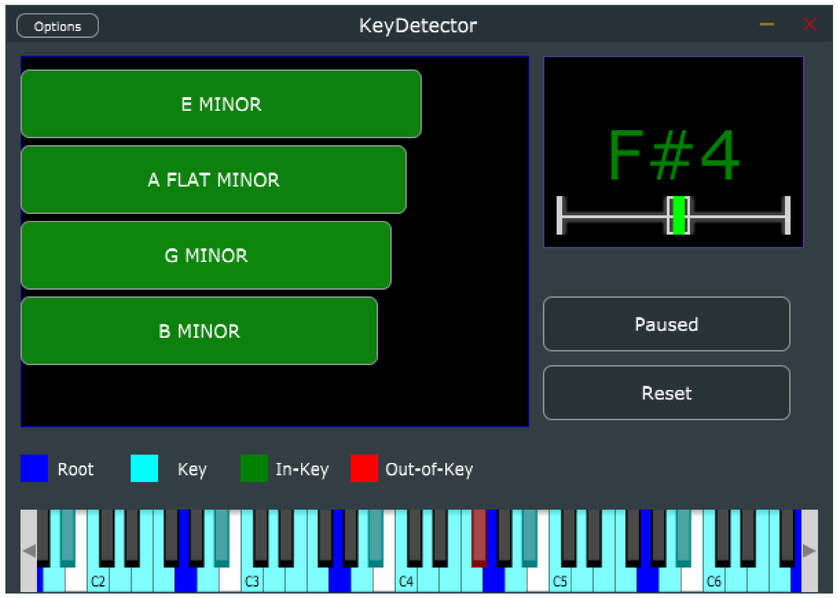
Our project fixes this problem by operating in a similar manner to an instrument tuner. Up to 10 seconds of audio is stored, then analyzed every three seconds while the detector is in “Playing” mode. It then provides the user a ranked listing of the four closest keys. When a bar is selected, the scale displays on the below piano roll, along with the active note.

Development

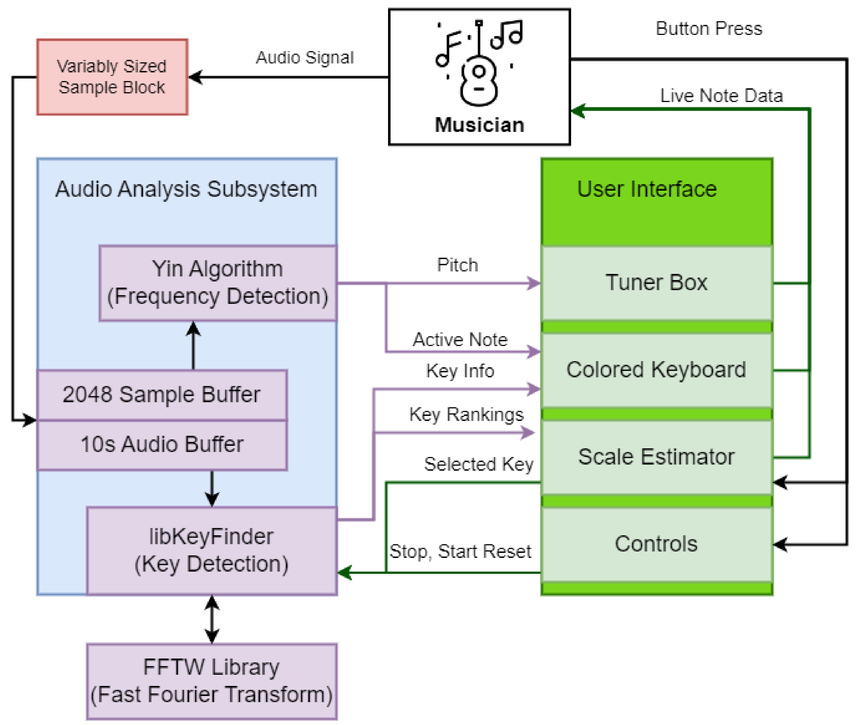
The Key Detector runs primarily as a .vst plugin for a Digital Audio Workstation (DAW). Development of .vst plugins occurs almost exclusively in C/C++ for its speed, and JUCE is the most established framework.

For pitch detection, the Yin algorithm was selected and integrated for its speed and robustness against noise. The libkeyfinder library performs key detection through several manipulations of Fast Fourier Transform output.

User Interface Screenshot: G Minor Selected



Architecture



Challenges

The initial challenge was learning how to use JUCE. This required following several tutorials, and referencing existing projects. Specific implementation struggles included:

- Integrating existing algorithm solutions.
- Linking external libraries like FFTW and ensuring cross platform compatibility.
- The audio thread cannot block for any time, key detection required ring buffers and threads to perform expensive computations asynchronously.
- Modifying open-source libraries to include project specific functions like ranked-keys and note highlighting.

Future Work

The current product is a working proof-of-concept. A port for mobile platforms like Android would be the widely useful platform for this application. This port would need the following additions:

- Gradual calculation of key for computation efficiency.
- Flexible GUI sizing.
- Licensing and a user friendly installer.
- FFTW library integration into Android.


JP Wang
Computer Science


Joe Welage
Computer Science


Klayton Hacker
Computer Science

Advisor: Prof. Boyang Wang