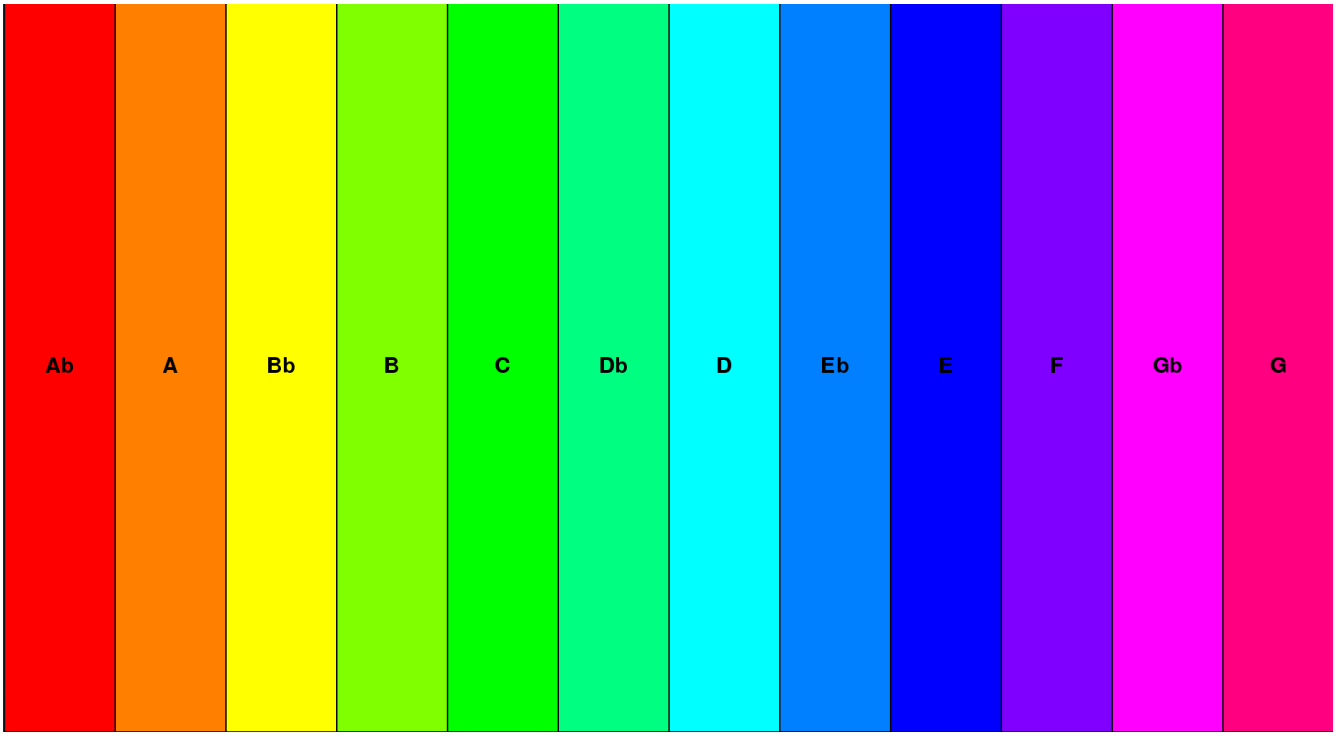
Virtual Keyboard

Julian Stone and Utsav Gupta

Project Overview

In this project, we aimed to make a “note board” that the user can play in a variety of different ways. This “note board” is essentially a musical keyboard with twelve notes (Ab to G). We wanted the user to be able to operate the note board with a mouse, with their computer keyboard, or with their body. Mouse and keyboard operation would be through event detection and body operation would be using computer vision to detect body location.

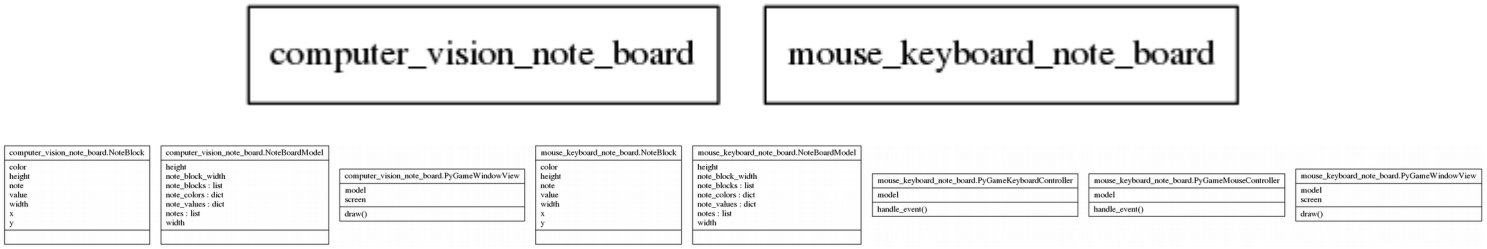
Results



We have two separate note boards: one that works with mouse and keyboard and one that uses computer vision. Both display a graphical note board with twelve note blocks labeled with their respective notes (as shown above). This graphical note board acts as a visual cue for the user to know where each note is. It remains the same so that the user doesn’t become confused as they are playing notes.

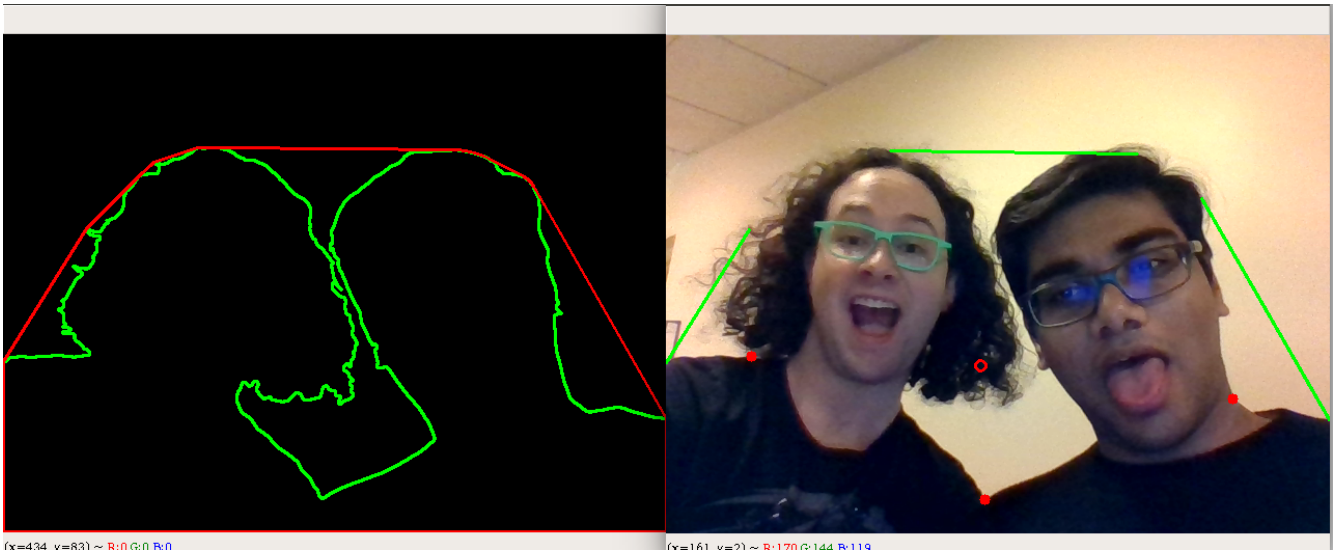
For the mouse and keyboard version, the graphical note board is used for mouse clicks primarily. Which ever tile is clicked, the corresponding note is played. The computer keyboard-based note playing only uses this as a visual aid as the computer keyboard uses the top letter row (“Q” through “]”) to play the notes. For the computer vision version, the graphical note board is also a visual aid that just tells the user which direction they must move their body to play certain notes. For example, the Gb note is further to the right, so the user would know to move further to the right to play notes close to Gb.

Implementation



Both version have a large portion of code in common. They both have identical code that constructs the GUI and any objects associated with it. These include a NoteBlock class, a NoteBoardModel Class, and a PygameWindowView class. The NoteBlock class houses all of the attributes for each note that is displayed. The NoteBoardModel class houses a collection of note blocks to be displayed. Lastly, the PyGameWindowView class turns the NoreBoardModel into a graphical display.

From here, the versions split. The mouse/keyboard based version uses PyGame to document and record mouse and keyboard events. Two separate controller were created - one for mouse and one for keyboard – that handle the PyGame events and act on these events as needed. For the computer vision version, we use OpenCV to detect images in front of the camera. It makes an outline of what it sees and finds the center of the object in front of it (see below). We use this center by mapping it to the coordinates of the note board. It translates this into a note index, which determines what note should be played.



Reflection

Overall, we were very efficient. We were able to have the graphical note board done on day one and had it working with a mouse the next day. It all just fell into place. We decided to us a divide and conquer method to address this idea as we believed it to be quite complex. Julian tackled most of the PyGame and Utsav worked heavily on the computer vision and the optimization of Sonic Pi. We each set up our bits of code so that it would be easily integrated with the other person's work.

That being said, we still believe that this was properly scoped. While we were very efficient, we did find difficulty and challenges that made us feel that we chose a proper project. Due to our efficiency,

we could have pushed for a little bit harder of a problem, like using our fingers through computer vision to operate the note board, but we believe that we would have needed a bit more time for that outcome. As a bit of improvement for next time, we both believe that we could make clearer and more reasonable checkpoints so we don’t get to far behind or ahead of the work.