Interactive Visualization

Project Overview

The goal of the project was to create a “paper piano.” A user would able to move their finger across notes on a paper keyboard, and the program will use their finger location to play notes. While there were some issues with our more sophisticated version, we were able to create a minimally viable product which plays a few notes. The advanced version has issues with the integration of finger tracking with the MIDI player (we intended for a user to select keys with a red object), and experiences extreme slowdown when playing frames. We used timidity as a MIDI player, the package music21 to play music, and Open CV to have user interactivity through video.

Results

The minimally viable product works fine. The program splits the screen into several parts, where each part represents a note. If a notes part detects motion, then the corresponding note is played. There is enough lag between each note play to ensure there is no overlap. The only issue is the user’s computer must be held very still with a solid background (preferably a picture of our piano interface) in order to function. Instead of sliding fingers across the keys, the program works best if the user’s finger only enters the video frame on the note they wish to play. In order to experience these results, please use the program called “interactivity.py.”

While the minimally viable product works, we decided to use the color red to track users finger across a screen to play a wider variety of notes (including some sharps and flats). We also wanted to have a video overlay of a piano, which we created in GIMP (Figure 1), and show a red marker to indicate where the user’s finger was. When the program is run music plays and finger tracking appears to work. However, there is considerable lag when trying to show video. In order to look further into these results, please use the program called “redmusic.py.”

Figure 1: Piano Interface

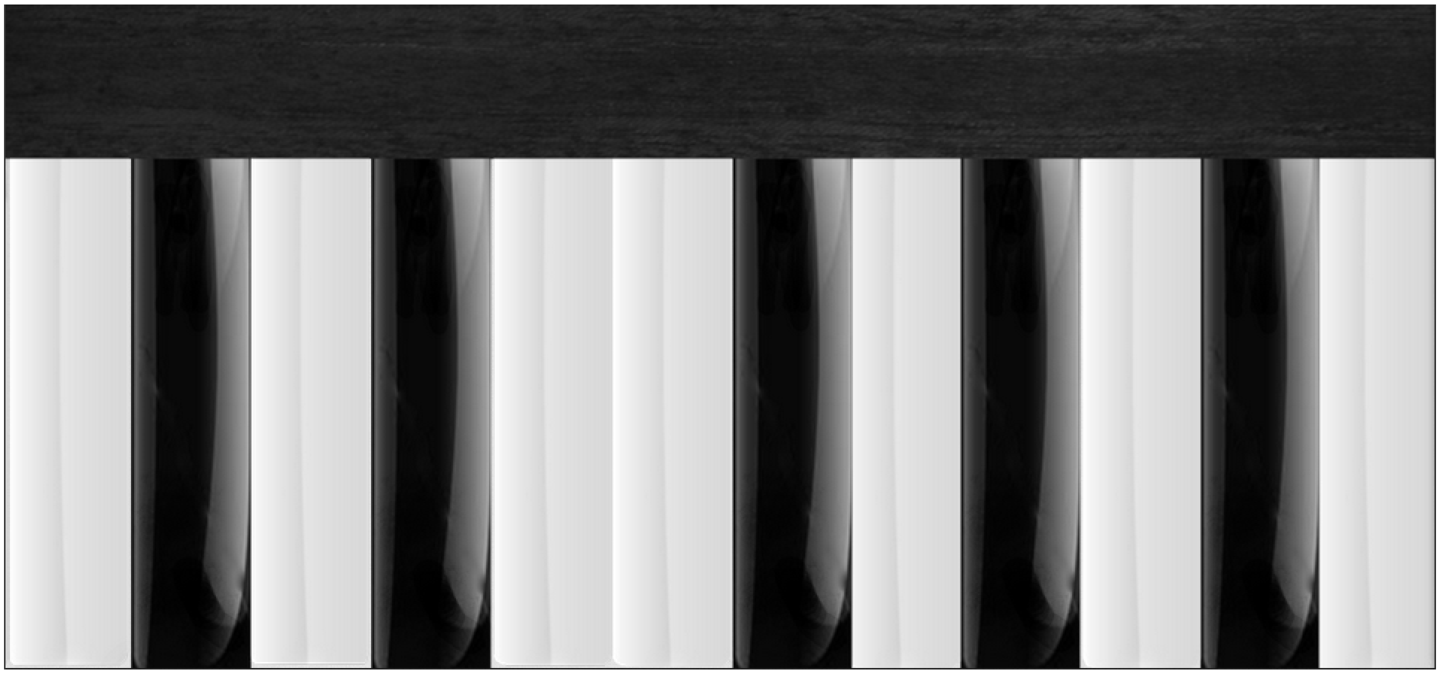
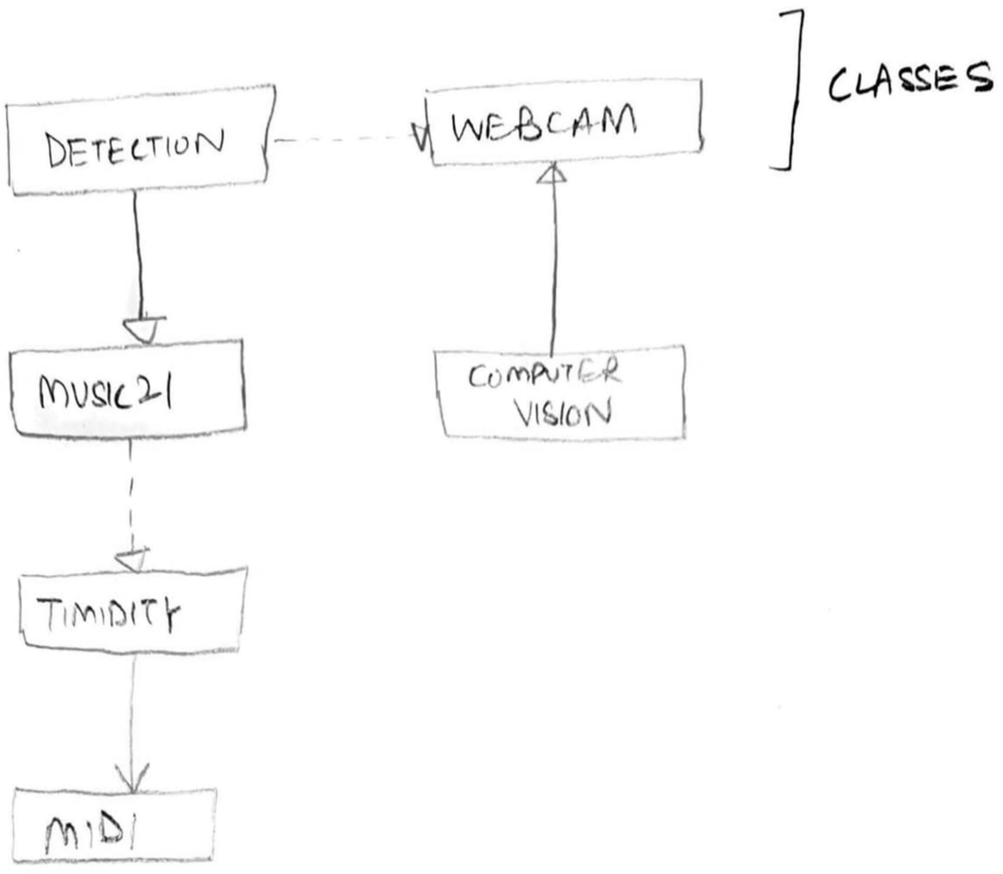


Figure 2: UML Diagram



Implementation

For the first iteration of the paper piano, we use a class called 'Webcam' that captures frames from the webcam. We use a separate class called 'Detection', that can detect changes in the regions (which are the zones that the screen is divided into, representing the keys of the piano), and outputs the note that the user wants to play, as a string. The divisions are made using basic mathematical operations. We further use the MIDI player to translate this to sound, so as to play the right frequencies for the notes – the notes are strings as elements of a list dedicated to store notes for one complete register.

For the second iteration of the paper piano, we use color tracking to track a color that is in sharp contrast with the keys of the piano (since that makes it less prone to error) – we chose the color red.

This was a big design decision for us since we considered how users would prefer to use a more intuitive program – one that can be played by actually placing your finger on a printed version of the piano, which would contain the natural and sharp keys, matching a real piano more closely than the air piano. It would also give the player an ability to play the sharps and flats more clearly and in a hassle free manner (instead of using a button, as in a previous iteration).

We give the program a threshold range – if it sees anything in this range, it treats that as the user's finger(s), and outputs a note that the user wants to play, as a string. We further use the MIDI player to translate this to sound, so as to play the right frequencies for the notes. As a later iteration, we want to display the printed piano on the screen and show a pointer that tells the user where their fingers are, or in a sense, where the computer thinks their fingers are, in order to make this application more intuitive and user-friendly.

Reflection

Creating a basic prototype went fairly well. We believe it was a good scope for the project, albeit a little too simplistic. When we wanted to include features like live tracking, UI overlays, and more notes, errors began to occur. We split the programming into two parts, and then integrated each part. One of us initially worked on integrating the music coding, learning about music21, and creating graphics. The other worked on the interactivity aspect. Once the music coding was done, both team members worked on fine-tuning user interactivity. Most of the work was completed separately with frequent communication.

This project taught us a great deal about the creative possibilities of Python. If we had more time with this project and were able to perfect the note-playing process, a lot more features could be added. There could be selections for different instruments, the ability to play chords, and script saving capabilities. Even what with we were able to do, we learned much about Open CV and music21. If we were to do the project again, we would have tried to narrow the scope of the project sooner. At first we were discussion having all sorts of creative options, and underestimated the difficulty of the project.