Anderson Sutandinata
15112-Fundamentals of Computer Science
November 25th, 2020

Term Project Design Proposal: “Cheapsaber” with TP2 updates

Project description: With the price of a VR headset being a barrier to many people, I decided to simulate the popular virtual reality game *Beatsaber* using only everyday objects you could very easily find or make at home. This project will track an object that a user will swing, and treat it as a saber. The game will then display notes coming towards a player, that the player will then have to swing at in the correct direction.

Competitive Analysis: Obviously, this project will be very similar to the actual game of beatsaber. The gameplay will be similar where you have to swing a lightsaber in order to hit a beat note in time. While in the game you use two sabers and a vr headset to slice “beats”, this project will focus on only one saber for now but not use a vr headset. Furthermore, this game will display the screen on your laptop and only use its bui;t in camera, to eliminate the need to buy anything.

Although I could not find any similar projects focused on cv beatsaber anywhere online, I could find projects using similar ideas. Within the scope of 15-112, a similar project exists being ar pong, which also uses ar tracking of a real-life object, and implements it into a game. This project will be different, as it treats the object being tracked as a 3d object rather than just tracking its xy position. Apart from that one project, I have been unable to find any other projects that attempt to recreate beatsaber with opencv tracking.

Algorithmic Plan: The hardest part of this code will definitely be the conversion of a 2d saber read by the camera, into a 3 dimensional model. To accomplish this, I plan on representing the saber’s state in each frame as a 3 dimensional vector where the z axis is how close the saber is to the camera. Since we can assume that a saber doesn’t increase in length, we can take it’s highest length in the calibration section of the project. Given a saber’s magnitude and it’s magnitude in the x y axis, we can determine how far into the z plane the saber goes. For the actual slicing, I plan on treating the saber as a polygon and use the sutherland hodgeman algorithm for slicing each block.

Structural Plan: The code will be split up into three main files: one containing all the cv tracking, another controlling the gui interactions with the user, and the last containing the randomized map generation.

Timeline Plan:

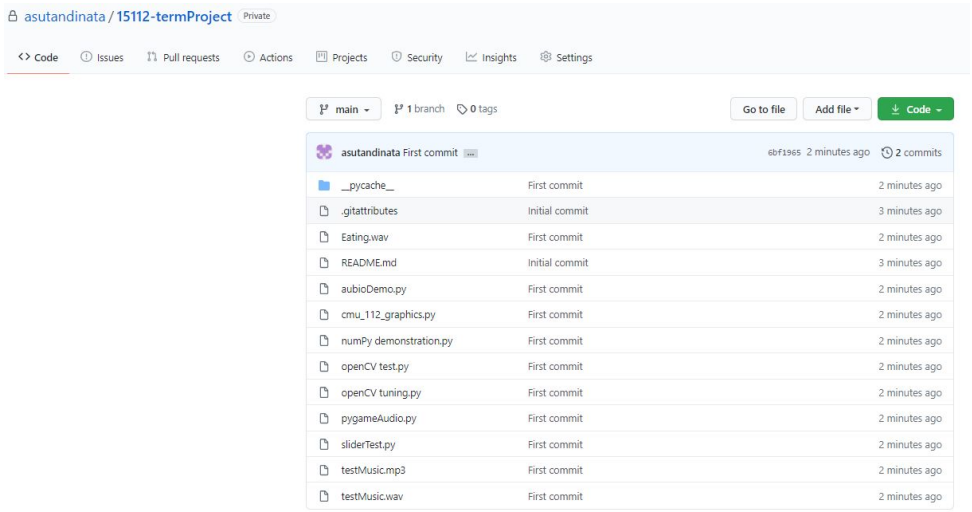
TP0: Have a demonstration of the saber tracking and other tech demos complete.

TP1: Have the tracking of the saber be able to represent the saber as [x,y,z] , and determine what direction/velocity a swing is in. Also have a calibration screen ready for calibrating a saber in different conditions

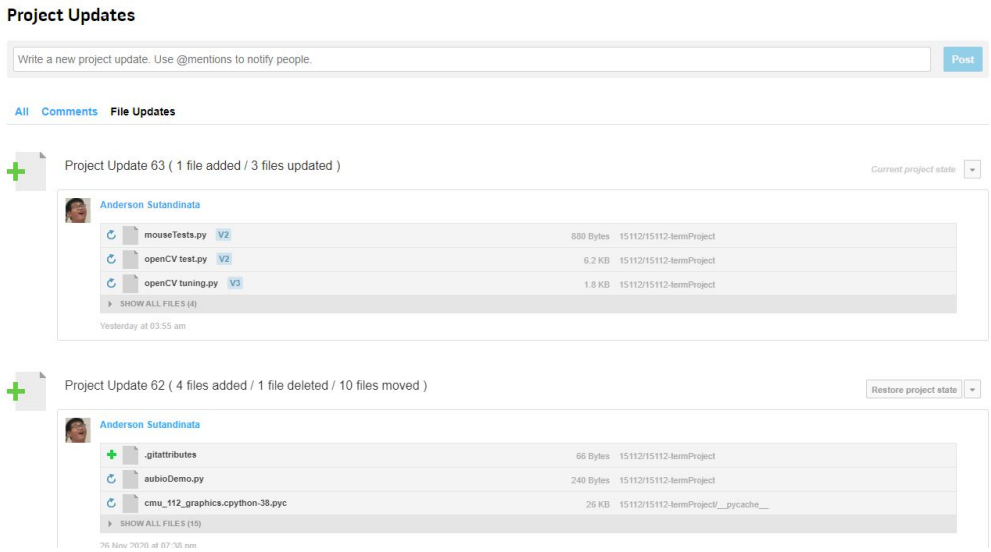
TP2: Have the notes displayed come towards you, and ensure that the saber slices the notes when it's appropriate. Also have the music playback work when playing the game, and a basic ui for selecting if you want to play a game or calibrate your lighting.

TP3: Have the code be able to run songs consistently, maybe add pause feature

Version Control Plan: A private github repository will be used to store this project in the cloud, and help ensure version control in case my code happens to become irreversibly bad. I have included an image of the private repository I have created for this project.



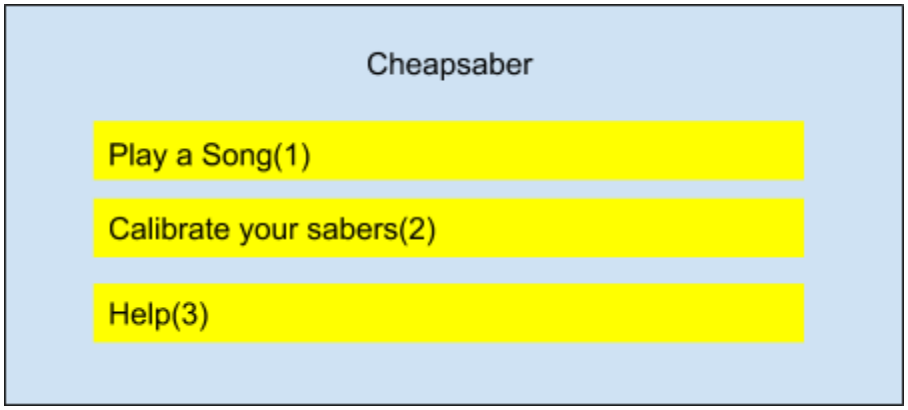
Grabcad will also be used for backups as a secondary cloud storage platform.



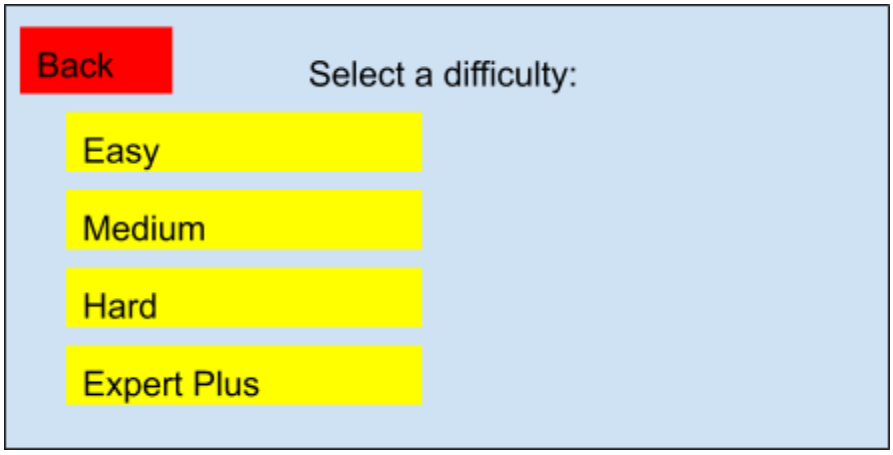
Module List:

- TKinter for graphics
- Pygame for audio playback
- OpenCV for vision tracking
- Numpy to assist with opencv
- Time to determine angular velocity of sabers

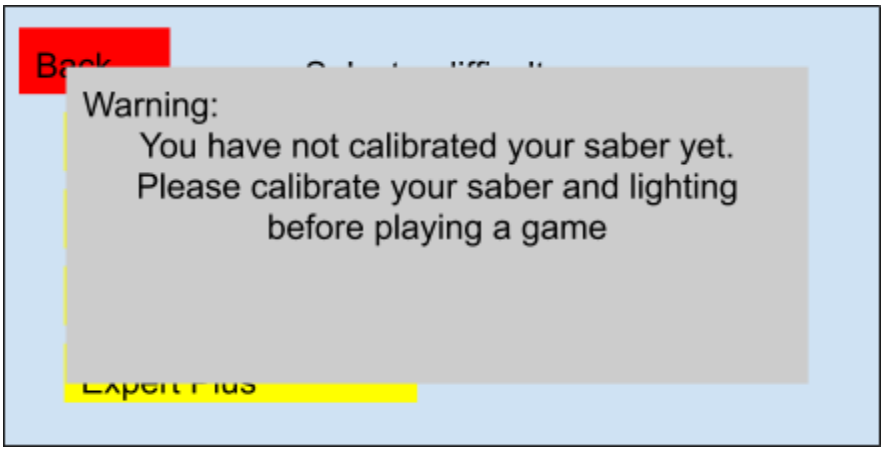
Story Board:
User enters a main menu with 3 options: (1)starting a game,(2)calibrating your saber, or (3) an instructions screen



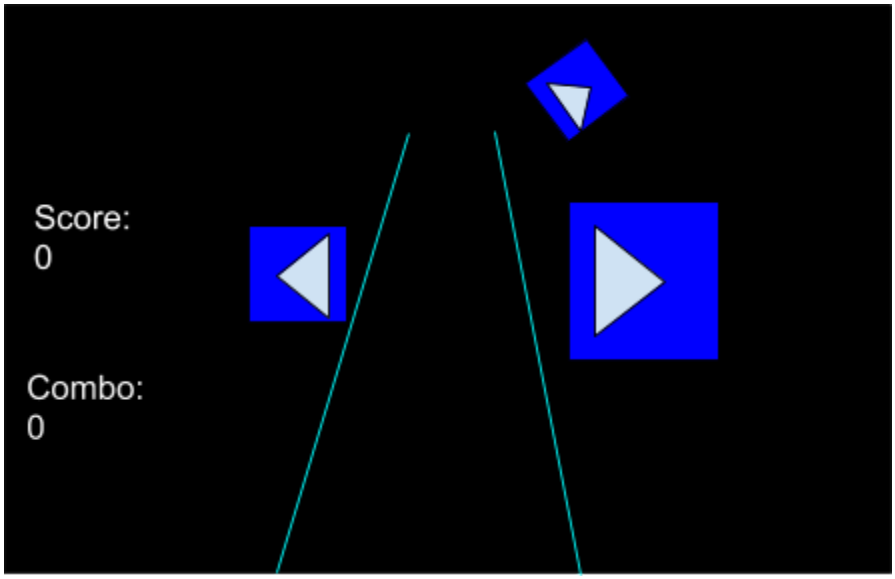
If a user starts a game(selection 1)
1a. If a user decides to play a song and they have already calibrated their saber, they will see the following screen:



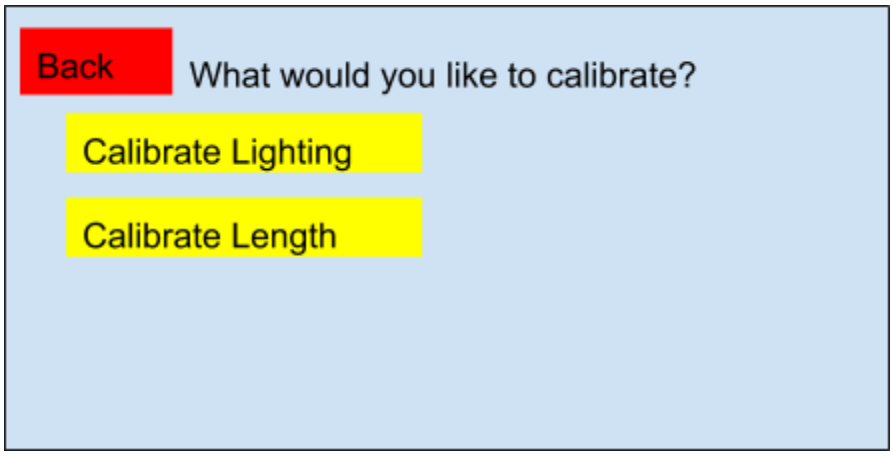
1b. If the user has not calibrated their saber, they will receive a warning saying they cannot play until they have calibrated



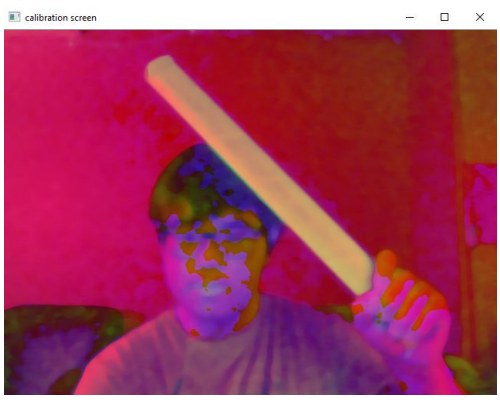
1c. If a user selects a level, and has calibrated their sabers, they will enter a level screen where they can see their saber, and notes appearing



Once a song is over, the user is returned to the main menu
If a user calibrates sabers(selection 2)

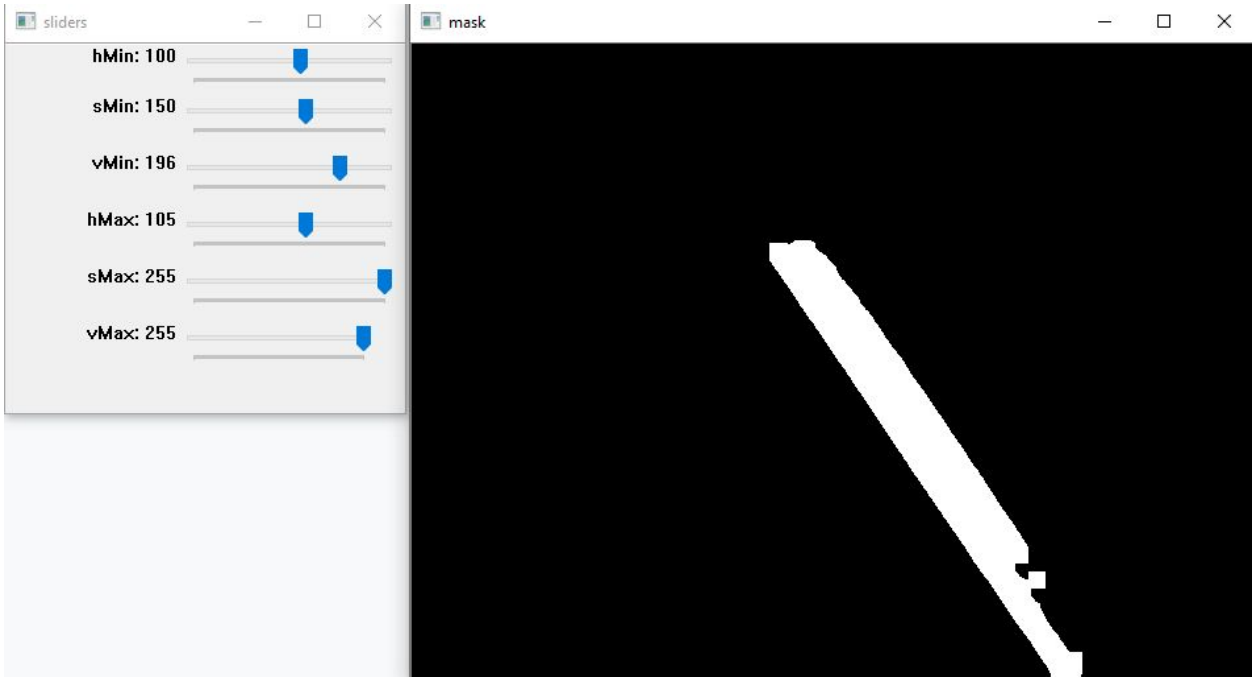


2a. If calibrate lighting is selected, a window will pop up with your camera and the user will click on their saber to perform a rough calibration

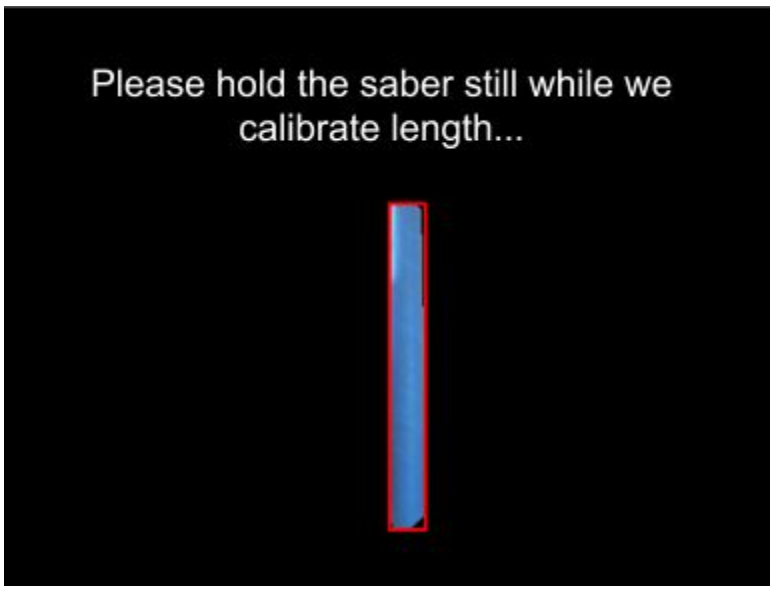


Actual image from the lighting calibration where user is shown an hsv filtered camera screen and they click on their saber

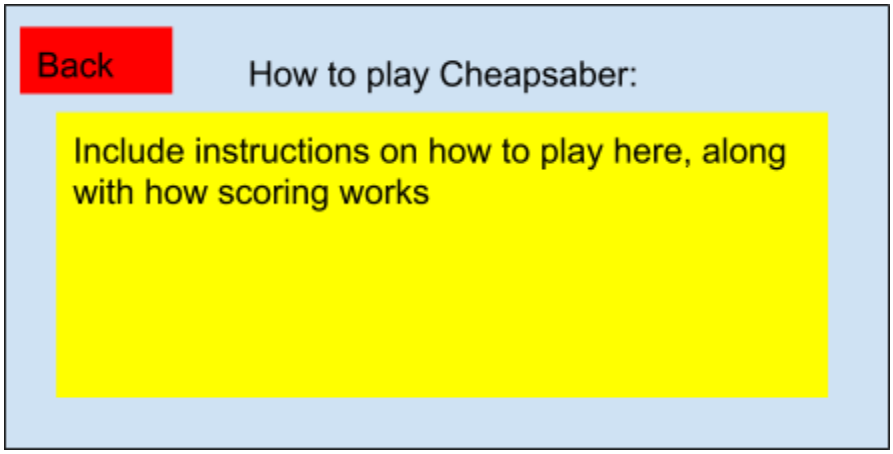
The user will then be able to modify the sensitivity to perform fine adjustments on the lighting calibration for improved saber tracking, while being displayed what the computer actually sees. The values start off based on how the user clicks, in the previous setup window, and



2b. If calibrate length is selected, the user will be asked to stand still, and hold their saber right in front of them, for a few seconds. Once complete, the computer will automatically exit this setup.



3. If help is pressed, the user will see a help screen providing instructions on how to play the game



TP2 Update:
Since TP1, a few changes have been made to this project. First off, the note slicing will be done as a 2D slice, rather than a 3D slice. Second, sutherland hodgeman will most likely not be used as its use case is going to be slightly different than what I had planned. Instead, I will simply just calculate two new polygons based on the slice type made(vertical, horizontal, diagonal, etc.). From there, I will display the note breaking, in the direction of slice.

I also plan on changing the post mvp goals slightly. I would like to focus more on gameplay, and polishing the program than trying to add some 3d slicing. The next few steps will be polishing what I currently have to run with fewer bugs, improving the slicing animations of the notes and bombs. I'd also include UI improvements such as including a timer to let the player know how much longer the song is. A stretch goal would be to include in game 'walls' that the player would have to dodge, just like in the real game. The haar cascade facial detection algorithm would be used to complete this.