Interactive Programming

Bill Du and Harper Owen

Project Overview

Our goal was to create an interactive game that allows 2 individuals to control their own game-piece through webcam connectivity in order to play a classic version of pong.

Results

The game has a fully functional interface that connects with players via a webcam. The players use colored objects to control their paddles and hit the ball across the board to score points. We used pygame for the graphics of the game, and although they are simplistic, they replicate the style of the classic arcade game.

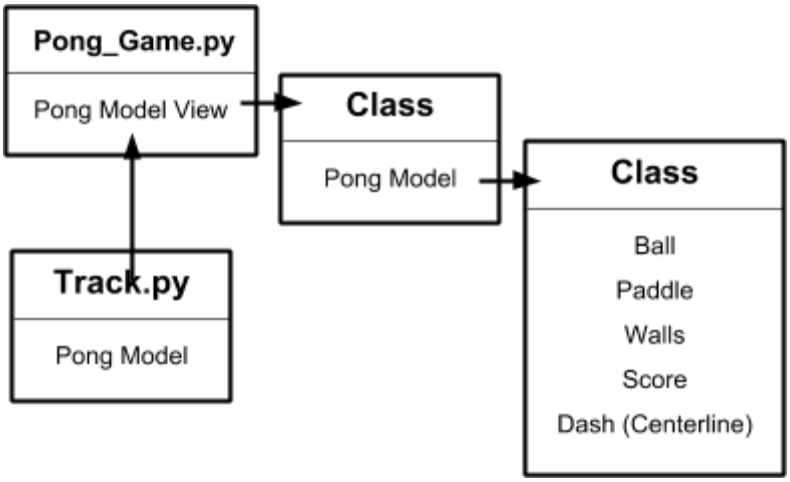
The players can choose either a blue or yellow token to guide their colored paddle up and down across the field while the ball. This is achieved through holding the token in front of a webcam, which filters for color and location of the object, feeding that information to the game's graphics display.

Implementation

For the pygame side of this project, we chose an adaptation of the in-class brick breaker code. This consisted of several base object classes, including the ball, paddle, wall, score, and centerline classes. These were housed in a parent class, Pong Model, that defined some of the variable arguments for the objects as well as defining collision between them. This, in turn, was fed to the class Pong Model View, which was where pygame rendered all of the information onto the screen.

The OpenCV software is responsible for feeding position information directly to the paddle class. This program takes the frame input from the webcam and filters for the specific colors (blue and yellow), then uses opencv Structure and Contour to register objects and determine their centers. This then relays the position to the Score Class using the Pipe process, which allows the code to continuously update the paddle location.

A design choice that we made was using the pipe process to feed information from one file to another. We could have enclosed our Track file in another method within our pong file, but in the end we decided to keep things clean and separate by using this method.



Reflection

This project was appropriately scoped in that we struggled through a couple of problems, but were still able to reach our goals in a reasonable amount of time. If we wanted to improve our game we could potentially work on image processing, blurring each frame to prevent noise, or targeting colors with more accuracy. We also could have used the inheritance concept to integrate our classes, but because it is functional as is, we didn't feel a need.

Moving forward, the use of classes and their versatility will be very important, as will the pipe process. We would have liked to see a different opencv tutorial, because the toolbox was dealing with facial recognition which we couldn't really apply. Still, the in-class brick breaker walkthrough was a great way to get us started and over the first few hurdles of the pygame learning curve.