## Project Overview

For this project, we created a two player pong game whose paddles are controlled using OpenCv. We broke our project into two halves. One consisted of a pong game for with we used pygame. The paddles for this first step were controlled by the mouse and keyboard arrow keys. The other half was a object tracking program implemented with OpenCv. After getting a good start on both of these components, we integrated them so that the object tracking replaced the keyboard and mouse controllers.

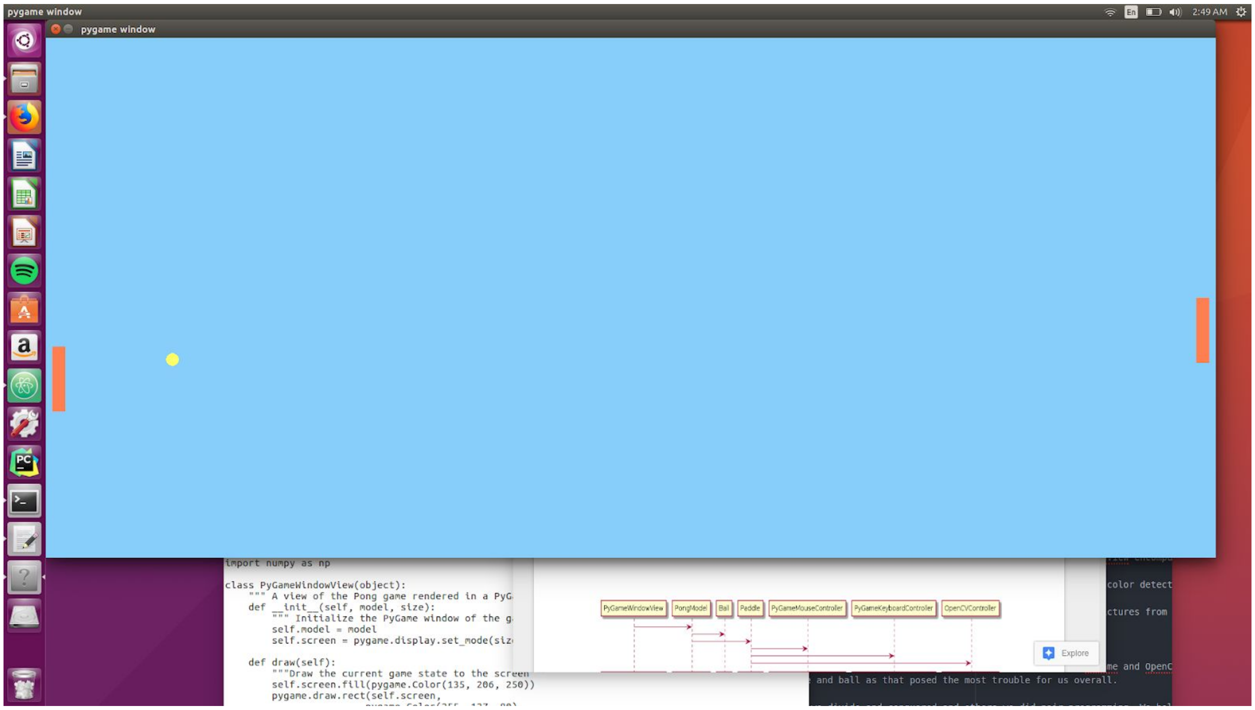
## Results

In this project, we were able to design a game of Pong with OpenCV. Our project uses the color detection of OpenCV in order to operate as the paddles someone would use to play the game. The two colors we used was green and blue and created individual filters for each of them so that player 1 and player 2 can be identified uniquely.

We used many classes in order to develop each aspect of the game such as the game window. Other aspects of the game, such as the interactions between different components like the paddle and ball were put into these classes, however very difficult to do. Our final results produced this game.

## Implementation

The main components of the game were the controllers of the paddle and the pong model. We depended a lot on the paddles and their interactions with the boundary, the ball, and the method of control. On the most basic level, the paddle was controlled by the mouse and keyboard. For the mouse controller, the paddles were easy and remained on the screen all the time; however, for the keyboard controls, we needed to set boundaries for the controls so they remain on the screen. For this we used “if statements” in order to make sure that the paddles, and in a similar method the ball, remained on the screen.



```
import numpy as np

class PyGameWindowView(object):
    """ A view of the Pong game rendered in a PyG
    def __init__(self, model, size):
        """ Initialize the PyGame window of the g
        self.model = model
        self.screen = pygame.display.set_mode(siz

    def draw(self):
        """Draw the current game state to the screen
        self.screen.fill(pygame.Color(135, 206, 250))
        pygame.draw.rect(self.screen,
```

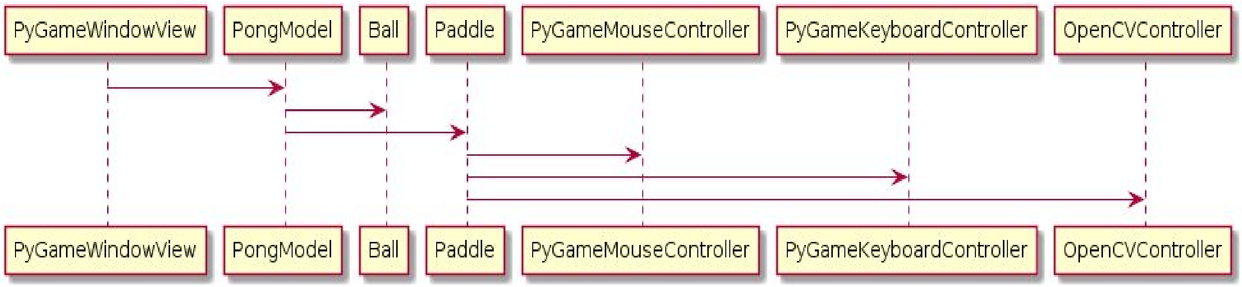
color detect

ctures from

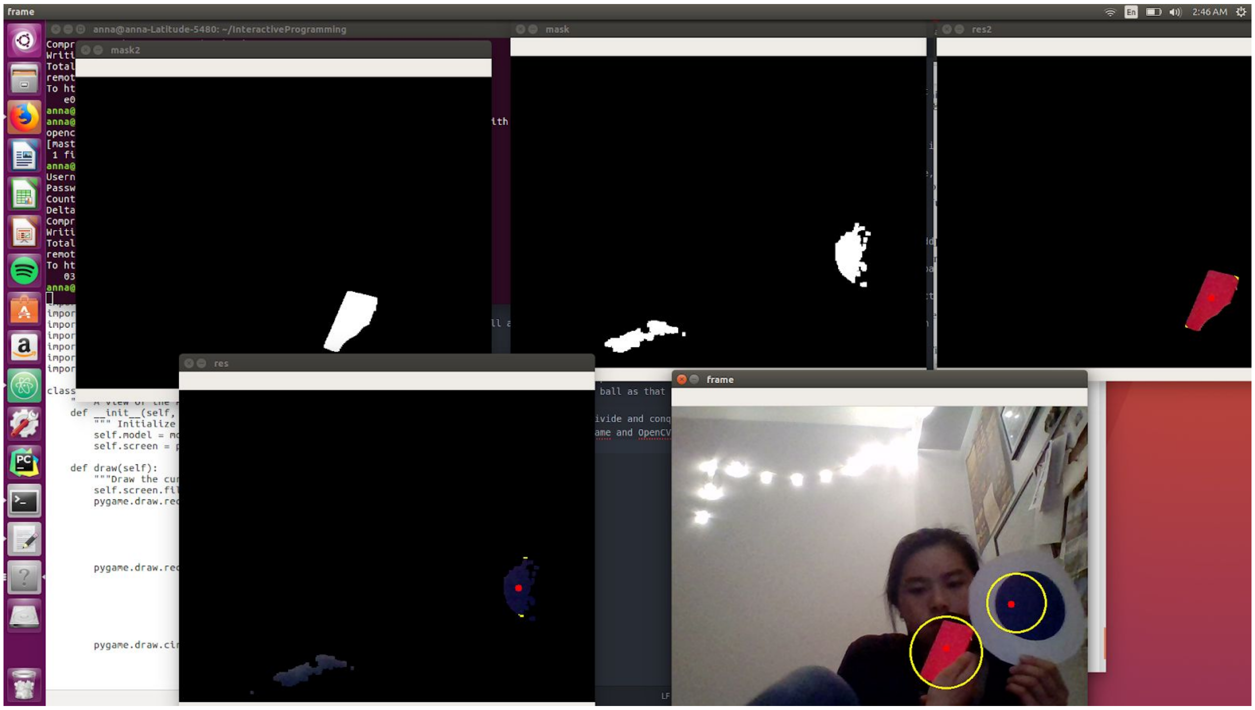
me and OpenC

and ball as that posed the most trouble for us overall.

The following UML diagram shows the flow of the classes as they build into another with the PyGameWindowView encompassed the entire code:



In our code, we were given the option between the type of controllers we want to use and we used OpenCV color detection because we wanted our game to depend on the use of the color detections to serve as the main form of playing rather than the keyboard or mouse. From this, we were able to make the game more exciting and reflex based compared to the normal game of pong.



```
Compr
Writi
Total
remot
To ht
   e0
anna@
anna@
openc
[mast
 1 fi
anna@
Usern
Passw
Count
Delta
Compr
Writi
Total
remot
To ht
   03
anna@
```

```
impor
impor
impor
impor
impor
impor

class
    def __init__(self,
        """ Initialize
        self.model = mo
        self.screen = p

    def draw(self):
        """Draw the cur
        self.screen.fil
        pygame.draw.rec



        pygame.draw.rec



        pygame.draw.cir
```

```
ith
ll a
ball as that
ivide and conq
ame and OpenCV
```

Additionally, when designing the ball and paddle, we drew them and gave them color rather than import pictures from the internet to practice coding with pygame and to make the game more authentic. However, this made the process in the long run harder as the main functions that involve collisions in pygame used the Rect function that we did not use as we drew these images.

## Reflection

This project overall went well with some challenging aspects as we had little to no experience with pygame and OpenCV. We both wanted to try something new which led to our project design and overall it went well. We learned that when we were not programming together, we need to include more unit test so the other could understand the code more, so going forward on group projects, we will seek to make the code more readable for both partners. However, before this project, we wish we knew more about the dynamic of collisions between the paddle and ball as that posed the most trouble for us overall.

Our working style was pretty diverse as sometimes we divide and conquered and others we did pair programming. We believe this occurred because of our individual schedules calling for us to work separate so things can get done and other times we did pair programming so we both could understand and to get the code done. The main issues were when we both did not understand how to fix an error or when we both were confused because that made the process slow down a lot overall. For the most part, we reported to research and utilize the resources for pygame and

OpenCV, for example the tutorials and documentation documents, but we asked for help from ninjas when needed.