Grace Montagnino
Quinn Kelley
March 16, 2018

Interactive Programming Reflection

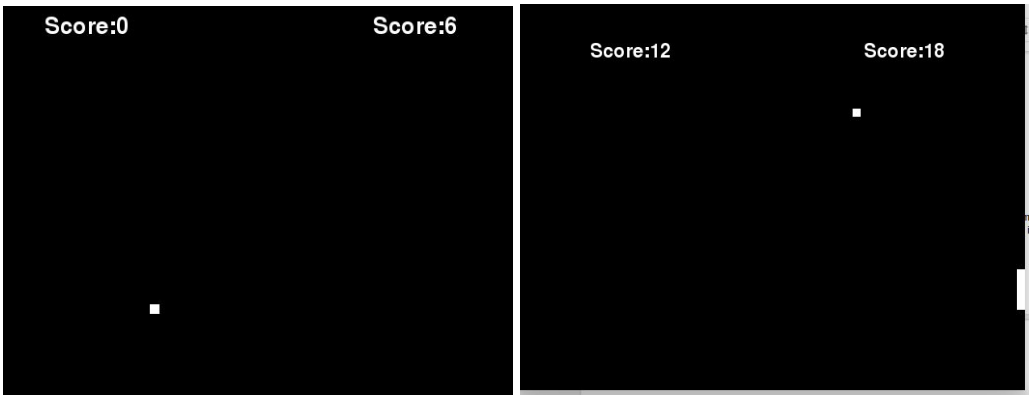
Project Overview

For our interactive programming project we made a python adaptation of pong and air hockey. We used the pygame library to create our two player input system along with our graphics. Our game is a two-player game where each player controls a paddle. The players bounce the ball back and forth and try not to let the ball get past their paddle, or the opposing team gets a point. If they are able to move their paddle to the right spot to hit the ball, the ball will bounce off and begin travel in the direction of the opposing team.

Results*]*

Our final product is a game that blends the games of air hockey and pong together. The game works like pong in that players are restricted to a single axis of motion, but is more like air hockey in appearance. Below is an image of the game set-up:

The game starts with the puck choosing a random velocity. With this set-up the game is more fair because neither player has any idea how the puck is going to move, all they know is it will start in the center of the field. Examples of this are in the collage below.



Each player uses keyboard controls to move their paddle up and down to prevent the puck from getting past them and into their goal. The object of the game is to get the puck past your opponent's paddle to score a goal. When the ball gets past the paddle, a point is awarded to the opposing team, and a new round begins (but the score is carried on). With each new round the puck gets a new random velocity as in the beginning of the game. Scoring is updated throughout the game, where each players number of points is kept directly above their paddle's goal station. This is what the scoring system looks like:

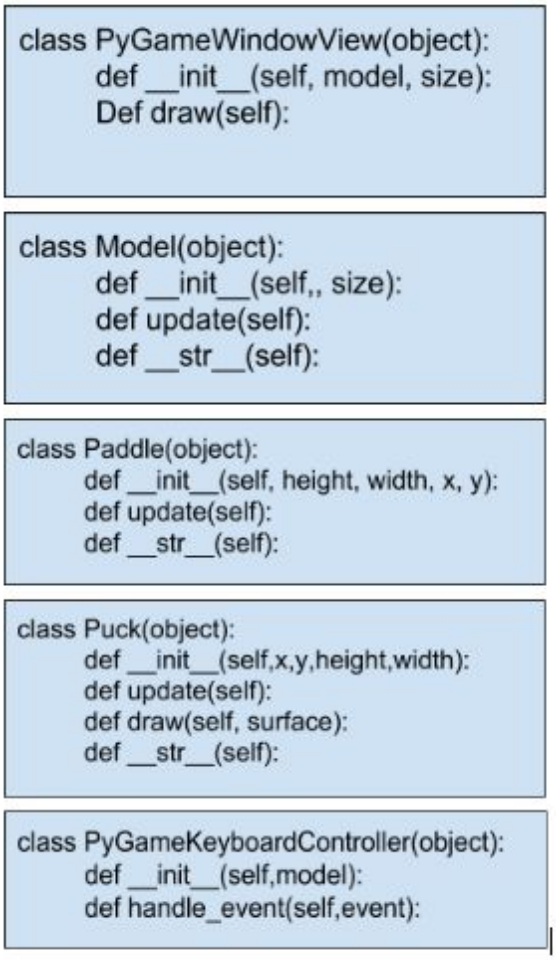
```
def update(self):
    """ update the state of the puck """
    if self.y>=480 or self.y<=0:
        self.vy=-self.vy
    if self.x>640:
        self.x,self.y=320,240
        self.score2=self.score2+1
    if self.x<=0:
        self.x,self.y=320,240
        self.score1=self.score1+1
```

Implementation

We divided our system into separate classes, examples including sections for Paddle and Puck. These classes contained the attributes for the objects, such as x-y position and velocity. We used the Pygame Keyboard controller module in its own class to alter the position of the paddles. This worked by creating a dictionary of the keys which were pressed down at any given moment, and changing the sign of the paddle's velocity depending on which key was pressed. We also created a PyGameWindowView class, which drew the objects (and the game scores) at their updated states.

One modeling choice that we made was in the Keyboard controller class. At first, our paddles were controlled by detecting when a key was hit and altering the position of the paddle accordingly. This made the game very frustrating to play, because you would have to repeatedly

hit the keys in order to get the paddle to move where you wanted it, so we changed this handling method in favor of holding keys down to change velocity.



Reflection

Overall, this project was a positive experience for both of us. We timed out the work in such a way that it was never overwhelming, but there was always something that could be done. This allowed us to make good progress while minimizing the stress. That being said, when we originally planned out our project, we were concerned that a multiplayer game would be too challenging. Once we discovered that this was not as big of a deal as we were initially thinking, we raised our expectations for the game, assuming we had low balled it. We changed our plan from pong to air hockey, thinking that the challenge was more suitable to what we wanted to get out of the project. But as we got into the weeds of the code, we realized that the seemingly small differences between air hockey and pong were not so miniscule from a programming standpoint. Thus, we both over-scoped and under-scoped our project, but ended up landing nicely in the middle with a game that is a merge of pong and air hockey principles. Moving

forward, we feel that if we had spent more time considering what the code would involve beyond classes (collisions, boundaries, scoring) and spent a little bit more time thinking about how those would be implemented, then we could have more accurately scoped our project out from the beginning. We also wish we had spent more time trying to understand the basics of pygame and the assumptions it makes because we ran into several instances where the error centered around pygam operating differently than we expected it to.

As far as working as a team, both of us were on similar pages as to how we thought that would best happen. We started out by pair programming to get the foundation laid out together. After this we sat down to plan out the class structure for our project. From here we divided and conquered, always willing to step in to aid the other person if their part was giving them trouble. The division of work was quite smooth, we had no major issues with balancing work loads. The biggest issue that arose out of our divide and conquer strategy was merge error, and unintentionally overwriting a line of the other person's code. But we were able to sit down and slowly work through the code line by line to handle the merging conflicts. Next time, we could be more careful and aware of how the code we write affects the code already written to try and avoid these challenges. Overall, we worked together comfortably, as we worked at similar paces and levels, as well as both maintaining a similar definition of a successful project and teamwork.