# Medieval Pong Process Work: Obstacles

Ryan Thompson

Process:

I started my program by first coding the necessary variables for the obstacles, by making the program function code require the variable for the Raylib.DrawTexture code so that when it is defined in the main program, that I can code each level differently. For each level that I coded that were stationary, I coded them to draw out each obstacle on the screen with rows and column variables that are used with a for loop index. The third level I decided to add movement code to make the obstacles move up and down within the screen boundaries, but my first iteration of the code didn't work so well as I attempted to move a single obstacle around the middle of the screen within a certain boundary but, couldn't keep it within the boundaries and get the ball to move at all.

My main process for making the obstacles show up on the main program was to code so that the obstacle can be initialized declared in the level program which would then be called into the main program. However, although I managed to get the obstacles to appear in a grid like format in separate functions (levels), trying to call that second program in to the main one was getting more and more difficult as when you add a variable for a separate program (Obstacle[] obstacles) you can declare functions in to the update and setup for drawing or initializing the code (obstacles.DrawObstacle). The problem became that it wasn't allowing me to call those other functions, always defaulting to; levels[].Equals(obstacles), and couldn't figure out how to solve it. So, I decided to just move the level functions into the main program.

For the textures of the obstacles, my original attempt to implement them would be to draw them over top the block shape obstacles then color the blocks blank to act as the collision border of the texture. However, trying to draw it over the shape didn't seem to show up on the screen no matter how many times I did it, so instead I changed the code I already had for the square shape, to work for the texture.

There was a lot of trial and error trying to get the collision code to work, I attempted to code the collision code directly into the update function using both for loops and foreach loops, using the Raylib.CheckCollision code. Unfortunately, that wasn't working either as the ball kept phasing through the obstacles. Finally, after asking Tristan about some collision code that he had used from a previous project, I decided to incorporate it into my obstacle collision function (changing it from a void to bool function, adding the ball as a variable and defining each side), afterwards, the ball was able to collide with the mob obstacles.