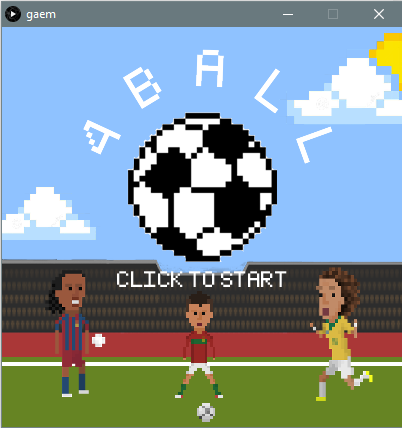
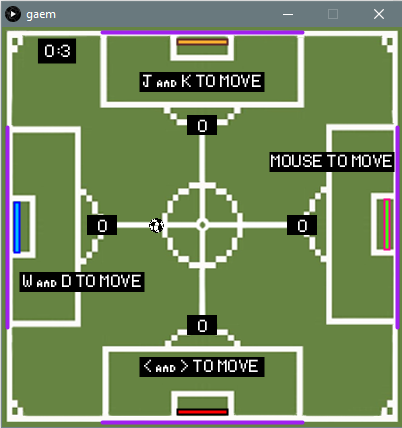
**JAVA Game Project Overview**

Layout Images:

**Program/Method Hierarchy:**

Within the class Game() are all of the methods or groups of similar methods used within the program in order of greatest to least importance:

1. beginScreen() and startup() – these two methods draw all of the media shown on the starting screen apart from the title and allows the user to enter into the main menu screen. Without this method the game could not be reached at all
2. menu() – this method allows the user to select from a choice of two game modes: ‘ 4 player‘ and ‘1 player’ and triggers the start of the game
3. drawBall(), drawPaddles(), drawField() and drawNet() – these 4 methods do exactly what they say; they create and manage the appearance of the ball, paddles, and the field in the game modes
4. movePaddles(), keyPressed() and keyReleased() – these three methods work together to allow the user(s) control each paddle separately or at the same time depending on the game mode by using keys and/or the mouse
5. instructions() – this method creates text at the start of the game to notify the user(s) the controls for both game modes
6. velocity(), borders() and collisions() – these two tricky methods work all of the mathematics behind how the physics of the game behaves under certain restrictions to make the game playable and fun at the same time
7. clock() – the clock method is responsible for the clock in the top left when playing, it allows certain effects to occur throughout certain times throughout the game, for example, the instructions go away after 3 seconds
8. score() – the score method allows the user to see and keep track of the score of the game effortlessly in the 4 player mode
9. Collision() – adjusts ball velocity to simulate collision with the pad, while copying brickbreaker physics; hitting the edge of the paddle creates a faster rebound speed in the direction of that corner (ex: if the ball is coming from right and hits the left corner it will go left after collision)
10. end() – this method allows the user to restart the game or return to the main screen
11. limiter() – sets limits to how fast or slow the ball can go
12. titleText() – creates main text banner of main screen – made separate since it was complicated to make it arch around the ball

**How program runs generally:**

As mentioned above, the program generally runs by using mouse coordinate locations and a set of Booleans to direct the user into the selected game mode. In addition, several methods are added to import and draw media for aesthetic purposes and make the game easier to navigate. From there, a clock is made by referring to the computers time when the selected game mode starts, counting down up to the start of the game. Then, several controls are created by referring to keyPressed and key Released functions to allow objects to move at the users will. Then, many if statements are used to create borders and restrictions on the game to control the physics of it and make it playable. Finally, an extra feature at the end recognizes if the ball has left the screen and the game is over, both changing the score and giving the user the option to continue playing or not.

**Restrictions:**

The program has a restriction of having an invariable same screen size and having its controls still awkwardly close together making it a bit difficult for 3 people to share a keyboard and the person with the mouse to still see what’s going on.

**Major Problems/Flaws:**

Everything in the game seems to work perfect except for the scoreboard. The scoreboard was difficult to code so that only one point is rewarded or deducted when a goal is scored since the draw() method constantly loops, therefore, occasionally multiple points are rewarded depending on the velocity of the ball as it crosses the goal line; a slower velocity allows the loop to count the ball in the ‘goal’ position many times as it is passing it, thus counting multiple goals instead of 1. Additionally, the user with the mouse may have a slight advantage since he can move a bit quicker with more precision. Otherwise the game is quite consistent and solid after hours of testing.

**Next Steps:**

What I would have changed next time would be to make more classes and/or use multiple .pde files which would allow my code to look more neat and easy to navigate through. In addition, I would have added powerups and different themes like hockey or basketball to add an extra element of surprise and fun to the game. Finally, making the games a fixed length would have been useful as an additional mode where each game is first to 7 points and the series is a best of 7 (plus sudden death rounds).