Modified Project Explanation

My modified project using Java and Processing is based on the “Capture the Flag [Fixed Maps]” project from openprocessing.org. In the original game, two players could play, one with the W, A, S, and D keys and another player with the Arrow keys. The game plays like capture the flag and you can capture the enemy player if they are on your side of the field. Additionally, the map could be changed by pressing the space bar.

To start the modification, I fist needed to translate the original project from JavaScript to Java. This was mostly easy because both languages have object-oriented concepts. The issue came when converting variables. In JavaScript, variable types are automatically set whereas they are explicitly defined in Java. I needed to determine what type of variable each thing was especially when deciding to have integers or floats. This issue persisted with the array handling. The original JavaScript code had an array containing both elements of obstacles and the players but in Java, this would require a common parent class. I attempted to create this through the “collider” class but it was found that the easier option was to simply handle player collision separately from obstacle collision.

Moving on to the observable modifications, I switched from Processing’s default renderer to the Java FX2D renderer and improved performance to 60 frames per second. This makes the players move faster and leads to an overall faster game. Then, I adapted code from another project on OpenProcessing to run a countdown at the start of the game, which the original did not do at all. I added another map to the game to give it more variety. In the original game, changing the map was sudden and immediate. I modified the obstacle system so that the map will change by having obstacles move to their position and change sizes so that they reach their target coordinates. I also made the map change run at random intervals between 10 and 20 seconds in a match as well as changing to a new map at the start of each game. This meant that the functionality to use the space bar was disabled. The next key modification was the use of particles. Simple circles to show something significant has happened. Particles were designed to play at the start of the game, when a player is captured, and when a player wins the game. In the original game, the game ended when one player had 10 wins and displayed the score as a number above each player’s spawn. In the modified copy, the game runs as a best 3 out of 5 and the score is shown using circles above the player’s spawn points. Finally, in the original, when a player won the game simply read “Player # Won!” where # is 0 for red and 1 for blue. Now, once a player reaches 3 wins the game will fade to black and the text “Red Wins” or “Blue Wins” will appear in the colour of that team. Additionally, particles will fall like fireworks on each side of the text in the colour of the winning team. Lastly, a “Play Again” Button is available for the players to start another game without re-running the program. There was a plan to include an “Exit” button but there was an issue with calling exit() in the code which would crash the renderer.

Some of the issues encountered in this modification process include issues mapping JavaScript functions to their Java counter parts. This was seen when the original program wanted to call a function “isKeyPressed(int keycode)” and “collideRect()” neither of which exist by default in Java Processing. To account for this, I had to custom implement the function isKeyPressed by mapping information from keyPress and keyRelease in a meaningful way so that we know if a key has been pressed. collideRect was also implemented by simply recreating the original functionality. One of the early bugs was that the red player could not collide with the top arena wall and the blue player could not collide with the bottom player wall. This issue was narrowed down to a check where the red player could not collide with wall 0 and the blue player could not collide with wall 1. This was due to an issue in the original JavaScript translation. Another issue was times when code was called twice. Originally, each cube is checked for collision and rendered for each player. By moving the rendering so it only executes once, performance is saved because there is not a second rendering attempt. Lastly, I was also able to optimize player performance to draw only one rectangle with a border instead of 2 individual boxes.

All of this performance gain has significantly improved the speed of the game and responsiveness of the players. That and the visual upgrades help this game feel like the original only with more developed graphics and functionality.