Name: Chan Guan Yu

Student ID: 30666503

Assignment 1 – Report

The SVG elements are created using the namespace of the canvas. For each SVG element, it contains its own attributes which are set using Object.entries, in a key-value pair.

There are states used, which are the velocity of the ball, the scores of player and AI and the state of the game (whether it has ended or not).

The gameplay relies on the mouse, which is used to move the paddle up and down, by using fromEvent<MouseEvent>, it is able to detect the mouse movements. In this case, only up and down movements are allowed. The mouse event also uses pipe to filter and map, ensuring that the paddle of the player does not move out of bounds.

For the movement of the ball, it is subscribed into the main interval of the game pongObs, and that it has a fixed speed with a velocity that changes depending on different cases such as when the direction needs to be changed. Of course, the ball needs to be checked on whether it has collided with the ceiling and floor of the canvas, which is done using the conditions in the filter function.

For the movement of the AI paddle, it depends on where the ball is moving. It will move in a position where it ensures that the ball will hit approximately the centre of the paddle. To further improve the movement, if the ball is further away from the paddle, there will be a slight increase in speed to allow the AI to reach faster, similarly, if the ball is closer to the paddle, its speed will also increase but in a lesser amount.

The collision of the ball with the AI or player's paddle is similar. If the ball collides with either paddle, it will change in its x velocity, so it would simulate real world physics (though not entirely accurate), allowing it to bounce of the paddle. The velocity state of the ball would be altered.

For either AI or player, if the ball gets past the wall of either side, the opposite side scores. There are two different subscribes, one for AI and one for player, where it determines which side the ball has gone past using the if statement. Instead of using 0 and 600, -10 and 610 are used to ensure that the ball has completely gone past the walls of canvas, reducing the chances of any errors

that could occur. The ball would then be reset in its original position near the middle of the canvas, and it would travel in the direction of the side that has just lost a point. For example:

AI scores, score increases by 1, ball will now move towards player

And vice versa.

After either side reaches the winning score of 7, the game ends. The state of the game is altered, allowing the game to finish. There is also a restart option available. By restarting the game, the scores, paddles and ball are then reset to their original positions. The function restartGame is created to avoid duplication of codes, since it works in the same way for either AI or player.

A “Restart Game” will then be displayed to allow the player to restart the game by clicking on it.

I have tried to follow the FRP style, by implementing functions such as map and filter. For example:

```
filter(({ballCX, ballCY, ballR}) =>
              (ballCX - ballR > 0 || ballCX + ballR < 600) &&
              (ballCY - ballR <= 0 || ballCY + ballR >= 600))
```

Above shown is a filter function used to determine whether the ball has collided with the ceiling and floor of the canvas.

Another example:

```
const mouse = fromEvent<MouseEvent>(canvas, "mousemove")
```

Above shown is a rxjs function fromEvent to create an Observable mouse movement, which is used for the movement of the player’s paddle, allowing it to be moved up and down.