FIT2102
Assignment 1
Tan Sook Mun
30695759

Game Design

For my pong code, there are 3 functions in the pong function.
Score : keeping track of the score of the user, function
Restart: Called when the user restart the game
startGame: starting the game. Where the ball,paddle and scores are updated

First there are 2 types which are ball and state where it is the place to keep the values of the ball and the place to keep the values of the game state.
This is every useful place to store and encapsulate the values of game state and the ball state.
In the pong function it first checks if the game is over if it is then dont start the game. If the game is not over then start the game. Using the codes form tutorial i use even keys to get the mouse move and keyboard control

```
const paddle = document.getElementById("paddleUser")!;
var keyDown = fromEvent<KeyboardEvent>(document.body, 'keydown').pipe(filter(e =>[38,40].includes(e.keyCode)))
const up =keyDown.pipe(filter(e=>e.key=='ArrowUp'),map(e=>{return {x:0,y:-1}}))

keyDown.pipe(filter(e=>e.key=='ArrowDown'),map(e=>{return {x:0,y:1}}),merge(up)).subscribe(e=>{
  paddle.setAttribute('y',String(e.y + Number(paddle.getAttribute('y'))))
})
// for user mouse
const svg  = document.getElementById("canvas")!;
  // control user's paddle using mouse`
const paddleUser = document.getElementById("paddleUser")
const mouse = fromEvent<MouseEvent>(svg, "mousemove")
mouse.pipe(
filter(({x, y}) => x < 600 && y < 600),
map(({clientX, clientY}) => ({x: clientX, y: clientY-110})))
.subscribe(e => {
  paddleUser.setAttribute('y', String(e.y))
  })
```

First, get the SVG object that is the paddle user. Create a keyboard event that pipes any keypresses and filter only those that is arrowUp or arrowDown. Here i used their respective key code. Return the correct x and y values. Then map it on the canvas. I also created a mouse Event which does basically the same thing as the keyboard event but it tracks the mouse movement. For both of it i use observable to see the keypresses and mousemoves and subscribe it and change the appropriate x and y values.

My function start game hold most of where the game functionality is.
I use the function `const paddlefollow = document.getElementById()` to get all of my svg objects. I created all my objects such as ball,paddle,text in the html SVG. This is so i can reduce the lines of code and is easier to call the object rather than create it in java script.

I created a subscription that updates in interval 10 which means it updates every 10 milliseconds.
In the subscription is where the action is. I add the x and y values of the ball to make it have a smooth move. When the ball moves I check if it hit any of the walls of the canvas. If so it bounces. All this is done using if statements. I did not change it to ternary because there is no else statement. Whenever the ball hits the ball, the score is updated when it hits the left or right wall.

For the ball to detecting the paddles I check that the x values is in the same x as initialball.x + initialball.r. I add the radius because the xy values is in the middle of the ball. Also i check if the y values is in the range of the paddles.

Lastly set the attribute of the respective object using `ballmove.setAttribute()`
To make the paddle bot follow the ball i used a line of code.

```
paddlefollow.setAttribute("y",
String(Number(ballmove.getAttribute("cy"))-75))
```

I set the paddlebot y values the same as the ball. Because it can only move up and down i dont update the x value. The -75 is so the ball bounce in the missile of the paddle.
This way there is no way the computer will lose.

```
if(initialState.gameOver) {
          subscription.unsubscribe();
          initialState.scoreBot ==7?
document.getElementById("Winner").innerHTML="Bot Wins" :
document.getElementById("Winner").innerHTML="Player Wins"
          document.getElementById("GG").innerHTML ="Game Over"
          restart()
        }
```

The last part of the subscription is to check if the game is over if it is then write the winner and the text game over. The function restart will call and will wait for the user to press space to restart the game.
In the restart function, I reset the values of my ball and game state to its original values. then call start game again.

Summary

So to summarise i created types so i can store the state and ball values. This is so the values will stay consistent and encapsulated in the type. I used observables and pipe, filter and map so the game play will be consistently updated every 10 mili seconds. I follow the FRP style by using only conts , observables and as many tenary function as possible. I also use some arrow functions. The way i manage my state through out the game is putting the gameplay in the same subscription. That way it will be updated together and not separately making it smooth gameplay.