FIT 2102 Assignment 1: Functional Reactive Programming

Name: Tham Yu Le

ID: 31063489

The general overview of the code is to create a State for the game and multiple Bodies for elements in the game to store their attributes. I then use an observables to detect user inputs which translates into moving the paddle elements in the game. The whole loop of the game is ran on an interval observable to allow for natural movement of game elements. Interactions between game elements, such as collision between the ball and other Bodies and also the edges of the canvas. Doing so changes the attribute of the ball body and thus the game state, this allows us to see the changes in the UI after running the game state through the function that is subscribed to our interval observable.

I used quite a lot of higher level functions such as map, scan , filter, in my main game loop, the interval Observable in order to do further operations on multiple functions. Filter is used in the keyObservable to detect the correct key presses. these functions take in other functions and are hallmarks of functional programming, they are used in this program in order to adhere to the functional programming style. To implement the restart feature I did not unsubscribe from the main interval Observable as that will prevent my observable that is looking for the restart input to keep running, thus I stopped the ball in the middle with zero velocity until the restart button is hit and then the state will be reset to the initial state with the scores and the winning quote being reset too

I decided to implement Observable streams to detect user inputs and also service as the main loop of the game as Observables provide the benefits of functional programming such as composability and reusability. Not only that but observables are just another container of elements, but whose “push-based” architecture allows them to be used to capture asynchronous behavior which is typical of functional reactive programming. In my code I try to separate specific reusable interactions as functions, such as “ballCalculations”, “moveObj”, “paddleAi” and so on. Each of them does calculations for the ball’s velocity scale when hitting the separate parts of the paddles, moves Bodies automatically and how the Computer paddle follows the ball respectively, doing so allows me to reuse code. In line with FRP style principles, I did not declare any variables in the program, all of which that are declared are only constants and functions. And there are also no side effects and I don’t change the game State directly but make a deep copy with the new attributes and pass that as the new state. I also made sure that all my functions are pure, that they have no side effects other than to create the return value

and also always produces the same results for the same input. I have also implemented rng into the game code but it is pure as the output only changes with the seed input. With the same seed it will always output the same values in the same interval.

The way that I handled State in my programming is by first coding functions which takes in State as its parameter and does calculations on what should be changed in the state, in the return of the functions, I do a deep copy of the state with the changed elements. By doing it this way, I do not cause any side effect by changing the value of a variable declared outside the function scope. All of these functions are then called in the reduce state function. Which is called by a scan function in the interval Observable with the initial function. This serves to constantly produce a new state every interval which is then passed into the update view function in the subscribe in order to use the information present in the State to update the UI ad let the player see what is going on.