FIT2102 ASSIGNMENT 1: Functional Reactive Programming Report

Code summary

The program functions as such:

- Each interactive element e.g. the paddles and ball are objects having their own interface.
- The entire game, inclusive of interactive elements, is managed by the object inheriting from GameState interface.
- User key presses and changes in the GameState are recorded as observable streams.
- An interval is use as and observer which is subscribed to the observable streams.
- The observable streams area reduced into a single state.
- The state is passed to the updateView function that will change the visual elements based on the new GameState
- The Ball and paddles move based on adding units to their x and y position on the canvas.
- Changes in GameState is done through by deep copying the initial instance of GameState, the intialState, and then attributes are reassigned where necessary.

Interesting parts, high-level overview, and justifications.

Handling initialState, observables and observer. (FRP)

The object's that will change over time are recorded through the initialState. The initialState acts as an observable stream that allows the observer, named subscription, to keep track and model the data so that it can be updated on the view for the user to see. Note that the key press observable from the user is also handled by the initialState. By coupling all these observables changes under a single initialState, it eases the coding of functionality and reduces spaghetti code by being able to access all the observable objects as attributes of a single object. Different functionalities of the program will make different copies of the initialState. These copies will be handled by the reduceState and subscription function. These functions will merge the copies and reduce them into a single observable stream, which is taken in by the updateView function to show display the effects of the state changes.

Movement of objects.

All objects that move along the screen contains an x and y attribute that denotes their position on the canvas. Moving these objects require the changing of these attributes over time. This is done by treating the changes of the initialState as an observable stream. The paddles and the balls are of their own type, so they require their own set of functions to handle their movement.

Ball movement.

The Ball contains and additional pair of attributes, dx and dy, which represents the direction and magnitude to displace the ball's position by on its respective axis. This makes coding the movement of the ball across the canvas much easier as it separates the responsibility of moving the ball and directing the ball.

Handling collisions.

Collisions are handled by declaring the range of positions taken up by the paddle and checking if the ball's position overlaps with those positions. These collisions are broken down by the sides of the canvas: the sides which do not have a paddle and the sides which do have a paddle. A condition is set for the ball's y position that reverses its dy whenever it goes out of the range $0 < y < CanvasSize$. Another condition is used for the paddles, where the dx reverses when the ball's x position is within the reach of the paddle (the reach is from the paddle's y position up to its height) and the paddle's range (the line of x positions in front of the paddle.)

Angle Shots.

The ball's angle will change depending on how the ball hits the paddle. When the ball hits the upper part of the paddle the dy attribute will decrease by 0.5 and will increase by 0.5 if it hits the lower part of the paddle.

Tick function and automated movement.
The tick function is what allows the objects to move without user input. The tick function calls and chains the other functions that are involved in moving the paddles and the ball and passes down the state changes to the subscription function.

Maintaining function purity
The return value of the functions is heavily dependent on the dependency of arguments, so the functions are kept pure by relying only on the initialState. When the functions are called, they do not change the InitialState and instead make a new deep copy of the initialState to be returned. The same can be said for changing the attributes of initialState, copies of the paddle and balls are also returned. Globally accessible variables are kept constant as well, so these values can not be changed to affect the return value.