Frogger with FRP – Assignment 1 report
Name: Kang Zheng Rong
Student ID: 32797990

This Frogger game is completed with functional reactive programming and rxjs library. Observables are used to ease the asynchronous behaviour and to handle the transformation of objects in the game.

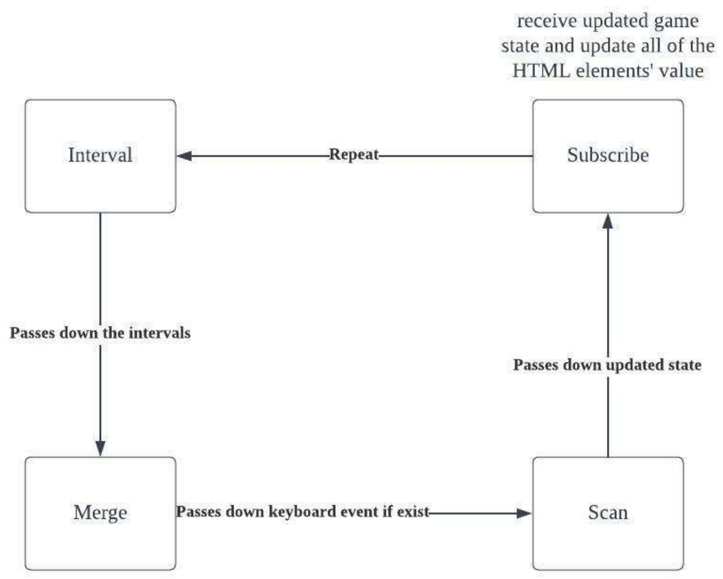
Before starting everything, we need to separate the canvas into ground and river section. Next, we are going to create all the obstacles, the main character "frogger" and the in-game text. These need to be crated before hand for our code to run and change their state.

Purity

updateGame will be the only function that is impure in our code. It only executes in subscription. We know that pure functions will always output the same result if we pass in the same parameter and nothing outside the scope will be affected. For all of the function except **updateGame**, they will deal with the game state and does not access any global values or perform any side effects.

Observable Stream

The observable stream that I have coded are the **game** in line 289. It is the main component that asynchronously update the frogger game. The interval will use pipe to transform elements from one container to the other. Firstly, the stream will merge with up$, down$, left$, right$, these are keyboard event that will eventually return a Move class object. This object consists of changes to x and y position. After that, the stream will use scan. Scan is an accumulator function which takes in initialGame and executes reduceState that computes the changes to the current state of frogger and returns a new copy that included all the changes. While computing the changes I have also added constraint to the frogger which prevent the frogger to jump out of the canvas. Since scan always return a new state, there is no occurrence of mutability.


receive updated game state and update all of the HTML elements' value
Interval
Repeat
Subscribe
Passes down the intervals
Passes down updated state
Merge
Passes down keyboard event if exist
Scan

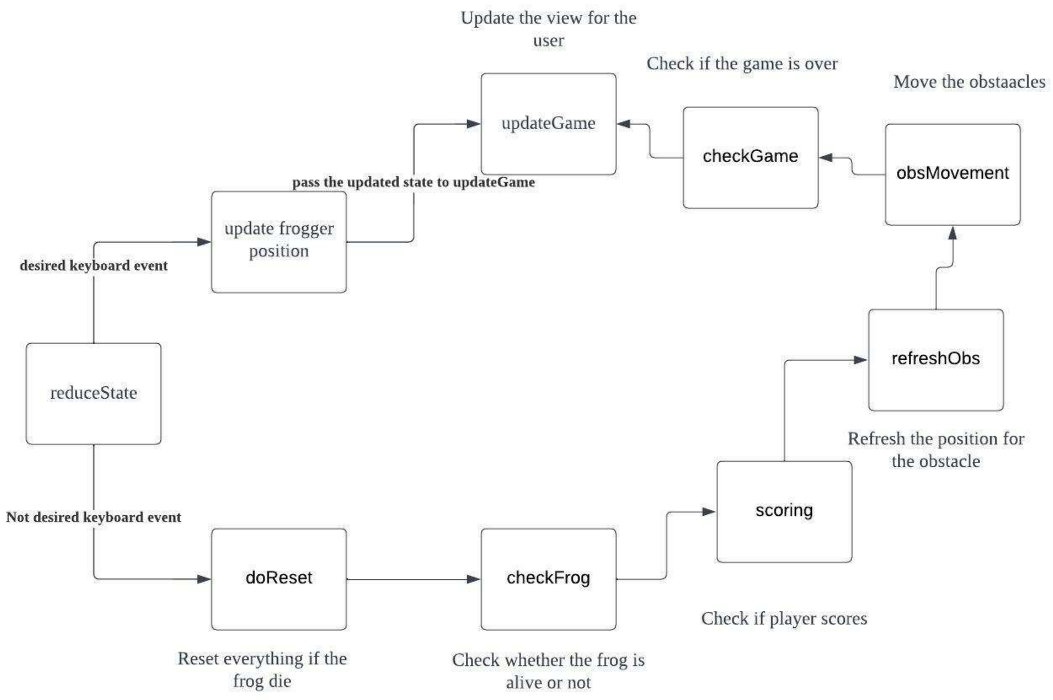
Frogger with FRP – Assignment 1 report
Name: Kang Zheng Rong
Student ID: 32797990

<u>ReduceState & updateGame</u>

The only way to manipulate the values of x and y position of frogger in the state is executing reduceState with scan. reduceState will accept parameters of (s, m) where s is of type State and m is of Move object. The game state that is passes into reduceState is the previous state that have values that needed to be updated. If a desired keyboard event is detected it will create a Move object. If m is an instance of Move, we will make changes to frogger x and y position with the values in m. Otherwise, reduceState will perform a nested function that is not control by the player.

This nested function calls checkGame, obsMovement, refreshObs, scoring, checkFrog and doReset. Each of this function will take in a state and return a new state with some modified values based on the function. These function does not have any side effects and mutation which allow them to be a pure function while be able to change the values of attributes.

After scan is done, the final updated state will be passed into the updateGame in the subscription. Finally, all of the respective HTML elements will be modified and all of the mutable actions will be contained within the subscription to prevent any side effects.


Update the view for the user
updateGame
Check if the game is over
checkGame
Move the obstaacles
obsMovement
pass the updated state to updateGame
update frogger position
desired keyboard event
reduceState
refreshObs
Refresh the position for the obstacle
Not desired keyboard event
scoring
doReset
checkFrog
Check if player scores
Reset everything if the frog die
Check whether the frog is alive or not

All of the function in the nested function are straight forward except checkFrog as we are checking for collision there. In the checkFrog function we will first determine if the frog is on the ground or river. Then, check is it on a log. Finally check if frog collides with the car. If any collision happens, we will reset the frog position to initial.

Frogger with FRP – Assignment 1 report
Name: Kang Zheng Rong
Student ID: 32797990

High Score

We will be able to keep track of the high score by adding it to the game state. It will not be change back to its initial value when the game is reset. Thus, the high score can be tracked. It is done in doReset function.

Game Design

All of the movement will be done by arrow keys.

Objective is to control the frogger and reach the opposite site.

Player score every time they reach opposite site.

While completing the objective we must avoid all possible collision.

The speed of the obstacles will increase every time we reach the opposite site.