FIT2102 – Assignment 1 Report

Space Invaders Game

Name: Nadeem Emadeldin Hamed Hamed Abdelkader
Student ID: 30146224

Introduction

A classic Space Invaders game performed with the help of TypeScript and RxJS. The main goal of the game is to demonstrate Functional Reactive Programming.

How to Run

1. Unzip the folder
2. Open terminal/cmd inside the unzipped folder
3. Type npm install in terminal/cmd
4. Type npm run build in cmd
5. Open the spaceinvaders.html file in any browser

Summary of the workings of the code

Main Idea

The basic idea is that we implement MVC. This means we have a Model, View and Controller. In more details, we have a user who communicates with our View, then we have a Controller whose goal is to handle User Input, then the data has to be processed in a certain way and sent to a Model who has to save it as a state and update View.

How is it implemented?

The application is divided into several parts.

In the first part we get our dependencies. In our case, these are HTML Elements.

The second part is the one in which we attach event listeners for the elements through which the user can enter data. In this way we can intercept them, process them and send them to the store.

The third part is that we need some state that will store the data that the user enters. For this purpose we use the RxJS library, with the help of which we can create a similarity of a store, and accordingly when the user enters something, we can easily save it in a store and send it to our View.

Accordingly, the next part is the one in which we subscribes for the store, created above, and when we have new data then we update View.

High Level Overview of Design Decisions and Justification

Everything described above is for the purpose of performing Functional reactive programming (FRP). This means that we don't just want working code, we want transparent , easy-to- read and easily extensible code. This allows us to easily develop our projects in the future, easily fix code problems, etc.

How the Functional Reactive Style (FRP) was Followed

The Functional Reactive Programming style allows us to capture asynchronous actions like user interface events in streams. These allow us to “linearise” the flow of control, avoid deeply nested loops, and process the stream with pure, referentially transparent functions.

The Functional Reactive style was used with the help of rxjs’s operators (fromEvent, interval, merge, map, filter, and scan) to create pure observable streams. Which is like a push based lazy sequence.

The main advantages of such streams are as follows:

- “Applications of functions to elements of containers to transform to a new container (e.g. map, filter, reduce etc. over arrays).”
- “Use of function composition and higher-order functions to define complex transformations from simple, reusable function elements.”

How Was State Managed Throughout the Game

I defined an interface for State with readonly members and placed our initialState in a const variable that matches this interface.

```
// Game state
type State = Readonly<{
  time: number;
  ship: Body;
  bullets: ReadonlyArray<Body>;
  aliens: ReadonlyArray<Body>;
  exit: ReadonlyArray<Body>;
  objCount: number;
  gameOver: boolean;
  win: boolean;
}>;
```

```
// Initial state of the game
initialState: State = {
  time: 0,
  ship: createShip(),
  bullets: [],
  aliens: startAliens,
  exit: [],
  objCount: Constants.StartAlienCount,
  gameOver: false,
  win: false,
},
```

The state was incremented all over the program, so that at any given point it will hold the current time, the ship, an array of bullets, an array of aliens, an array of exited bullets, the object count and 2 Boolean values to indicate if the user has won or if the game has ended.

Future Work/Potential Improvements

The obvious improvement is adding the functionality for aliens to fire bullets at user's spaceship, which unfortunately I did not have enough time to implement.

Some other improvements in mind:

- Adding levels
- Adding score multiplier for streaks
- Ability to restart the game (can only be done by refreshing the page right now)
- Working Shields (they are not functional right now)

References

- https://tgdwyer.github.io/functionalreactiveprogramming/
- https://tgdwyer.github.io/asteroids/