FIT2102 Programming Paradigms 2022

Assignment 1: Functional Reactive Programming

Due Date: 09/09/2022
Weighting: 30% of your final mark for the unit

Overview. Students will work **independently** to create a classic arcade game using Functional Reactive Programming (FRP) techniques. Programs will be implemented in TypeScript and will use RxJS Observable streams to handle animation, user interaction, and other similar stream behaviours. **The goal is to demonstrate a good understanding of functional programming techniques as explored in the first five weeks of the unit**, including written documentation of the design decisions and features.

Submission instructions

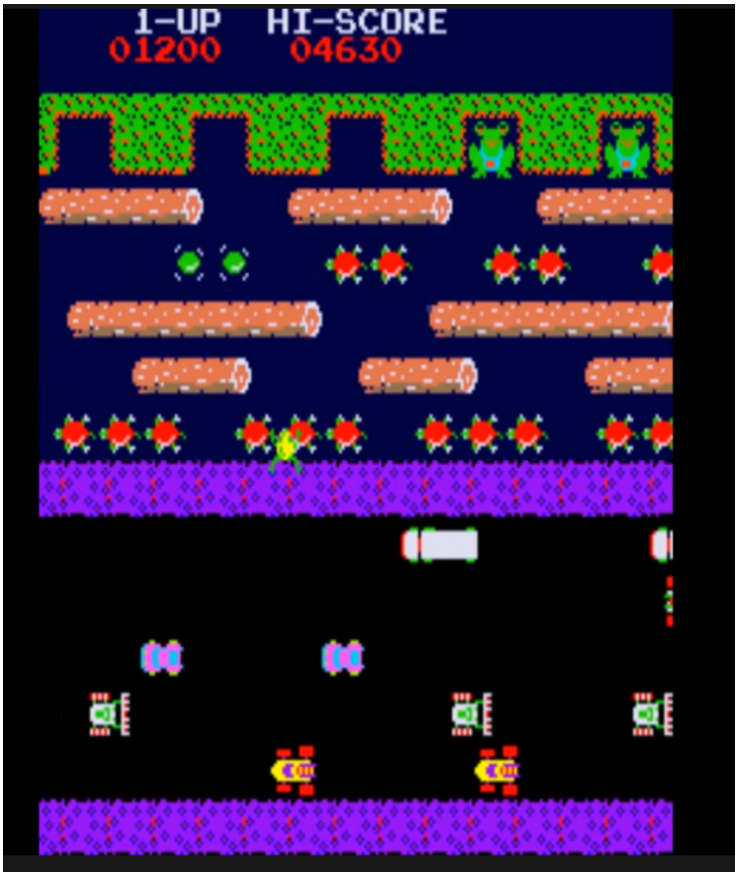
Submit a zipped file named <studentNo>_<name>.zip which extracts to a folder named <studentNo>_<name>

- **It must contain all the code for your program** along with all the supporting files as well as the **report**.
- It should include sufficient **documentation** that we can appreciate everything you have done.
- You also need to include a report describing your design decisions. The report must be named **<studentNo>_<name>.pdf**.
- The only external library should be RxJS libraries supplied with the starter code.
- **Make sure the code you submit executes properly.**
- **Do not submit the node_modules or dist folder.**

The marking process will look something like this:

1. Extract **<studentNo>_<name>.zip**
2. Navigate into the folder named **<studentNo>_<name>**
3. Execute `npm run build`
4. Open `index.html` in a browser

Please ensure that you test this process before submitting. Any issues during this process will make your marker unhappy, and in some cases, may result in a deduction in marks. **Failure to follow these instructions may also result in a deduction.**



Task description

In this assignment, we will use the RxJS Observable stream explored in the Week 4 worksheet to create the [classic Frogger Arcade Game (YouTube)] in an SVG image hosted in the **index.html** webpage.

The YouTube video is meant to give you an idea of the gameplay, but yours needn't look the same or work in precisely the same way, especially with regard to graphics. **Note that only a subset of the features shown in the video will be part of the requirements.**

You will also need to write a report detailing the design of your game.

Requirements

The game must be implemented in a good functional reactive programming style to get marks. A subset of the game's features will be required to get a passing grade. Additional features will be required to get a higher grade. To achieve the maximum marks for this assignment, you will have to use a little creativity and add some non-trivial functionality of your own choice – see the additional information document for some ideas.

Correct collision behaviour:

- In the **ground** section, Frog may move/stand on the **ground**, but dies when colliding with a car object
- In the **river** section, Frog may move/stand on **plank objects**, but dies when landing in the water (ground)
- Frog dies when colliding with any enemies (e.g. snakes, crocodiles)

Minimum requirements

All of these requirements must be reasonably executed to achieve a passing grade

- Frog which can move forwards, backwards, left, and right using one of the keyboard or mouse
- **Multiple rows** of objects **(at least 6)** appear and move across the screen
- Objects move at **different** speeds and directions (left-right)
- **Correct collision behaviour** (defined above) including at least one **ground** section and one **river** section
    - For minimum requirements, you do not need to include enemies
- Game ends when the Frog dies
- Indicate the score for the player
- Player scores points by landing the Frog in a **distinct** target area
- A 1-2 page PDF report detailing your design decisions and use of functional programming techniques discussed in the course notes

Full Game requirements

Meets minimum requirements and has additional features

- Keeps track of high score achieved across previous rounds
- **Multiple** distinct target areas that must be “filled” (as per the video)
- At least **3 distinct objects** with different interactions/behaviours (e.g. crocodile, turtle, car/plank, snake, fly) that aren’t just movement
    - Cars and planks count as one distinct object
    - An example might be cars and planks, crocodiles, and turtles
- At least one of the “ground” and “water” sections, including at least one row in the middle where there are no objects (safe zone)
- Smooth and usable game play.
- **Able to restart when game finishes**
    - This must not be done by refreshing the page, and should also not be done by recursively calling the main function (you should use state management to handle this)
- The game increases in difficulty after some **non-score-based** condition is met (for example, landing the frog in 5 target areas)
- *See video for an idea of appropriate gameplay*

Report

Your report should be 600-1200 words in length, plus up to 600 words for each significant additional feature, where you should:

- Include basic report formatting headings/paragraphs
- Include diagrams as necessary
- **Summarise** the workings of the code and **highlight** the interesting parts (don’t just describe what the code does, we can read the source code!)
- Give a high level overview of your **design decisions** and **justification**
- Explain how the code follows FRP style
- How state is managed throughout the game while maintaining purity
- Describe the usage of Observable beyond simple input
- **Important**: Need to explain **why** you did things
- **Do not include screenshots of code unless you have an exceptionally good reason**

We will be fairly lenient with word count, but excessively long reports may be penalised.

Plagiarism

We will be checking your code against the rest of the class and the internet using a plagiarism checker. Monash applies strict penalties to students who are found to have committed plagiarism.

Additional information

Some additional (not essential or required) information is also available on the Moodle Assessments section. This is used purely to provide context and answer questions students may have, and is not necessary to complete the assignment.