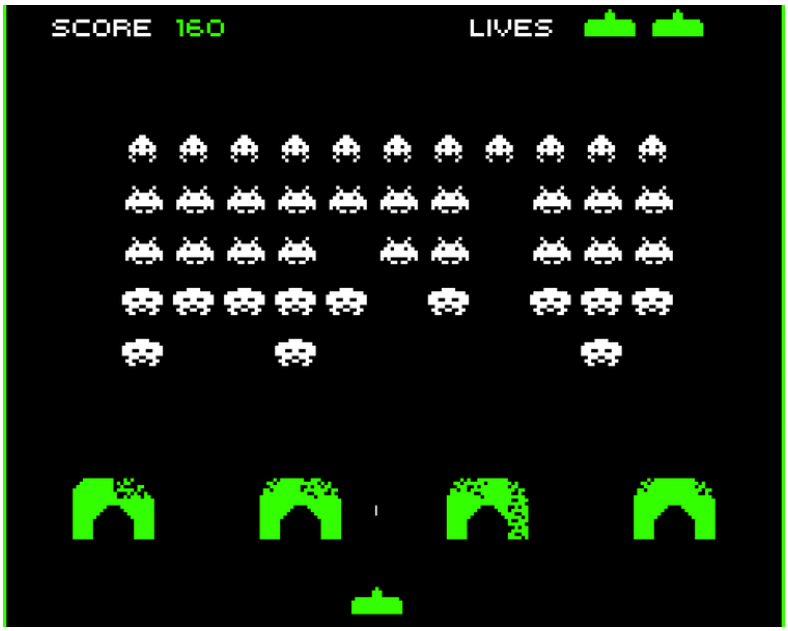
FIT2102 Programming Paradigms 2021

Assignment 1: Functional Reactive Programming

Due Date: 10/09/2021
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 and user interaction. Programs should demonstrate a good understanding of functional programming techniques as explored in the first five weeks of the course and should include written documentation of the design and features.

Submission instructions. A .zip file named studentNo_name.zip which should extract to a folder named studentNo_name. Do not submit the node_modules or dist folder. It should 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.**



Task description

In this assignment we will use the RxJS Observable stream explored in the Week 4 worksheet to create the classic Space Invaders game (YouTube) in an SVG image hosted in the **spaceInvaders.html** webpage. The YouTube video is meant to give you an idea of the basic gameplay, but yours needn't look the same or work in precisely the same way. You will also need to write a report detailing the design of your game. The baseline functionality required for a passing grade, what needs to be included in your report and a list of alternative ideas for achieving an HD, are listed below.

Minimum requirements. All of these requirements must be reasonably executed to achieve a passing grade (detailed marking rubric below).

- Implement a one-player Space Invaders game with a spaceship movable by mouse (while the mouse cursor is over the svg canvas) or by keyboard.
- Rows of aliens should appear and move across and down the screen.
- The aliens should fire bullets at the user. If the user is hit by one bullet, the user will die and the game will end
- Indicate the score for the player.
- A 1-2 page PDF report detailing your design decisions and use of functional programming techniques discussed in the course notes.

Full Game:

- Meets minimum requirements
- The user has shields, which disintegrate over collisions
- Smooth and usable game play.
- Able to restart when game finishes
- The game progresses to a new level after all aliens are shot
- *See video for an idea of appropriate gameplay*

All the above need to be implemented in a good functional reactive programming style to get a good grade. To get a higher grade you have to use a little creativity and add some functionality of your own choice -- suggestions below.

Ideas for getting an HD. Any one or more of the following (or something of your own devising with a similar degree of complexity) done well (on top of the basic functionality described above) will earn you an HD provided it is implemented using the functional programming ideas we have covered in lectures and tutes:

- Create unit tests and create a file **tests/main.test.js**
- Incorporate gameplay from other classic arcade games -- e.g., breakout, galaga, etc.

- Add power-ups to the game -- larger shields, slower aliens, faster spaceship, etc.
- Add combination bonuses, e.g if you hit 5 aliens in a row.
- Advanced (not recommended unless you already know how): Make a distributed multiplayer version, wrapping the comms in Observable (you'll have to provide your own server for this).
- Your own ideas!

Some of the above will require a little independent research. That's what computer science is all about!

Report. Your report should be 1- 2 pages in length, plus up to one page per extension, where you should:

- Include basic report formatting headings/paragraphs
- Include diagrams as necessary.
- Summarise the workings of the code and highlight the interesting parts.
- Give a high level overview of your design decisions and justification.
- Explain how you tried to follow the FRP style.
- How you manage state throughout your game.

Marking. The goal of this assignment is to assess your understanding of FRP. The marking is done in two parts.

1. Implementation of game features.
2. Use and understanding of proper functional programming style.

The idea here is that adding features to Space Invaders will grant you a higher grade *under the condition* that it is done in proper Functional and FRP style. For an example of the proper style, refer to the example Asteroids Game described in the Course Notes. Code that does not use Observable will not get a passing grade; code which relies on imperative code will be penalised. To achieve a mark in the HD range you need to implement a complete Space Invaders game with good style and a choice few additional features, as above.

Additional information

Similar to the tutorial exercises we will provide you with a starter project which you can download in a zip file from Moodle that contains:

- **spaceInvaders.html** currently just an empty SVG object. Your Space Invaders game goes here! Add instruction text to the page to explain how to play your Space Invaders game. Make sure the instructions are sufficient that we can properly try out all of your features.
- **spaceInvaders.ts** source code for your Space Invaders game. This is where most of your work will go.

Tips for getting started.

- Complete the Week 4 Tutorial Worksheet Observable exercises and begin studying Observable in the course notes.
- In week 5 Tutorial you will complete **observableexamples.ts**
- Once you have completed the above, work through the example Asteroids Game described in the Course Notes. Follow the same framework to begin adding functionality to **spaceInvaders.ts** as above.

More tips.

- Finish all the JavaScript and TypeScript tutes (up to the first part of Week 5) and the course notes FRP material first. They are designed to give you the experience you need to prepare for this assignment.
- Come to the workshops and tutes for important tips and assistance.
- Attend consultations given by the teaching team.
- Any general questions should be directed to the Ed forums when possible. However, try to avoid posting potential solutions. If you cannot make the consultations, you may make a ***private*** post with Staff on the assignment with code.
- Your code should include brief comments to explain logic and design choices where necessary or to refer to detailed explanations in your report. Please do not add comments that are self evident from the code, e.g.:
  - `const x = 1; // variable x is set to 1.`
- **Start as soon as possible**. Do not leave the assignment until it's too late.

Recommended coding practises.

- Structure your program in a consistent and coherent manner (group relevant functions, declarations, and variables together)
- Use block/section comments to clearly layout each part of your code
- Use nice indenting and formatting (can look into formatters if you wish prettier beautify)
- Use camelCase for names and UPPER_CASE for constants

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.

Marking rubric

It is important to realise that, as stated above:

- If you implement the **Minimum requirements** above, demonstrating application of functional programming ideas from our lectures and tutes, you will achieve a pass grade.
- You can receive up to a D for perfectly implementing the **Minimum requirements**, demonstrating a thorough understanding of how to use Observable to write clean, clear functional UI code.
- To achieve a HD you will need to implement the **Full game requirements**

- To achieve the maximum possible marks you will need to implement the full game requirements plus some aspect of **additional functionality** as described above.

Rubric table.

Code/Report quality	**Space Invaders implementation**		
	Minimum requirements	*Full game*	*Full game + extension(s)*
Any of the following are not acceptable: - Use of imperative code - TypeScript compile errors - `any` types - Not using rx.js - No comments - Missing or unreadable report - Missing instructions for how to play the game	Not passing.	Not passing.	Not passing.
Pure functional code (except in `subscribe` handlers), no compile errors, basic comments, basic report covering the implemented features.	P	C	C
The code leverages Observable, has generic types, and side-effects are identified; comments only describe the code. In addition to covering the implemented features, the report demonstrates basic understanding of FRP principles.	C	D	D
Small pure functions, immutable data and reusable code exploiting parametric polymorphism, side-effects are contained; complete comments explaining the rationale and choices for the code. Detailed report of implemented features that demonstrates strong understanding of Functional Programming and FRP.	D	HD (80-90)	HD (90+)