FIT2102 Programming Paradigms 2019

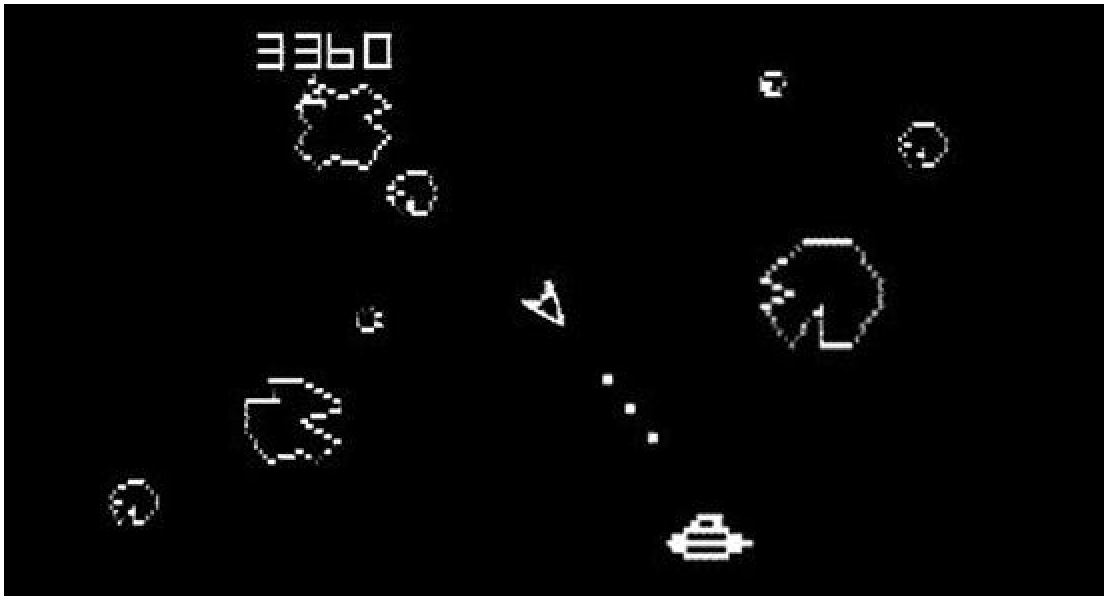
Assignment 1: Functional Reactive Programming

Due Date: Friday 13th September 2019 at 11:55pm

Weighting: 20% of your final mark for the unit

Submission Instructions: A zip file should be submitted via moodle by the due date. It should contain all the code for your program along with all the supporting files. To mark we will load asteroids.html into the browser and we will look at the asteroids.ts source code file. Make sure the html file contains instructions on how to play the game and for us to find all the features you have implemented. Make sure the asteroids.ts file contains extensive comments at the top detailing your design, that the code is thoroughly commented throughout, and also make it clear if we should look in any other files for additional work you have done, as discussed below. It should not include any external libraries (other than the test framework libraries supplied with the starter code).

Task Description: In this assignment we will use the Observable stream created for the Week 4 worksheet to create the classic Asteroids game (video) in an SVG image hosted in the asteroids.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. The baseline functionality required for a passing grade, 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 basic one-player Asteroids game, with a ship movable by mouse or keyboard -- direct movement is fine for the most basic implementation.
- Use basic SVG ellipse, rect or polygon elements for the ship and the asteroids.
- The ship should be able to shoot and destroy asteroids.
- A collision between an asteroid and the ship ends the game.
- Have proper wrapping around the edges, i.e. when an object goes off the right side, it appears again on the left, vice-versa and similarly top and bottom
(note: the map is a torus topology!).

All the above need to be implemented in a good Functional Reactive Programming (FRP) style to get a passing grade. To get a higher grade you have to use a little creativity and add some functionality of your own choice -- suggestions below.

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

1. Asteroids features implemented.
2. Use of proper FRP style.

The idea here is that adding features to Asteroids will grant you a higher grade *under the condition* that it is done in proper FRP style. Code that does not use Observable will not get a passing grade; code which relies on imperative code will not get an HD. As a rule of thumb, not using Observable will not give you marks, using imperative code would give you half marks. To achieve a mark in the HD range you need to implement a complete Asteroids game (see features) and a choice few additional features.

Asteroids features: 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 extra marks provided it is implemented using the functional programming ideas we have covered in lectures and tutes. To have a complete game, you will need to implement features such as:

- Big asteroids break down into smaller asteroids and only smaller asteroids can be destroyed.
- Have multiple levels, a score, a number of lives, etc.
- Change the ship to use “Thrust” instead of direct movement.

And, extension features (if you are shooting for HD90+):

- Add additional enemies which can target the ship.
- Extend Observable with additional methods that help to make your code cleaner or more “fluent.” This means you code *needs to use the features you implement*. (See Rx.js for ideas.)
- Add error handling to Observer/Observable (as per ES.Next proposal)
- Create unit tests to tests/main.test.js
- Incorporate gameplay from other classic arcade games: e.g. breakout, galaga, etc.

- Your own ideas!

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

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:

- **observable.ts** Contains the same Observable implementation as the Week 4 tute. You will use, and possibly extend, the given Observable for this assignment. The mocha tests provided in checklist.html for the existing functionality must still pass!
- **asteroids.html** currently just an empty SVG object. Your Asteroids game goes here! Add instruction text to the page to explain how to play your game. Make sure the instructions are sufficient that we can properly try out all of your features.
- **asteroids.ts** source code for your Asteroids game. This is where most of your work will go.
- **svgelement.ts** contains a little helper class for working with SVG elements. This file should let you do everything with SVG necessary to complete the minimum requirements. Feel free to add additional helper code here, but be sure to document it.

Tips for getting started:

- Complete the Week 4 Tutorial Worksheet Observable exercises and begin studying Observable in the course notes.

More Tips:

- Finish all the javascript and typescript tutes first (weeks 1-4). They are designed to give you the experience you need to prepare you for this assignment.
- Come to the lectures and tutes for important tips and assistance. You'll need to anyway to get the participation marks.
- Come see the Lecturer and the tutors in their consulting times. We try to get to your emails but we are flooded.
- **Start as soon as possible**. Don't leave it until it's too late.

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 HD and up you will need to do the above plus some aspect of additional functionality as described above.

Your submission will be marked on the following aspects:
- FRP style -- pure functions, re-usability, generic types.
- Use of Observable -- restrict side-effects to subscribe(), higher-order functions.
- General code quality and readability, especially comments.

Remember: a fully imperative implementation will not get a Passing grade!

The table below provides guidelines for what you can expect given a perfect execution of what is reported in column Code Quality. In general, better code quality will lead to better marks rather than having more features.

Code Quality	**Asteroids Implementation**		
	Minimum Reqs	*Full Game*	*Full Game + Extension(s)*
Use of some imperative code, <any> types, no comments.	Not passing.	P	P
The code leverages basic Observable, has generic types, and side-effects are identified; comments only describe the code.	C	D	D
Small pure functions and reusable code, side-effects are contained; complete comments explaining the rationale and choices for the code.	D	HD (80-90)	HD (90+)