

FIT2102 Programming Paradigms 2020

Assignment 1: Functional Reactive Programming

Due Date: 18th September 2020, 11:55pm

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.



Task description

In this assignment we will use the RxJS Observable stream explored in the Week 4 worksheet to create the [classic pong game \(YouTube\)](#) in an SVG image hosted in the [pong.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 basic one-player pong game with a paddle movable by mouse or keyboard while the mouse cursor is over the svg canvas. The second paddle should follow the ball.
- The ball should bounce at an appropriate angle and position when hitting the paddle. When the ball *hits the wall* of a player, the opponent scores one point.
- Indicate the scores for each player.
- The game should end when a player reaches 7 points and it should indicate the winner.
- 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
- Changing y-velocity of ball depending on what part of a paddle the ball strikes
- Smooth and usable game play.
- Able to restart when game finishes
- See [video](#) for appropriate gameplay

All the above need to be implemented in a good functional reactive programming 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.

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 paddle, faster ball, etc.

- Action Replay: after a goal replay the last 5 seconds (maybe in slow-motion) for the user to watch
- 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 one page per extension, where you should:

- 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 Pong 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 Pong 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:

- **pong.html** currently just an empty SVG object. Your pong game goes here! Add instruction text to the page to explain how to play your pong game. Make sure the instructions are sufficient that we can properly try out all of your features.
- **pong.ts** source code for your pong 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**

- The final task in **observableexamples.ts** was to implement the function `keyboardObservable()`. Once you have this working, use similar ideas to begin adding functionality to `pong.ts` as above.

More tips.

- Finish all the JavaScript and TypeScript tutes (up to the first part of Week 5) 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.:
 - `let x = 1; // variable x is set to 1.`
- **Start as soon as possible.** Do not leave the assignment 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.

Rubric table.

Code/Report quality	Pong implementation		
	<i>Minimum requirements</i>	<i>Full game</i>	<i>Full game + extension(s)</i>
Use of some imperative code, `any` types, no comments. Basic report with minimal detail.	Not passing.	P	P

The code leverages basic Observable, has generic types, and side-effects are identified; comments only describe the code. Report shows basic understanding.	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 that demonstrates clear understanding of Functional Programming and FRP.	D	HD (80-90)	HD (90+)