CGRA151 2019 Final Project: a simple game or interactive artwork

The specification

You will write a computer game or "interactive art work" of your own design. In this specification we use "game" to mean either.

Learning objectives

To learn how to manage a substantial programming project. To experience the challenges of game design, including game-play, visual design, interaction, and graphics.

Submission part 1 — a plan (due 3 September) — 2 marks

You must submit a **one-page plan** for what you intend to do. This is so that your tutor can discuss your plan with you in Week 7 or 8 and advise you on whether you have scoped the project correctly. They will check whether you are being over-ambitious (i.e., spending a lot of time for no extra marks) or under-ambitious (i.e., specifying something that will score poorly).

Your plan should be completed *using the template available on the website* and it should cover the following points:

Vision: what you intend to produce. Three paragraphs. One on each of the following:

1. **Game concept**: An overview of the game.
2. **Game play**: The design of the interaction that you plan to have.
3. **Visual design**: The design of the graphical style and colour scheme you plan to use.

Timetable: key milestones in the development of the project. You should spend about 12 hours on this project spread over the three weeks from 17 September to 8 October. For your plan, we require that you have three milestones in this time:

24 September: **The core of the program**. All the critical components will work, at least roughly. You could hand this in as your submission, if you really had to. In your plan, you should specify what you will definitely get working by this date.

1 October: **A reasonable submission**. Everything should be working, and you should be happy to submit this. In your plan, you should specify what such a "reasonable submission" would consist of.

8 October: **A well-polished submission**. The final week should be spent polishing your reasonable submission into something better. In your plan, you should specify what features of the program would be considered "polish". You might like to suggest a couple of extensions that you will not commit to producing but that you could attempt to implement if you have time.

Your game does not have to be original, but your programming must be your own. You should use existing games, artwork, and designs as inspiration. You may submit up to nine pages of sketches, images, screenshots, and notes as appendices to your one-page plan, which can demonstrate either early ideas or sources that you are using for inspiration.
If you use appendices, please submit a PDF file. If not, a one-page text file is fine.

Submission part 2 — the program (due 8 October) — 14 marks

You will submit both a **Processing sketch** and a **single-page text document *or* single page PDF document** reflecting on your experience (*use the template on the course website*).

Your game will be a single Processing sketch. It may require several tabs, which are stored as separate files in a single sub-directory. To submit, you need to use the zip utility to pack that directory into a single zip file. For example, if you name your sketch *ProjectSketch*, you could run this command to make the zip file:

```
        zip –r ProjectSubmission.zip ProjectSketch
```

The "`–r`" option tells zip to include all of the files inside the directories. The file that you submit is `ProjectSubmission.zip`. See the course website for details of how to use the submission system.

Your sketches must run on the version of Processing used on the ECS machines without needing to be modified.

Marking will be done by tutors in one-on-one sessions with you during your usual Lab time in Week 12 or 13.

Requirements

You will write a computer game or "interactive art work". It needs to have the following features:

- Written in Processing.
- Will run in the version of Processing installed on the ECS computers.
- Uses 2D computer graphics.
- Has user interaction (that is, the user must interact with the program which affects the game in some way, through the mouse or keyboard or both).
- Has graphical objects that interact with one another in some way (for example, the ball interacting with blocks from Assignment 2 Completion).
- Has at least two "levels" or some other way to reset the game to a different configuration. This is to demonstrate a couple of programming skills: (1) that you are able to reset without having to quit and re-run the program from the start; (2) that you can separate game play from the graphical objects in such a way that your program provides the same game play in at least two different situations.

Assessment

You will principally be marked in your one-on-one tutor session. You will submit one page of reflection on your experience, which can be used in off-line moderation of your marks.

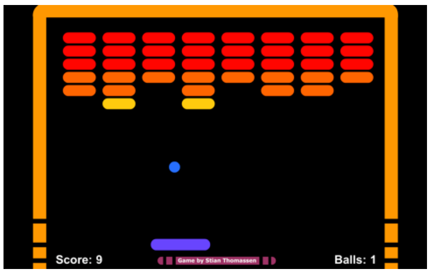
For this assignment the tutors will assess the following criteria:

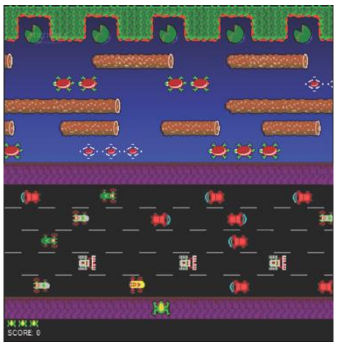
- Vision (how challenging was what you tried to do)
- Achievement (how challenging was what you actually achieved)
- Technical (what technical challenges did you have to solve, how good is the programming)
- Design (how good are the visual results and the game play)
- Requirements (did you meet all the requirements in the list above)

While we would like to see challenging games attempted by the more confident programmers in the class, we want people to attempt something that they can comfortably complete within the time available. Therefore we will allocate 60% of the marks towards the technical, design and requirements criteria (i.e., did you produce a good implementation), with 30% for vision and achievement (i.e., how challenging it is), and the remaining 10% for your reflection on your experience. *The marking rubric is available on the course website*.

Some inspiration for possible projects

Breakout. You could create a proper game out of the bouncing ball assignment. You would need to consider how to make it more compelling and add extra features, like special bricks that change the ball's behaviour.



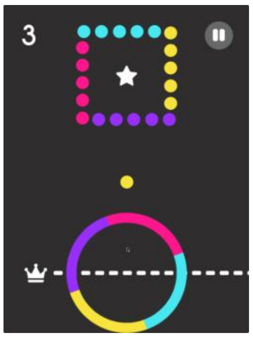


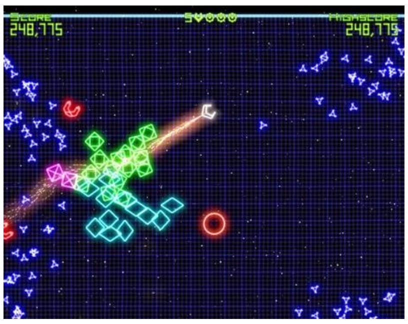
You could implement an '80s arcade game like *Frogger* or *Pacman*, perhaps as an abstract shape-based game.

Pacman is grid based, so you can represent the playing arena a two-dimensional array in your program.

Frogger has distinct horizontal zones, each with a limited number of objects in it .

Color Switch is an attractive, compelling game using simple 2D computer graphics http://colorswitchaz.com It might provide inspiration for a clean and simple graphical style. The game play of *Color Switch* might be challenging to program but simpler ideas could work well.





You could try a 2D version of *Asteroids*, with attractive geometry. This screen shot is from *Geometry Wars*.