INFO3220: Object Oriented Design

Bernhard Scholz
Bernhard.Scholz@sydney.edu.au

Semester 1, 2017

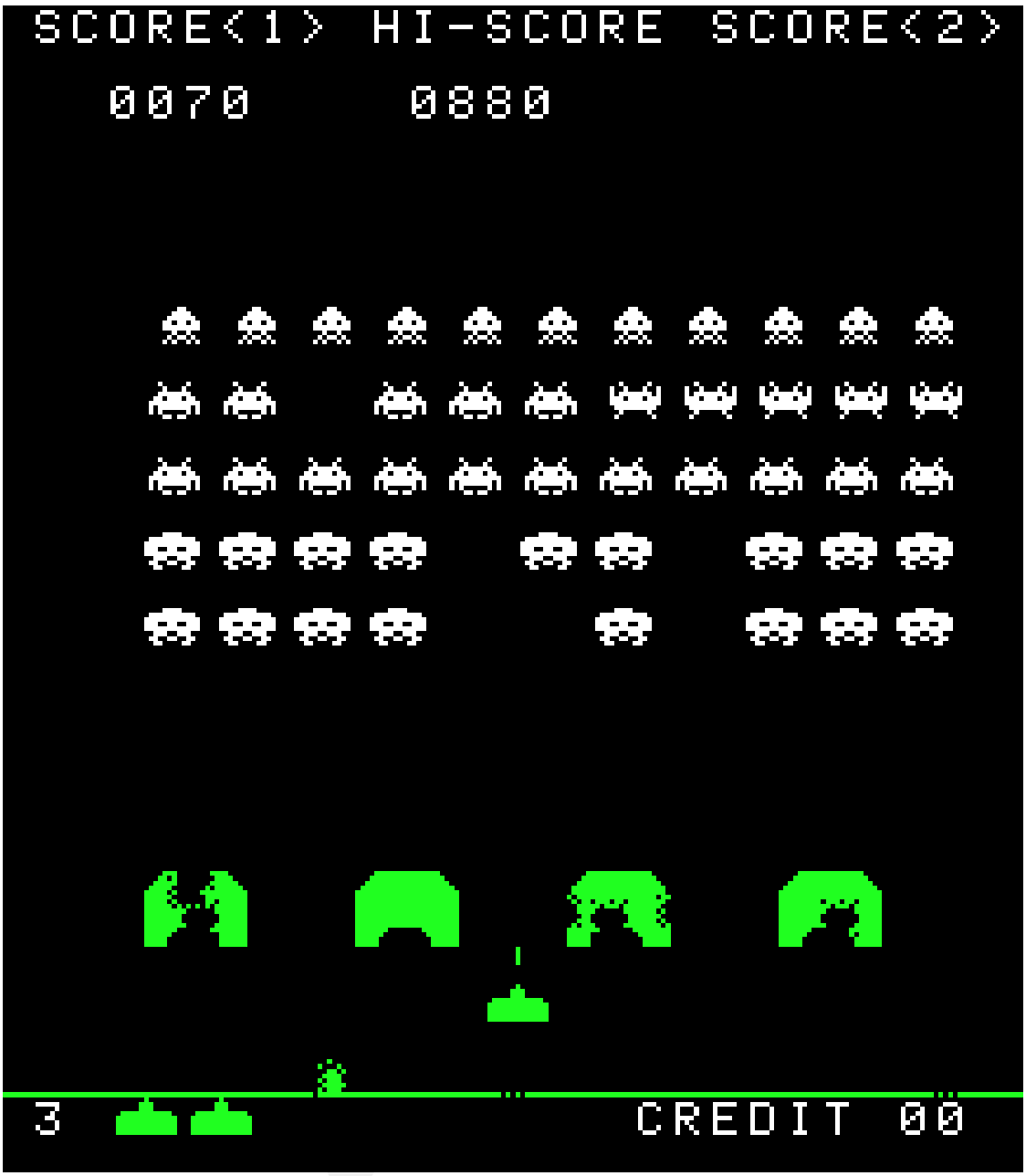
Contents

Assignment Overview

Assessments: Assignment

There is one assignment in three stages. After the first stage, each builds on the work of the previous one.

The assignment is to build an advanced version of the computer game *Space Invaders*. The computer game is one of the earliest shooting games that was created by Tomohiro Nishikado in 1978. A single player defeats waves of aliens with a laser cannon to earn as many points as possible for each shot alien.



[1]

With explosions, of course.

Hell is Other People 's code. (misquoted from Sartre.)

The first stage you will do yourself.

At stages two and three, you will extend someone else's code from your class. Your tutor will determine whose. Make sure your code is nice to maintain...

There is no group coding in this course[2]

But how will it be graded?
And while we're at it, what if I get some code I really hate, or that doesn't work?

[1] By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=17676369

1. We will attempt to find code that works. If we can't then that doesn't bode well.
2. The grade for Stages after the first will be based on what you *add* as well as how well you work with the existing code (simply replacing previous code with your own is not acceptable).
3. The assessment for the first Stage will be easy because we're assessing the difference between *nothing* and your code.
4. Marks for subsequent stages will be based on the *difference* between your code and its previous version.

Space Invaders
Stage one: A lonely shooting space ship

Write a graphics program using Qt Creator to model the sideways movement of a space ship, that can shoot laser beams.
There will be a small text ".config" file in which the details of the space ship's initial position, the movement, and shoot instructions.

Stage 1: A lonely spaceship

Stage 1 Overview

The first stage of your assignment is to create a graphics display using Qt Creator (whichever version works but a recent version if possible!). Remember you must demo your code on the lab machines.

- Configuration file in a format that we DON'T specify — **NOT QSettings**. Use a TEXT file. Call it what you want.
- The space ship moves according to a sequence of instructions in the config file.
  - i.e. you are not using keyboard events yet, whatever the configuration file says to do is what your ship should do for its first few moves of the game.
- An instruction is either a movement to left, a movement to the right, or shooting a laser beam.

Config file
The Configuration file stores:

- The size of the space ship as a string. The options are: tiny, normal, large, and giant. (These will be used in Stage 3).
- Space ship's starting position as an x coordinate.
- A sequence of movement/shooting instructions.

Marking guidelines
This Stage is worth 5%, and is marked out of 100.
50 marks are attributed to the following:

- compiling and running on the lab machines;
- sensible / appropriate OO design.
- reading the configuration file;
- rendering the space ship in four different sizes
- making a space ship move left/right and shoot a laser beam

15 marks are attributed to the following:

- appropriate use of a *creational* design pattern;
- sensible error checking, e.g., for missing or incorrect configuration files;
- clear code using meaningful variable and method names as well as informative commenting and documentation;

10 marks are attributed to the following:

- handling memory efficiently;
- clear code structure that will be simple to maintain and extend;

25 marks are attributed to extensions to the functionality that are simple and effective, **using only Qt libraries and what you write**: e.g.,

- sound effects
- ability to change colours and shape of space ship from within the program and then save the changes to the existing configuration file,
- pause and resume the sequence of instructions (must not use the space bar or left and right keys, they are reserved for Stage 2 and 3),
- animating the spaceship (glowing turbines etc)
- downward scrolling background stars.

Penalty of 1 mark if you go on and implement the next stage.

Some notes on Stage 2

Stage 2

What's required for Stage 2

1. One to two A4 pages typed review of the Stage 1 code you received. Portable Document Format (pdf) only. This component is worth 5%.

2. Extensions of the Stage 1 code. This component is worth 10%, and is marked out of 100.

Marking guidelines

Stage 2 is worth 15% to your final grade.

The first part is 5% and is a one-to-two page review of the code you received.

Don't skimp on the writing.

Submit via eLearning as a pdf or plain text only. If you submit in anything else you will be penalised. *It will be submitted using turnitin, within eLearning.*

Illustrate using standard UML if you believe it will help.

For full marks in your review you must comprehensively cover each of these components (they almost look like headings):

Documentation: how well was the code described, either as in-code comments or if there was separate documentation?

Extensibility: how well designed was the code for the extensions that you hope to make to it?

Design: what design patterns did the code use? Comment on whether you think they were good or poor choices, and justify your comments.

Implementation: was the coding well done? What would you have done differently? What was good about the implementation?

Style: comment on the style of the code. Were names, layout, code clichés consistent?

If you have studied the code thoroughly the above points can easily be made.

Marks will be *removed* for

- poor spelling, grammar and/or punctuation;
- bad organisation;
- unprofessional terminology such as "the previous guy was an idiot" or "this is rubbish coding";

- incorrect format submission (-1 mark penalty each time).

Stage 2 (cont'd)
Stage 2 :

- Aliens are initially laid out as a sequence according to the configuration file. Aliens have a bitmaps and a position.
- Aliens must have different bitmaps.
- Aliens have the ability to shoot the space ship. The shooting happens periodically. The beam travels vertically.
- When the space ship hits an alien, it disappears; a displayed score counter is incremented.
- When a beam of an alien hits the space ship, the game stops.

Marking scheme
The programming part is worth 10% for Stage 2, and is marked out of 100.
50 marks are attributed to the following

- it must compile and run on the lab machines;
- the previous behaviour must be preserved;
- appropriate use of a *structural* design pattern for game, aliens, beams are used.
- it must correctly use the configuration file to play the game.
- crash safety: the program will not crash given invalid configuration files;

15 marks are attributed to the following

- clear code using meaningful variable and method names as well as informative commenting and documentation;
- aliens can be grouped into swarms and show a single behaviour
- the configuration file can deal with swarms of aliens

10 marks are attributed to the following

- Aliens can move in a periodic fashion in a fixed trajectory.
- The aliens trajectory is encoded in the configuration file.

25 marks are attributed to sophisticated extensions. Some suggestions are listed below,

- Aliens can shoot their laser cannons in angles (not just in a vertical line).
- Aliens are actively hunting the space ship.

Discuss your extension idea(s) with your tutor before you go ahead!
Penalty of 1 mark if you go on and implement the next stage.

Some notes on Stage 3

Stage 3
For Stage 3 : Its worth 10%, and marked out of 100. The aim is to complete the computer game.

- just code: add to Stage 2 to complete the computer game. This will include:
  - proper control of the space ship with cursor keys (left/right/space) and mouse
  - add levels to the computer game. After all aliens are eliminated, the player reaches the next level.
  - have a score board (name and score)

Marking Scheme I
40 marks are attributed to the following

- it must compile and run in the lab (but this time you may use your laptop)
- preserve the previous two stage's functionality
- appropriate use of a *behavioural* design pattern for the game
- clear code using meaningful variable and method names as well as informative commenting and documentation

Marking Scheme II
20 marks are attributed to the following

- a *memory efficient* design.
- change the speed of the game
- add levels to the game

Marking Scheme III
15 marks are attributed to the following

- a sensible testing framework for your code[3];
- have a score board and menu settings for configuring the game

Marking Scheme IV
25 marks are for some cool extensions. Examples are:

- provide some swarm intelligence for the aliens
- have different types of cannons for the space ship
- add more features ...

Make sure you discuss your extension idea(s) with your tutor before you go ahead with them.

[3] Note: that doesn't mean you don't have to test your code before this!