INFO3220: Object Oriented Design

Bernhard Scholz
Bernhard.Scholz@sydney.edu.au

Semester 1, 2016

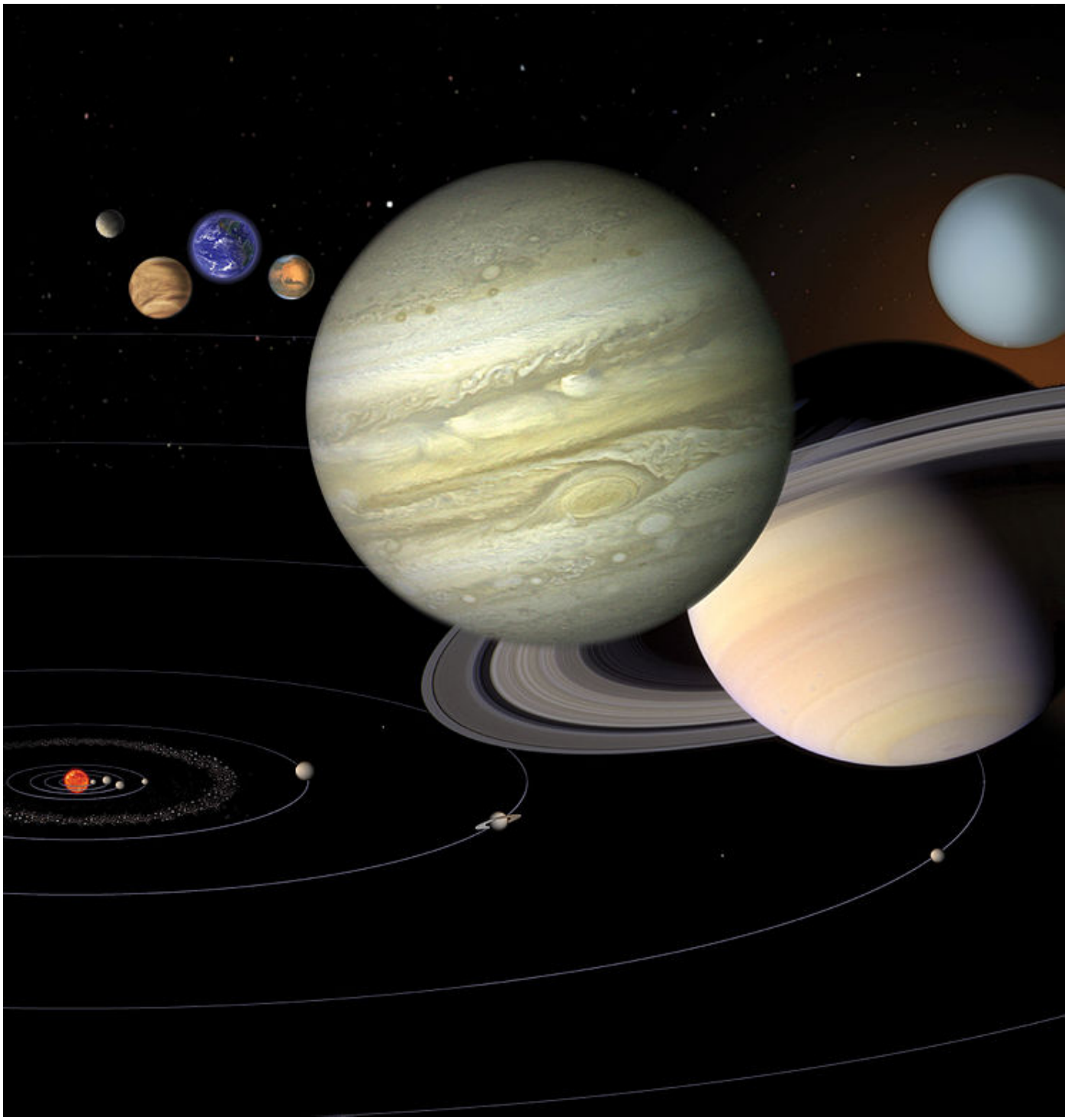
Contents

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 a *Planetarium/N-Body Simulation.*



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...

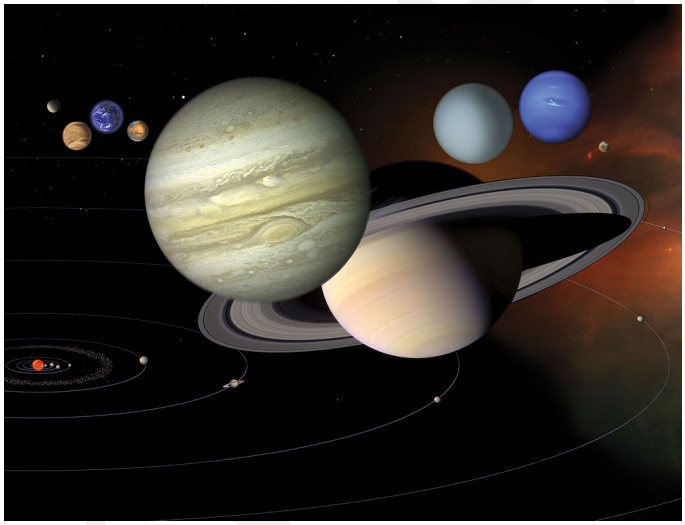
There is no group coding in this course[2]

[1] Source from Wikipedia

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. 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.

Our Solar System
Stage one: Our Solar System in 2D



Write a graphics program using Qt Creator to model the basic dynamics of planets using Newton's laws in 2d.
The planets and sun should be rendered as circles of different colours and sizes that move over time using Newton's law.
There will be a small text "config" file in which the details of name, circle size, circle colour, mass, initial position, initial velocity vector of every planet is stored.

Stage 1: Ad aspera ad astra

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 format. Call it what you want.
- The simulation runs in constant time step hard-coded as a constant in the program.
- The planets are rendered according to the specification in the config file.

Config file
The Configuration file stores:

- The name of the object (e.g.,sun, mars, etc.)

- Rendering colour of object
- Radius of object
- Mass of object
- The initial position vector (in 2d)
- The initial velocity vector (in 2d)

Marking guidelines
This Stage is worth 5%, and is marked out of 100.
For 50 or more marks, the following must all be achieved:

- it must compile and run on the lab machines;
- sensible / appropriate OO design.
- it must read the configuration file;
- define our solar system
- objects are moving according to Newton's laws

For the next 15 marks, the following must also be achieved:

- 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;

For the next 10 marks, the following must also be achieved:

- memory is handled efficiently;
- clear code structure, that will be simple to maintain and extend;

The remaining 25 marks are awarded for:

- Extensions to the functionality that are simple and effective, **using only Qt libraries and what you write**: e.g.,
  - ability to change colours, position, velocity and size of planetary objects from within the program and then save the changes to the existing configuration file,
  - pause and resume the simulation,
  - use bitmaps for planets.

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 :

- Structure your planetary objects into
  - *Solar Systems* that contain a sun and planets
  - *Galaxies* that contain solar systems and black holes
  - *Cluster* that may contain either other clusters or galaxies
- Define zodiacs as a set of lines between stars and render them in the simulation

Marking scheme
The programming part is worth 10% for Stage 2, and is marked out of 100.
For 50 or more marks, the following must all be achieved:

- it must compile and run on the lab machines;
- the previous behaviour must be preserved;
- appropriate use of a *structural* design pattern for solar systems, galaxies, and clusters;
- it must correctly use the configuration file to render all planets on the screen
- crash safety: the program will not crash given invalid configuration files;

For the next 15 marks, the following must also be achieved:

- clear code using meaningful variable and method names as well as informative commenting and documentation;
- have clusters implemented in a compositional fashion
- the configuration file has a format that permits loading solar systems, galaxies, and clusters.

For the next 10 marks, the following must also be achieved:

- zodiac lines are definable in the configure file or via a dialog box; zodiac lines are rendered depending whether they are enabled via a "zodiac" button,
- Solar systems, galaxies,and clusters are treated compositionally in the force computations between planetary objects.

The remaining 25 marks are scored for one or more sophisticated extensions. Some suggestions are listed below,

- introduce numerical methods that are faster than $\mathcal{O}(n^2)$ where n is the number of planetary objects.
- the double integral over time of the acceleration introduces significant numerical errors over time; how can we make more stable calculations for the simulation

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 15%, and marked out of 100. The aim is to add a nice GUI to the simulation.

- just code: add to Stage 2 to complete the planetarium. This will include:
  - changing the time step for the simulation
  - zoom capabilities for zooming in and out of the viewing window
  - setting the centre of the view in the simulation
  - convert the simulation to a 3d simulation

Marking Scheme
For 40 marks, the following must all be achieved:

- 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 GUI
- clear code using meaningful variable and method names as well as informative commenting and documentation

60+ marks
For the next 20 marks, the following must also be achieved:

- a *memory efficient* design.
- interface to change the time step of the simulation
- interface to center the viewing window of the simulation
- interface to scale the viewing window

75+ marks
For the next 15 marks, the following must also be achieved:

- a sensible testing framework for your code[3];
- introduce an automatic adjust "button" that adjust the viewing window at a given point in time so that all planets are visible in the viewing window.
- have additional info panels that show information of planetary objects by clicking on planetary objects

85+ marks
To get the last 25 marks, you must add some cool extensions. Examples are:

- convert the simulation to a 3d simulation with 3d rendering of planets
- advanced viewing window management that sets zoom and angle,

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

- flying trajectories for viewer in space,
- compute correct overlap between planets for 3d rendering,
- more things with exclamation marks!!

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

