COMP612 Computer Graphics Programming

Assignment 1

This is an individual assignment. All work you submit must be entirely your own.

Where your implementation idea or inspiration has been taken or adapted from other published sources those sources should be acknowledged appropriately in the class header and detailed in your developers log book.

You must develop using the Eclipse version and jogamp libraries as specified on Blackboard. It is your responsibility to submit an Eclipse project that runs on a COMP612 classroom machine.

It is expected that you will work consistently on this assignment, from hand out to due date. Time will be allocated in class each week for you to ask questions, get help, and work on your assignment. Please be aware that this is not an assignment that can be completed at the last minute. Each class you will be exposed to new concepts, and as we work through these concepts you will be able to progress your assignment.

Additional Feature Date: Monday 12th March at 10:00.
Due Date: Thursday 29th March 2018 at 15:00.

Additional Feature Process:

A feature approval form (available on Blackboard) must be submitted, in class, and the ideas discussed with the tutor *before* the specified deadline. You may not implement the same feature as another student. A list of approved features will be made available on Blackboard. If your idea is similar to another student's then the plan for implementation must be different. Each student must have two unique additional features so it is in your interest to formulate and submit your ideas early. Only approved features will be marked.

Submission:

You must submit an electronic copy via Blackboard using the link provided. Your submission must contain:

- A zip file of your eclipse project and any resources required for the code to run.
- An electronic version of your logbook as a PDF file.
- Please ensure that your java class headers, eclipse projects and logbook file names contain your first and last name and student id.

The assignment will be accepted up to 2 days late. A penalty of 5% per day (or part of a day) late will be imposed.

If your ability to attempt this assessment or prepare for this assessment has been seriously affected by exceptional circumstances beyond your control you *may* qualify for **special consideration**. Applications for extensions should be made on or before the due date of the assessment and appropriate supporting evidence to be supplied within 5 working days of the application being submitted. Full details of how to apply and to determine whether or not your situation qualifies for special consideration can be found on Blackboard.

From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

Marking:

This assignment will be marked out of 60 marks and is worth 20% of the overall mark for the paper.

A marking rubric for this assessment has been posted on Blackboard.

When marking each part of your assignment I will take into account not only the visual effect but also the quality of the code you have written. Quality code is code that is well commented and documented (Javadoc) and code which is well designed. Good code should be simple, clearly laid out and be free from smelly code. See https://en.wikipedia.org/wiki/Code_smell to remind yourself of, or learn about, common smells such as duplicated code, overly long methods, etc.

Plagiarism:

Please be aware that any piece of your assessment might be tested with the plagiarism prevention software to which AUT subscribes.

Assignment objectives:

This assessment assesses your ability to:

- Handle basic 2D OpenGL animation programming
- Implement mouse interaction
- Use bitmap font drawing
- Develop simple particle systems
- Use transparency (alpha blending)
- Use Vertex-by-Vertex rendering
- Implement fixed graphics pipeline rendering
- Write simple 2D parametric functions to render geometric objects
- Use display lists to precompile static objects
- Employ OO design principles in a graphics application

Notes:

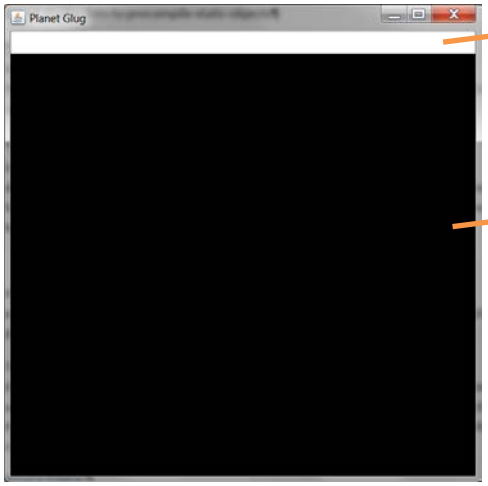
- In this assignment you do not have to handle a window re-shape.
- Your assignment will be re-compiled before running, so make sure it works on the lab computers, not just on your home computer.

From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

2D Scene: Alien Planet

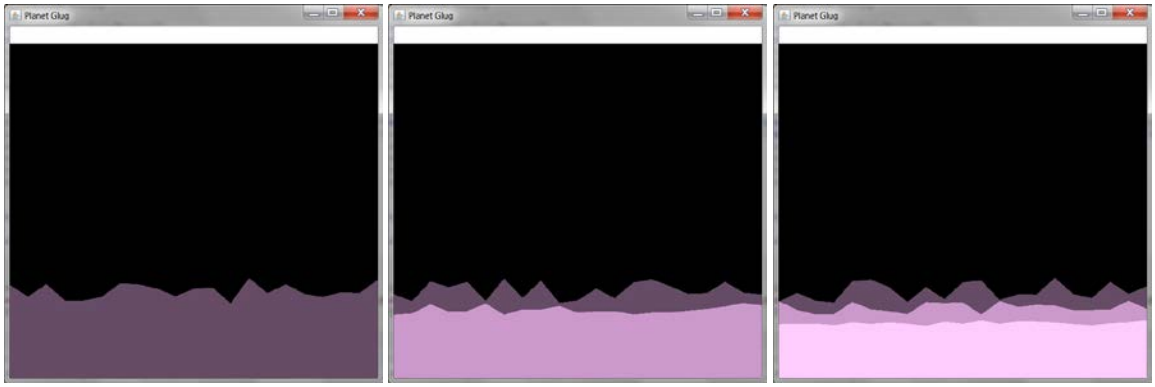
1. Basic 2D drawing [1 mark]

You will draw in two regions. In the top region you will draw an alien planet scene of your own design. You can choose whatever colours you like. In the top area you will draw buttons for controlling the application.



2. Planet surface [5 marks]

The surface of your planet should be formed by two or more irregular polygons. Layers and different colours should be used to give perspective. Each time the program is run the surface layer's vertices that form the top of the surface should be randomly generated. Each layer should be able to be created using the same method with different parameters to avoid unnecessary code duplication. Pre-compile your layers into display lists.



From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

3. Stars [4 marks]

The night sky should be filled with at least 100 stars, each rendered as a dot with differing intensities and colours that suit your planet's aesthetic. The position and colour of the stars should not change and should be determined randomly during the initialisation phase of the program. Stars should only be drawn in the drawing area of your scene. Pre-compile the stars in a display list.

4. Planetary Flora [4 marks]

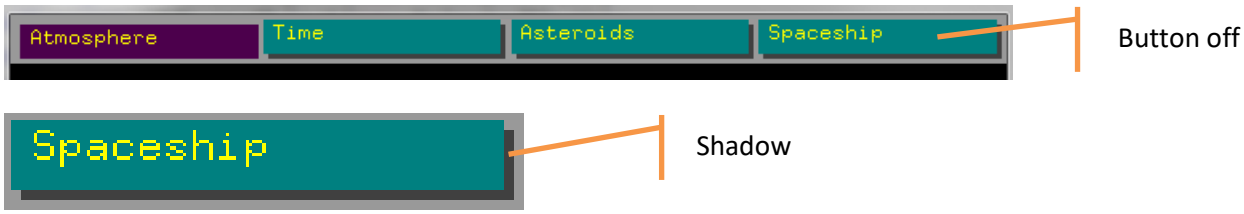
Design and render a number of randomly positioned alien plants of your own design. You should use vertex by vertex colouring to give the plants a 3D look. Plants should only grow in the planet's ground. Experiment with rendering order and sizes to get the look you want.

5. Toggle Buttons [5 marks]

In the button area you created in (1), add your buttons:

- Asteroids
- Time
- Atmosphere
- Spaceship
- any others required for your feature extensions

For the button names you must use GLUT bitmap fonts. The buttons should also have simple shadows. Note that you will have to code these buttons yourself using OpenGL. These are toggle buttons with 'on' and 'off' states. The buttons should change appearance (for example, colour) when clicked to indicate whether they are in the 'on' or the 'off' state.



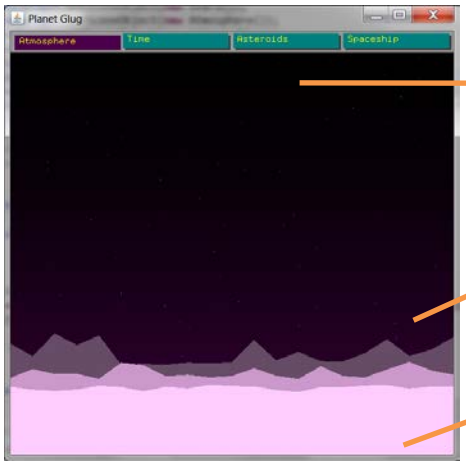
You should avoid code duplication by writing a Button class. You may want other classes, like one to manage all the buttons in your scene.

Make all the buttons operational.

From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

6. Atmosphere [4 marks]

To simulate an atmosphere, draw a rectangle over the night sky with a colour gradient. The colour you choose will be dependent on your planet's surface colour. In order not to erase the previously drawn stars, use alpha blending to make the rectangle semi-transparent. In my example the top colour is fully transparent and the bottom colour has an alpha value of 0.5 so that the atmosphere mask has a transparency gradient as well as a colour gradient. You'll need GL_BLENDING for this effect.



Top: Magenta (1.0, 0.0, 1.0) with an alpha = 0

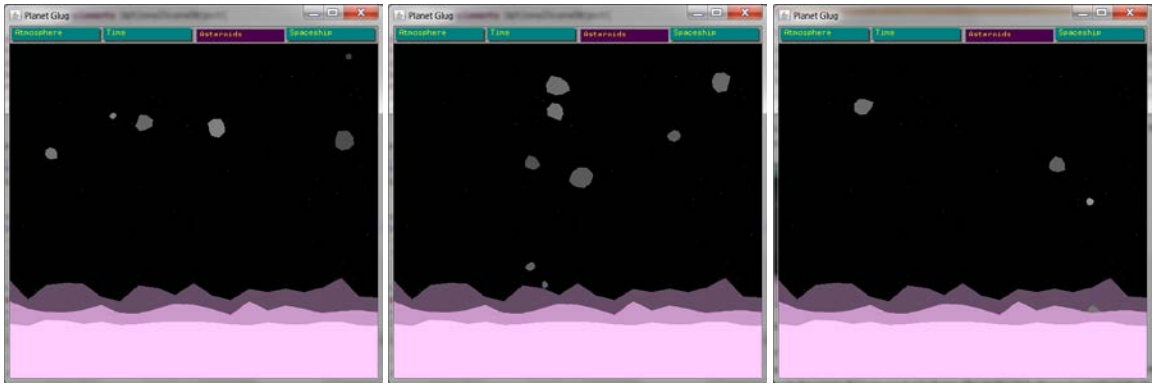
Near Bottom: colour interpolated ~ ¼ magneta/¾ purple with alpha close to 0.5

Bottom: Purple (0.5, 0.0, 0.5) with an alpha = 0.5 rendered *before* planet ground layers.

The atmosphere should only be rendered when the "Atmosphere" toggle button is on.

7. Asteroids [10 marks]

Develop a particle system to simulate asteroids. To conserve memory, you should reuse existing particles for efficiency (if you don't you'll eventually run out of memory). Each time a particle is recycled it should be randomised again. For asteroids, the particles are generated so that they appear from the top of the screen and so that they are added at different times. Each asteroid (or particle) is rendered as a circle—using a triangle fan—where the radius varies randomly at points around the circumference. Asteroids should fall downwards. For more realistic asteroids, vary their size and their falling speed. When the asteroid reaches the bottom of the screen, it should be recycled.

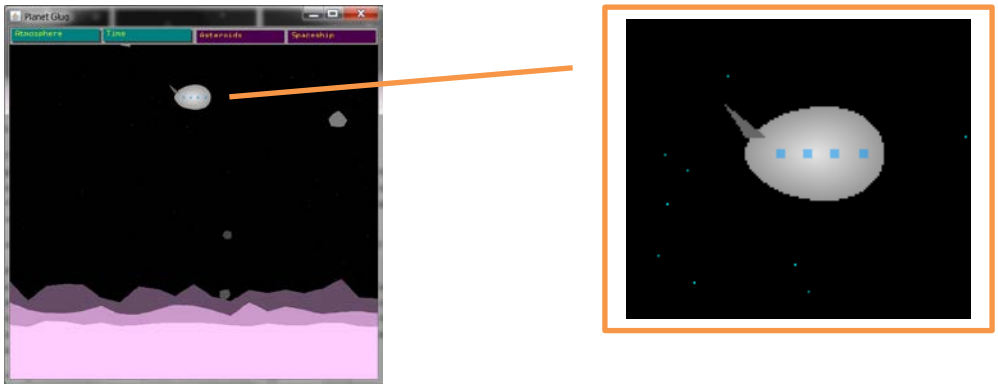


Asteroids should only be rendered when the "Asteroids" toggle button is on.

From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

8. Spaceship [3 marks]

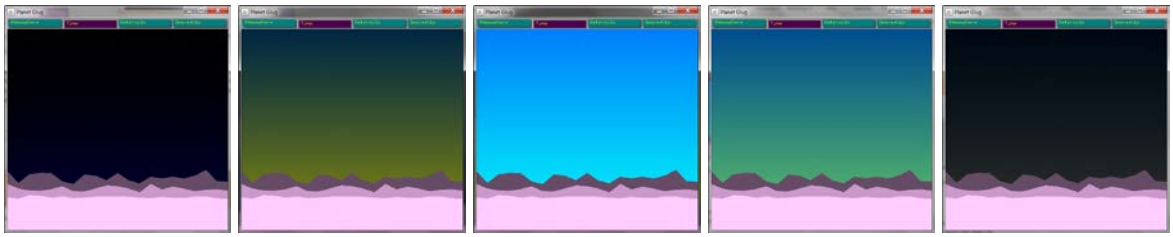
Draw a spaceship or UFO of your own design that flies horizontally (in a straight line) through the scene. You can use basic shapes like circles and ellipses to construct its shape. By using vertex colouring, you can make it look 3-dimensional. You can also use alpha blending to create a semi-transparent cockpit. My spaceship is egg shaped and has 4 windows and a tail fin.



The spaceship or UFO should only be rendered when the "Spaceship" toggle button is on.

9. Time [4 marks]

Implement a day-night cycle. To do this you need to use another semi-transparent mask that changes colour over time. At night, it should be the same colour and transparency gradient as for the Atmosphere. At dusk and dawn, the horizon should turn blue/orange. During the day, the colour gradient should be blue to cyan. It should also be opaque/non-transparent (to hide the stars). Don't forget to use GL_BLENDING. If you use alternative colours, make a note of this in your logbook.



The day to night transitions should only occur and the day-night mask should only be drawn when the "Time" toggle button is on.

10. Additional Features [12 marks]

Add *two* new features to your alien planet scene.

Use your imagination. Features are graded according to technical difficulty, novelty and visual effect. For example more marks will be allocated for animated effects or algorithmically difficult effects.

You are expected to investigate possible ways to implement your features and to evaluate and employ the best solution. This investigation should be presented in your log book in detail and include relevant references to resources you used to help you design and implement your features.

From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

11. Logbook

You must hand in your log book as it forms part of your proof of authorship. Remember, the onus is on you to prove you are the creator of your product; thorough record keeping is essential to this process.

Your logbook should record dates, time spent, a record of bugs and fixes, design details, additional feature ideas, and details as to how you realised those features.

You should provide a small statement in the last entry that critically evaluates what you did well, what you might do differently next time, and identifies any shortcomings of your application.

Basic Geometries

A few basic parametric equations and in some cases pseudo code for procedures to render basic geometries such as circles and ellipses can be found on Blackboard.