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 developer's logbook.

You must develop using the Eclipse version and jogamp libraries 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. It is estimated that this assignment will take 20 hours to complete.

Additional Feature Due Date: Wednesday 6th March 2019 at 12:00 pm (noon).
Assignment Due Date: Friday 22nd March 2019 at 3:00 pm.

Additional Feature Process:

A feature approval form (available on Blackboard) must be submitted, in class, and the ideas discussed with the tutor by 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.

Late assignments, without an approved extension, will be subject to a deduction of 5% (one grade e.g. from C+ to C) of the total mark available for each 24-hour period, or part thereof, up to a maximum of five calendar days. Assignments over five days late will not normally be accepted or marked and students will receive a DNC (Did Not Complete) for that assessment.

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 must 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.

Marking:

This assignment will be marked out of 50 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 well commented and documented (Javadoc) and well designed. Good code should be simple, clearly laid out and be free from code smells. 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

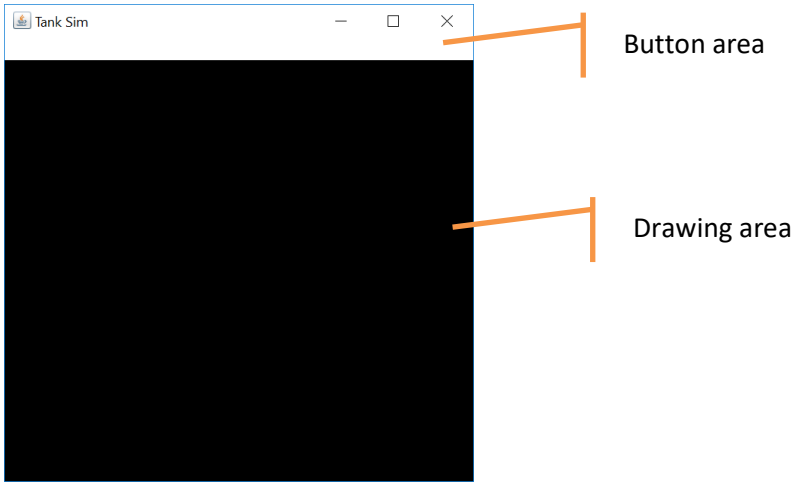
Notes:

- 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.
- You should read this assessment description in conjunction with the marking schema provided.
- To get nice looking polygons don't forget the turn on GL_SMOOTH and to get smooth lines enable anti-aliasing by enabling GL_LINE_SMOOTH. Remember for this to work GL_BLEND needs to also be enabled.

2D Scene: Tank

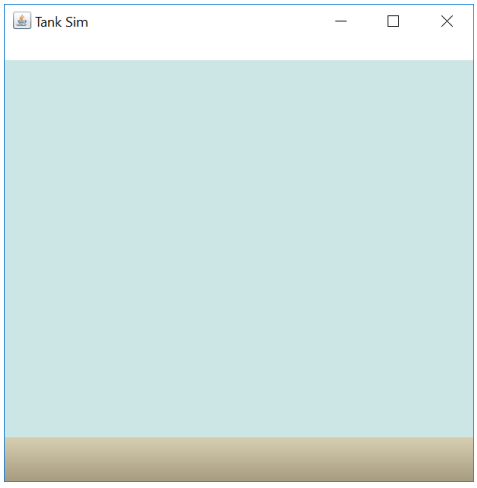
1. Basic 2D drawing [1 mark]

You will draw in two regions. In the bottom region you will draw a fish tank 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. Sandy Bottom [2 marks]

The base of your tank should contain a sandy bottom. Different colours should be used to give sense of perspective. In the image below I have made the drawing area colour (in 1) light blue.



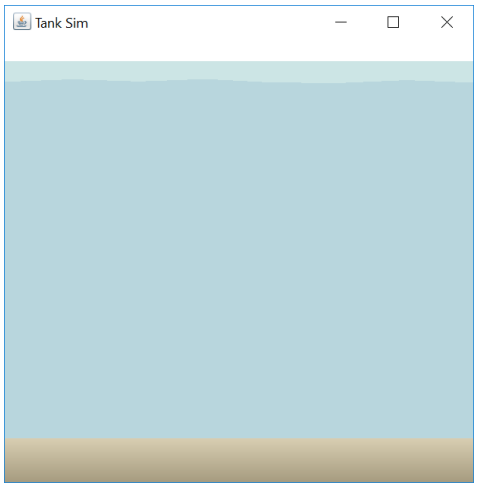
3. Water [6 marks]

Create a class called Water. Like the sandy bottom the top of the water should be a different colour to the bottom of the water to give a sense of depth. The water should be semi-transparent and be drawn after the Sandy Bottom so that the sandy bottom looks as if it is inside the tank.

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

Animating the Water Line

There are many ways you could approach this task. One of the simplest ways is to generate a collection of vertices that when rendered as a line-strip form a horizontal wave. To generate these vertices create the x values at a regular interval based on the number of points in your wave (for mine I have 8). The y values for each vertex are randomly determined within a specified water level range. If you generate two waves, you can render one wave and use the other wave as the target so that all vertices on the actual wave are animated slowly towards its corresponding target vertex. Eventually the target wave becomes the actual wave and a new target wave can be generated. Once you have this working as a line then use the collection of vertices to draw a quad strip that gives you the water as an area with the surface animated.

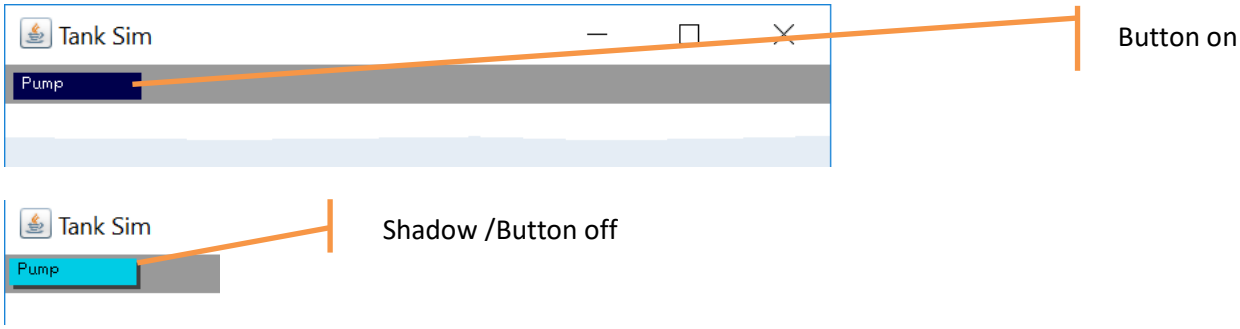


4. Toggle Buttons [5 marks]

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

- Pump
- Any others required for your feature extensions

For the button names you must use GLUT bitmap fonts. Note that you must code the buttons yourself using OpenGL. Toggle buttons are ones with 'on' and 'off' states. The buttons should change appearance (for example, colour and/or shadow) when clicked to indicate whether they are in the 'on' or the 'off' state.

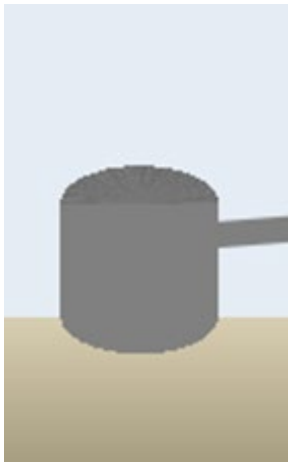


From the Office of Jacqueline Whalley, Associate Professor
<jwhalley@aut.ac.nz> | <+9 9219999 ext. 5203>

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.

5. Pump [2 marks]

The tank needs an air pump to aerate the water. Design a pump and draw it in your tank. You should make use of by-vertex attributes to get a nice look to your pump. I get the charcoal-like texture on the top of my pump (where the air is expelled) by using random shades of grey for each vertex. Because the geometry will be quite complex, and the pump will not move, to improve performance you should pre-compile the vertices and their attributes into a display list.

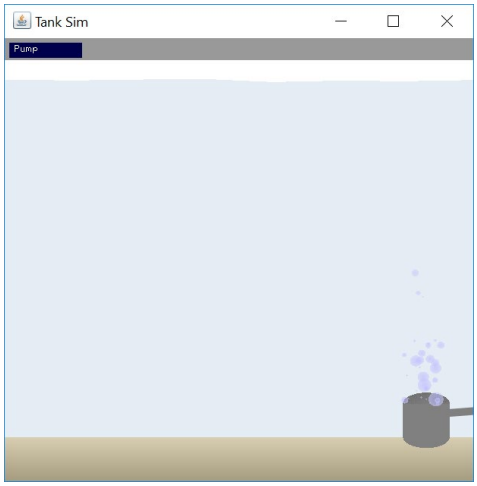


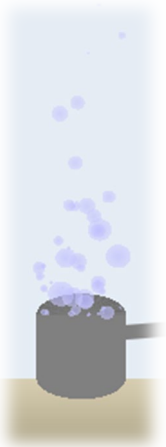
6. Bubbles [10 marks]

Develop a particle system to simulate air bubbles being released from the pump. 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.

Bubbles should only be rendered when the pump is turned on using the "Pump" toggle button. Each bubble (or particle) should begin as a randomly sized, semi-transparent circle. As each bubble rises toward the surface it should become smaller and more transparent. Bubbles should rise faster as they get smaller.

Bubbles must be gradually introduced after the pump is turned on, rather than created all at once, until you reach a maximum number of bubbles to avoid patterns and gaps emerging. A bubble must be recycled when it becomes fully transparent, its radius reaches zero, or it reaches the surface of the water.

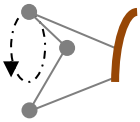


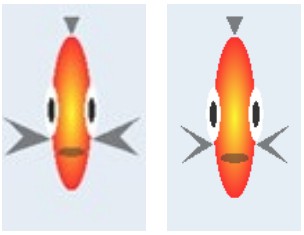


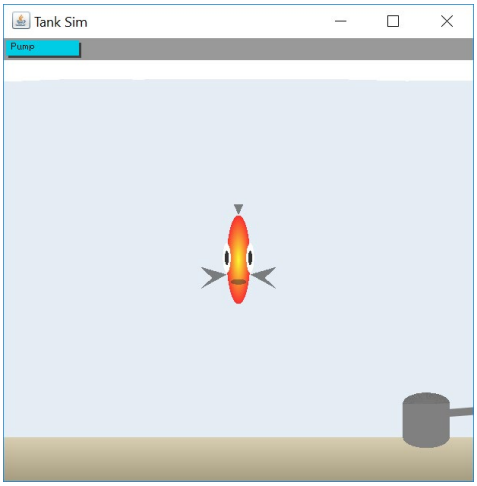
7. Fish [4 marks]

Draw a fish or other swimming sea-creature of your own design. It must have moving parts so that it appears to be swimming. 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 parts.

My fish is built using ellipses and triangles. To make my fish swim, I animated the fins' three outer vertices using an elliptical animation path.







8. Two Additional Features [12 marks]

Add *two* new features to your tank scene.

Use your imagination. Features are graded according to technical difficulty, novelty and visual effect. For example more marks will be awarded 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 logbook 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>

9. Logbook [8 marks]

You must hand in your logbook 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 **must** 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 pseudo code for procedures to render basic geometries such as circles and ellipses can be found on Blackboard.

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