# COMP3170 Assignment 3

## Objectives

This assignment covers the following topics:

- 3D modelling with triangular meshes
- 3D Transformations
- Perspective cameras
- Illumination and shading
- Texturing
- Transparency

Your task is to build a simple 3D world using height-mapped terrain, with water and lighting effects like this.

## Framework

A basic framework for the assignment is available via GitHub classroom:

https://classroom.github.com/a/p3fFYFbF

### Level files

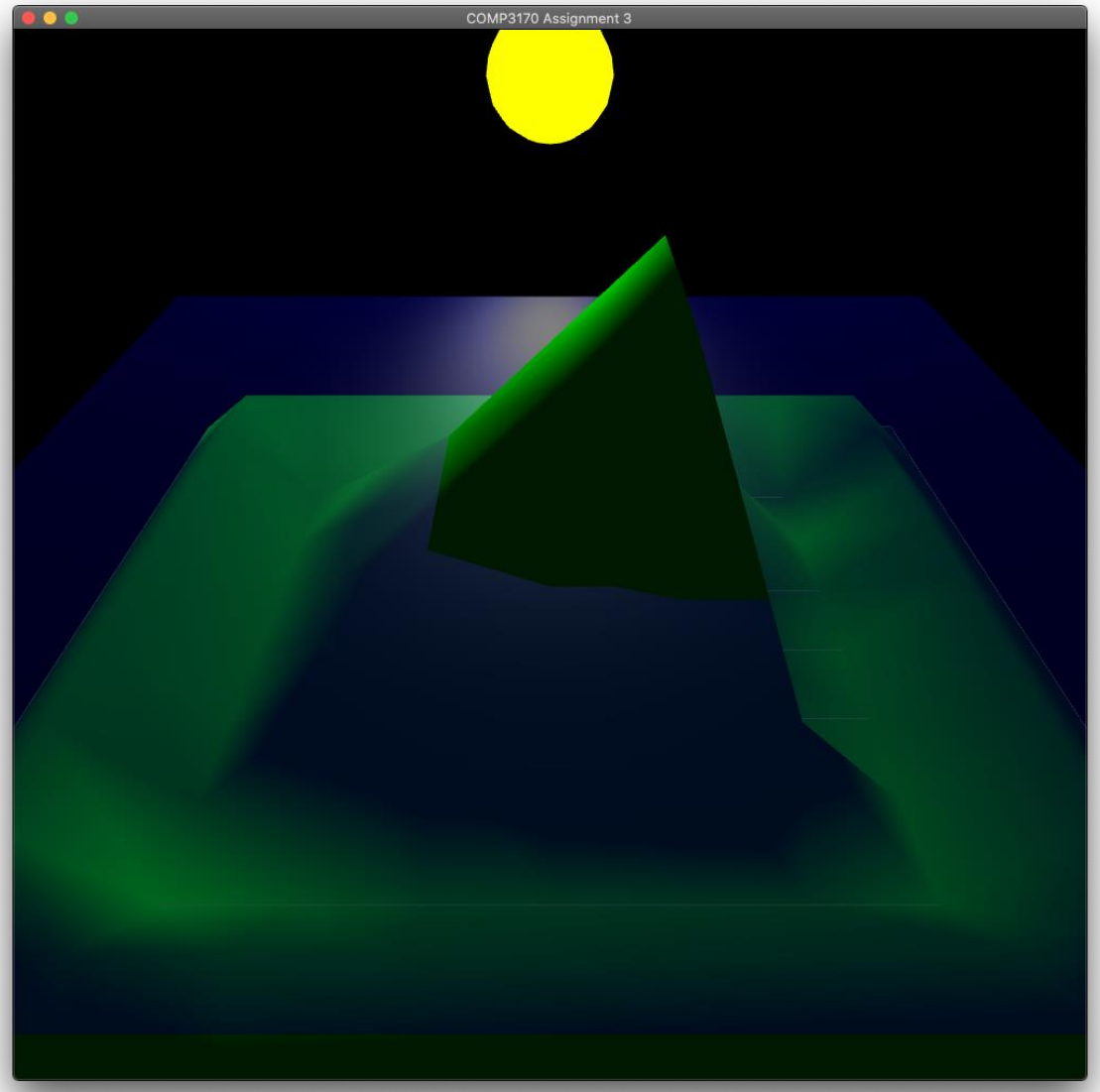
The assignment framework includes a class that loads a level definition from a JSON file. This file includes the following data:

level.json – an example level

```
{
	"name" : "Island”,
	"map" : {
		"width" : 11,
		"depth" : 11,
		"height" : [
			0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
			0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
			0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
			0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
			0, 1, 0, 1, 2, 2, 2, 1, 0, 1, 0,
			0, 1, 0, 1, 3, 3, 3, 1, 0, 1, 0,
			0, 1, 0, 1, 4, 4, 4, 1, 0, 1, 0,
			0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
			0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
			0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
			0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
		]
	},
	"waterHeight" : 2,
}
```

This defines height map, consisting of a 11x11 grid of points with different heights, with water covering everything up to height 2.

The resulting scene would look like this:




(Note this doesn't include texturing or ripple effects described later)

A simple framework has been provided to read these files Java objects. Your assignment should support this basic level format, and render any level that meets this spec.

## Scene features

Your mark will be based on the features you implement, from the list below. Each feature has a mark value attached. Completing more than 100% worth of features will not earn additional marks.

| Feature | Marks |
|---|---|
| Terrain | |
| - Heightmap mesh generation | 40% |
| - Diffuse & ambient lighting (smooth fragment lighting) | 20% |
| - Texturing (one texture) | 10% |
| - Texture blending with terrain height | 5% |
| Water | |
| - Diffuse & ambient lighting | 5% |
| - Transparency | 5% |
| - Specular lighting | 10% |
| - Ripple effect | 5% |
| Camera | |
| - Fly through camera | 5% |
| Light | |
| - Animated sun | 5% |

### General requirements

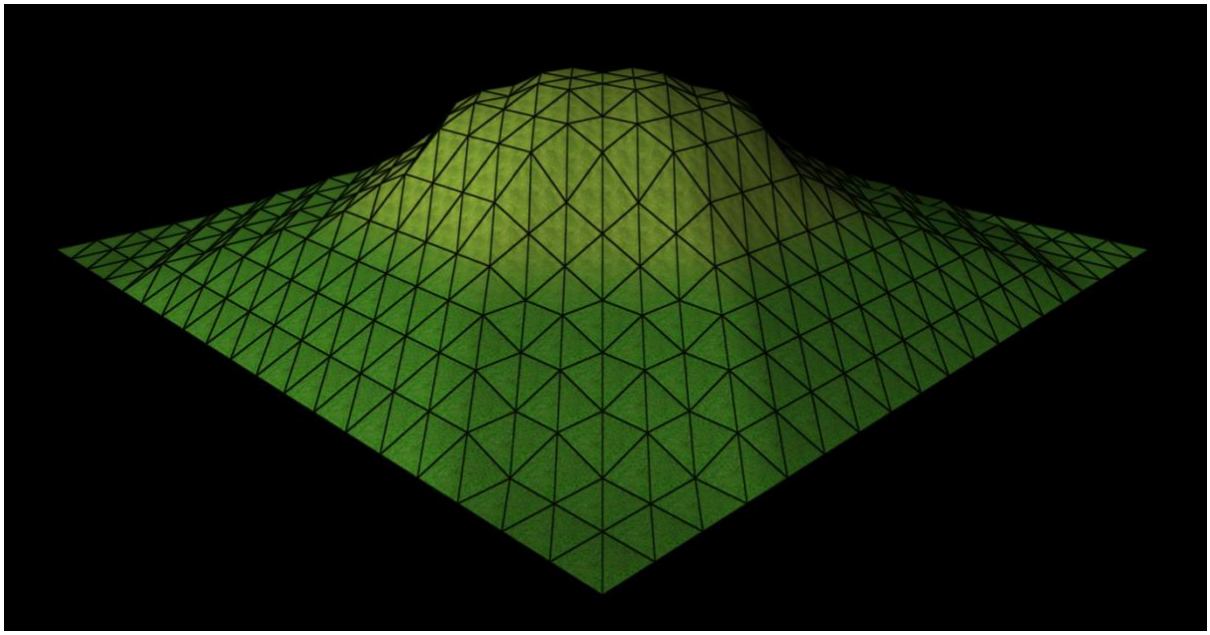
Your scene should be implemented using:

1) Anti-aliasing using 4x supersampling.
2) Backface culling
3) Mipmaps for all textures (with trilinear filtering)
4) Gamma correction (with a default gamma of 2.2)

### Terrain

**Height map mesh generation: 40%**

Use the height-map data provided to generate a 3D mesh. The mesh should consist of *width* by *depth* points (for the values specified in the level file) spaced in a grid in the x/z axes. The y-coordinate of each point is given by the corresponding value in the *height* array. These points should be joined into a mesh of triangles. So, a 20 by 20 height map will have 19x19 squares each represented with two triangles, as shown below.



**Diffuse & ambient lighting: 20%**
You should provide a terrain shader that implements diffuse and ambient lighting with the following specs:

- Smooth Phong shading (i.e. in the fragment shader)
- Appropriate vertex normals computed by interpolating the face normals of surrounding triangles
- Point light source determined by the position of the sun in the scene.

**Texturing: 10%**
The terrain shader should use a texture to determine the diffuse reflection coefficient of the terrain. The texture should be tiled so each 1x1 square of the terrain contains 1 repetition of the texture. Some appropriate textures are provided in the assignment framework, or you can use a texture of your own choosing, as long as you have a license to use and distribute it. Sources for all third-party assets should be documented in your report.

Textures should use mipmaps and trilinear filtering. Gamma correction should be applied (with a default gamma of 2.2).

**Texture-blending: 10%**
Multiple textures (e.g. grass and rock) are used at different terrain heights (at your discretion), with appropriate blending where they meet. (E.g. underwater terrain could use a sand texture while above water is textured with grass).

## Water

The water should be a single rectangle with width (x) and depth (z) dimensions to cover the entire terrain. It should sit at the height specified by the 'waterDepth' property in the level file. You should provide a specialised water shader to colour this.

**Diffuse & ambient lighting: 5%**
The water shader should implement ambient and diffuse lighting.

**Transparency: 5%**
The water should be semi-transparent showing the terrain below.

**Specular lighting: 5%**
The water shader should also implement smooth specular lighting, reflecting the light source.

**Ripples: 5%**
Implement animated ripples on the surface of the water by changing the normal vector (without changing the mesh geometry).

## Camera

**Fly-through camera: 5%**
The Camera should be a **perspective camera** with appropriate settings for field-of-view and near and far planes. The aspect ratio of the camera should dynamically adjust when the window is resized.

The camera controls should implement fly-through controls:

| Key | Movement |
|---|---|
| W or Up | Pitch upwards |
| S or Down | Pitch downwards |
| A or Left | Yaw left |
| D or Right | Yaw right |
| Space | Move forward (in the direction faced) |

## Lights

The scene should include a sun, rendered as a sphere and be treated as a point light for lighting calculations.

**Animated sun: 5%**
Animate the sun to make it rotate around the scene and change colour appropriately depending on the time of day.

## Report

The assignment framework includes a report template. Complete the table at the front of the report indicating which of the components above you are attempting. For each component, give an indication of where in the project it is implemented. Failure to submit a complete report carries a **20% penalty**.

## Submission

Assignment code and reports will be submitted through GitHub Classroom. We will mark your last commit before the assignment deadline.

## Marking Rubric

**Total mark** = (components completed) * (50% * code correctness + 40% * clarity of code + 10% for creativity).

Each component that you implement will be marked for completeness of the implementation against the specification. The overall program will be marked for correctness, clarity and creativity according to the following rubric.

| Grade | Correctness | Clarity | Creativity |
|---|---|---|---|
| HD (100) | Code is free from any apparent errors. | Good consistent style. Well structured & commented code. | Original use of graphics to achieve effects not included in lectures or pracs. |
| HD (90) | Code has minor errors which do not significantly affect performance | Minor inconsistencies in style, or patches of undocumented code. | |
| D(80) | Code has one or two minor errors that affect performance | Code is mostly readable but could be better structured | Creative use of graphics effects from lectures/pracs to add distinctiveness to submission |
| Cr(70) | Code has multiple significant errors | Code is inconsistently structured, but readable | |
| P(60) | Code is functional but contains major flaws | Code is poorly structured and patchily documented | Basic effort put into colour palette and shapes to give the simulations a consistent style |
| F (< 50) | Code is not functional | Code is undocumented and largely unreadable | Implements required features without effort put into style |