Student ID: 27033004
Name: Jonathan Edwards

COSC422 Assignment 1 Report

The Scene

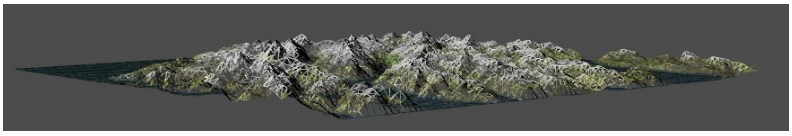


Figure 1 - Scene movement, Less Tessellation Levels (Blocky).

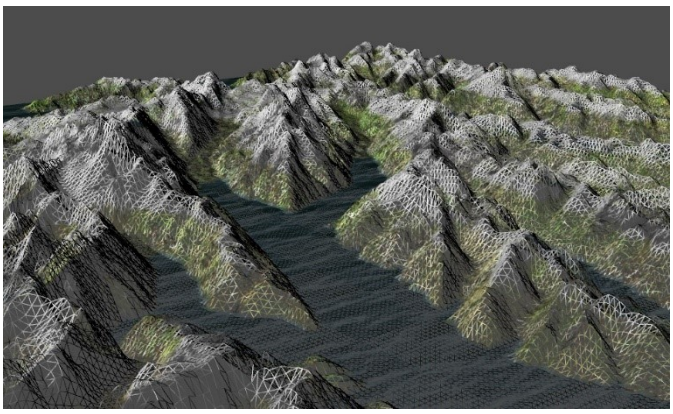


Figure 2 - Scene movement, More Tessellation Levels (Detailed).

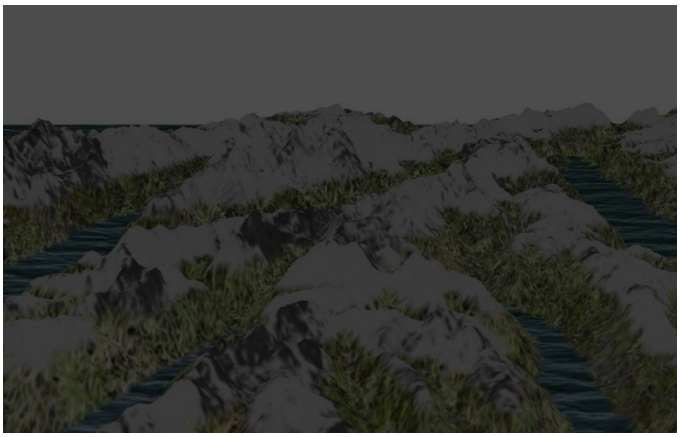


Figure 3 - Ambient Lighting

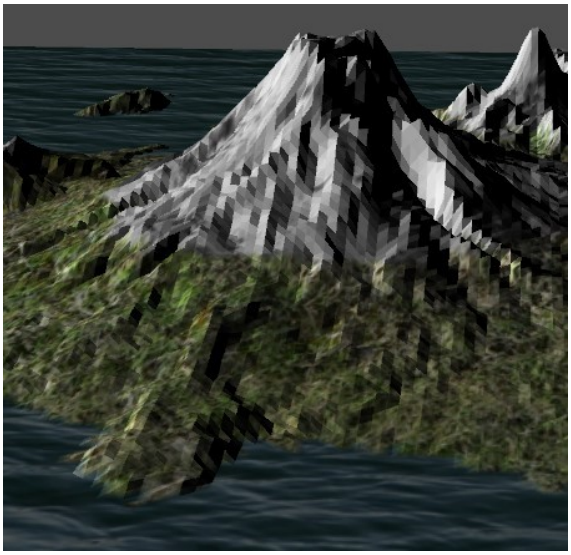


Figure 5 - Blending

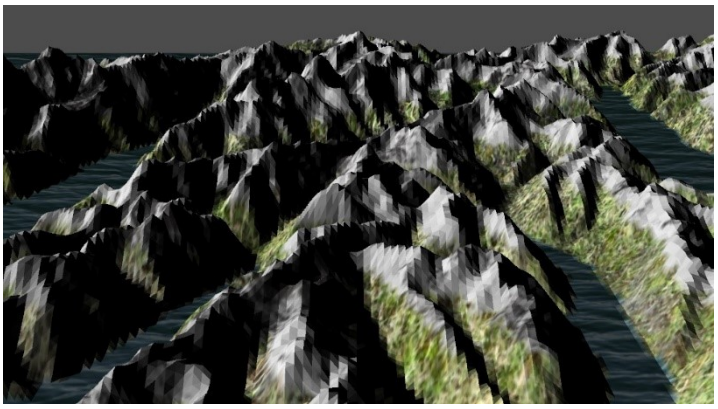


Figure 4 - Ambient + Diffuse Lighting

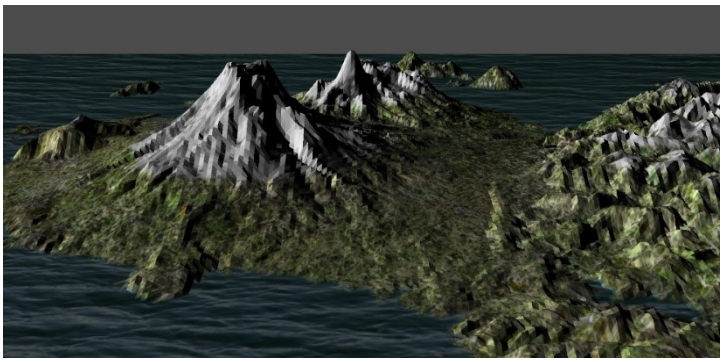


Figure 6 - Scene 2

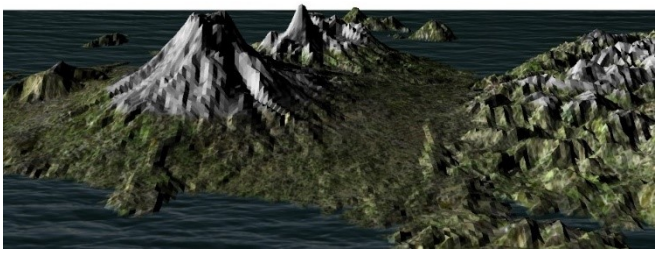


Figure 8 - Cracking Fix, white background

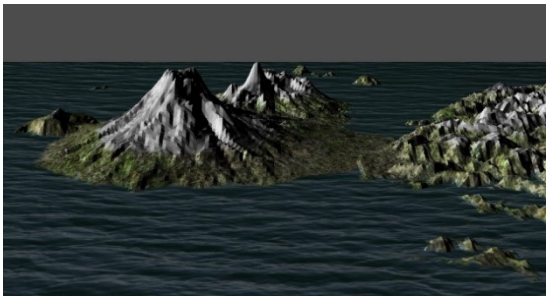


Figure 9 - Adjustable Water level

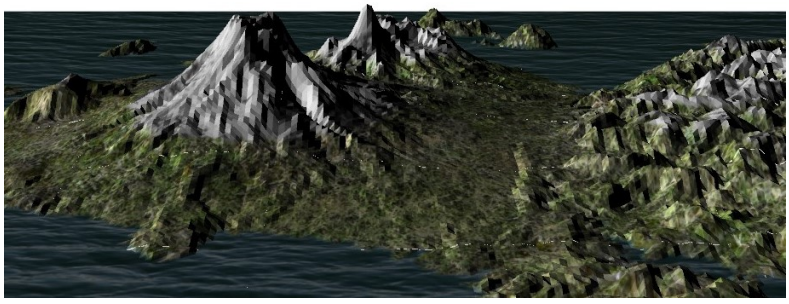


Figure 7 - Cracking Issue, white background

Student ID: 27033004
Name: Jonathan Edwards

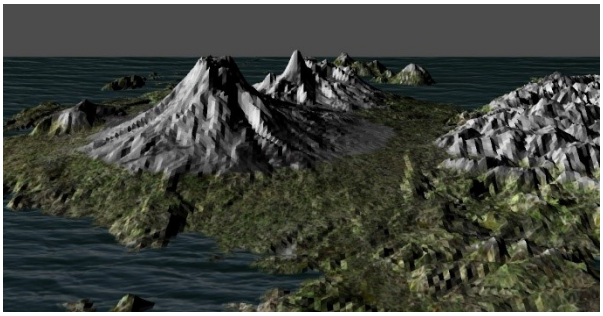


Figure 10 - Adjustable Snow level

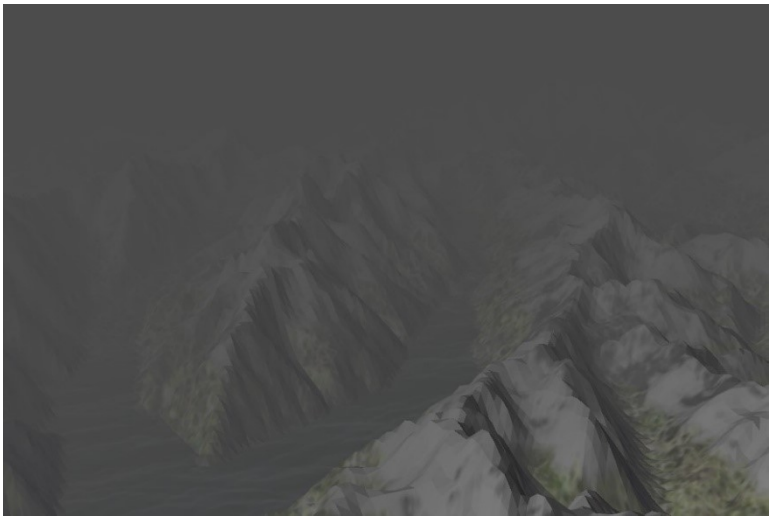


Figure 11 - Fog

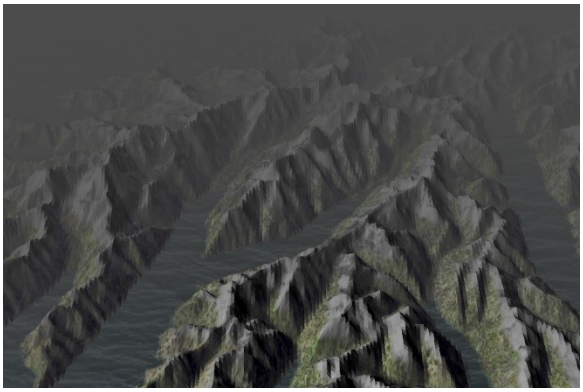


Figure 13 – Camera height

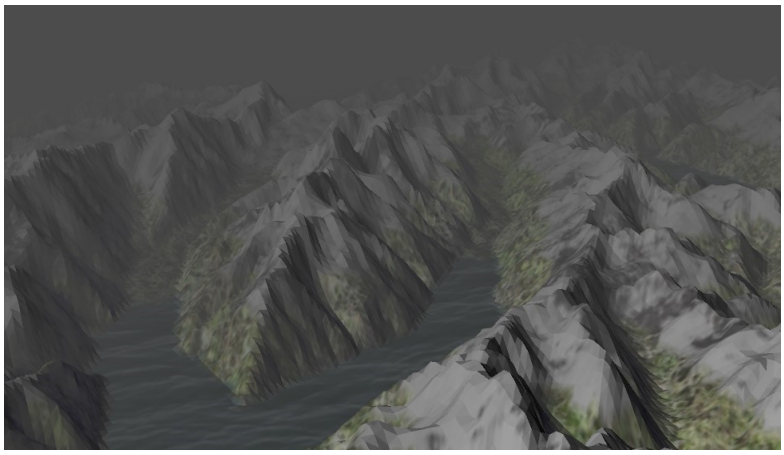
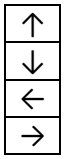


Figure 12 - Adjustable Fog density level



Basic Terrain Model

1. **<u>Dynamic level of detail – Camera:</u>**
   Move around the scene via keystrokes.
   Figure, 1 & 2 demonstrate how the Tessellation level changes. This can be seen when the camera actively moves around/over the scene (The Tessellation level is controlled by the Control shader). The scaling of the Height maps is done in the Evaluation shader. Spacebar Toggles the model from solid fill to wireframe & vice versa.
2. **<u>Lighting:</u>** This is computed in the Geometric shader. Figure 3 shows only ambient lighting and Figure 4 shows ambient + diffuse lighting. The diffuse lighting term was found via finding the Face normals of each sub patch dotted by the light pos.
   The Fragment output color is then outputted in the Fragment shader. Toggle 'l' shows the difference.

```
330     glm::vec4 light = normalize(glm::vec4(-100, 10, 10, 1.0));
```

```
diffuse = dot(light_pos, normal) * vec4(1);
```

```
22    #define     AMBIANT       vec4(0.3)
```

```
31    out_color = lighting_color * (tex_water + tex_snow + tex_grass);
```

3. **<u>Textures:</u>** Three additional textures were used, Water, Snow, and Grass. Figure 5 shows appropriate blending of textures, this was done by applying a weight to each texture in the Geometry. The weight was then timesed by the color of each texture in the Fragment shader. Texture mapping was done by,

```
22    vec4 tex_water = texture(_tex_water, tex_coords)  * tex_weights.x;
23    vec4 tex_snow  = texture(_tex_snow,  tex_coords)  * tex_weights.y;
24    vec4 tex_grass = texture(_tex_grass, tex_coords)  * tex_weights.z;
```

```
if (posn[i].y == water_level)
    tex_weights = vec3(1, 0, 0);    // water
else if (posn[i].y > snow_level)
    tex_weights = vec3(0, 1, 0);    // snow
else
    tex_weights = vec3(0, 0, 1);    // grass
```

4. **<u>Terrain Models:</u>** Two Height maps are used, MtCook and MtRuapehu. Pressing '1' loads MtCook and pressing '2' loads MtRuapehu. Figure 4 and 6 show this.

```
tex_coords.s = ((posn[i].x) - xmin) / (xmax - xmin);
tex_coords.t = ((posn[i].z) - zmin) / (zmax - zmin);
```

Extra features

1. **<u>Cracking:</u>** This was solved by computing for the length instead of the distance.

Student ID: 27033004
Name: Jonathan Edwards

```
19      float dist_avg = distance(vec3(eyePos.x, eyePos.y, eyePos.z), avg_n);
20      float dist_camera = length(vec3(eyePos.x, eyePos.y, eyePos.z));  /* Pos. relative to camera - Fixes cracking */
```

This can be seen in Figure 7 and 8. Toggling 'c' toggles cracking and toggling 'b' changes the sky color from grey to white and vice versa.

```
33      float level = int((dist - zmin) * 15.0 / (zmin - zmax)) + int(eyePos.w);
```

2. **Adjustable water levels:** Adjusting water level can be seen in Figure 9, compare with Figure 6 to see the difference. 'q' Increases and 'a' decreases the level.
3. **Adjustable snow levels:** Adjusting snow level can be seen in Figure 10, compare with Figure 6 to see the difference. 'w' Increases and 's' decreases the level.
4. **Fog:** Toggling 'f' for fog. Figure 11 shows this. In the Geometry shader a exponentiol is used softly fade the object in the distance. This is also clamped. The visibility color is then mixed with the sky color and other textures.

```
31      out_color = lighting_color * (tex_water + tex_snow + tex_grass);
32      out_color = mix(SKY_COLOR, out_color, visibility);  // Final output - Mix Fog
```

```
76          visibility = exp(-pow(length(posn[i] * density), GRADIENT));  /* Exponential curve */
77          visibility = clamp(visibility, 0.0, 1.0);  /* Clamp and send to Frag. shader */
```

5. **Adjustable Fog:** Adjusting snow level can be seen in Figure 12, compare with Figure 11 to see the difference. 'e' Increases and 'd' decreases the density level.
6. **Smooth Shadding attempt:** This was attempted in the Fragment shader to find the Vertex normals instead of the face normals.

```
vec3 fdx = vec3(dFdx(vPos.x), dFdx(vPos.y), dFdx(vPos.z));
vec3 fdy = vec3(dFdy(vPos.x),dFdy(vPos.y),dFdy(vPos.z));
normal = normalize((cross(fdx, fdy)));
```

7. **Extra Buttons:** Figure 13 shows lowering the height of the camera via '+' or '=' and increasing the height by '-'.

Control Functions

↑ Move forwards	'q' Increases water level
↓ Move backwards.	'a' Decreases water level
← Turn left.	'w' Increases snow level
→ Turn right.	's' Decreases snow level
Spacebar Toggles between wireframe and textured view.	'f' Toggle fog
'1' Displays Height Map 1	'd' Decreases fog density
'2' Displays Height Map 2	'e' Increases fog density
'+' or '=' Decreases the cameras height	'l' Toggle light shading
'-' Increases the cameras height	'b' Toggle sky color
'c' Toggle cracking	

Compiling & Running Application

On a Windows Machine, on the Makefile level, type: 'make && make program'

References to Sources

- Height Maps from the lab.
- All mesh models from https://www.textures.com/library
- JPG to TGA converter https://www.freeconvert.com/jpg-to-tga