Cosc 422 Assignment 1 Report

Name: Gongzai Li **Student ID:** 31751890

Basic Terrain Model

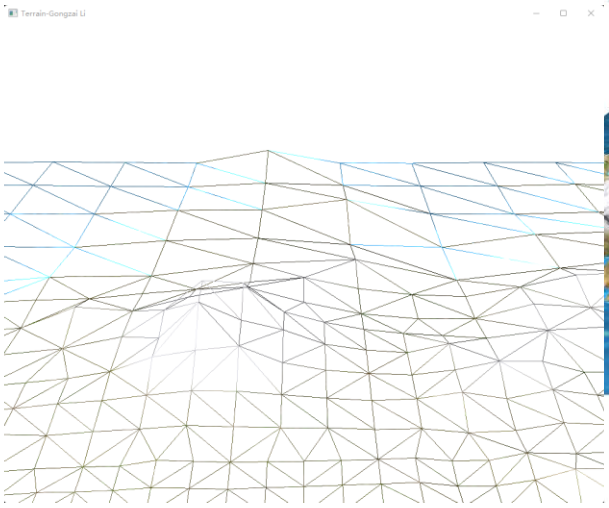


Figure 1 – Dynamic level of detail

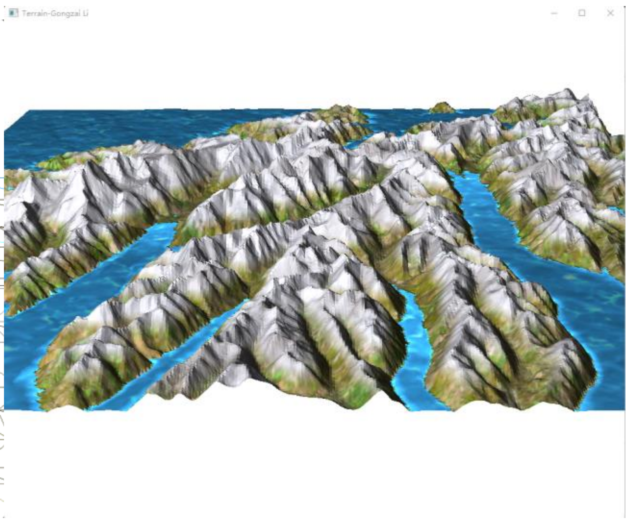


Figure 2 – Scene 1 (Push Key '1')

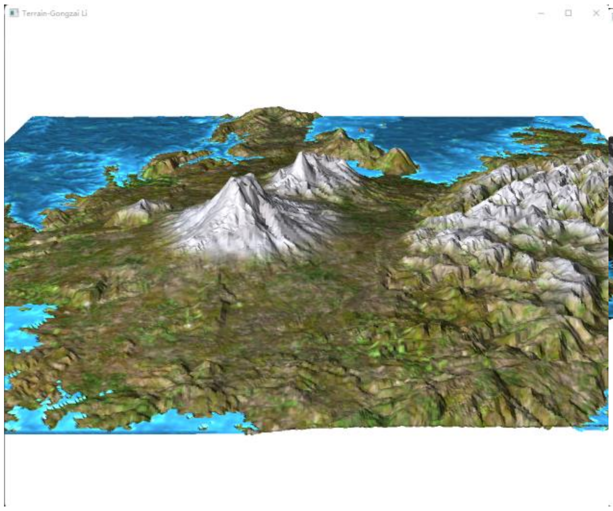


Figure 3 – Scene 2 (Push Key '2')

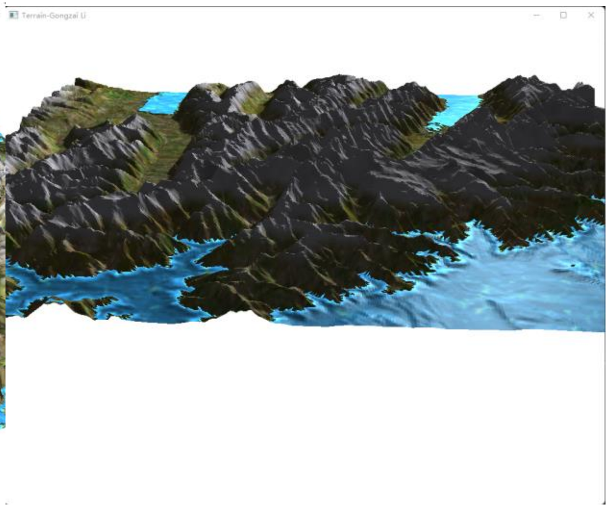


Figure 4 – Scene 1 (The Shadow Behind the Mountain)

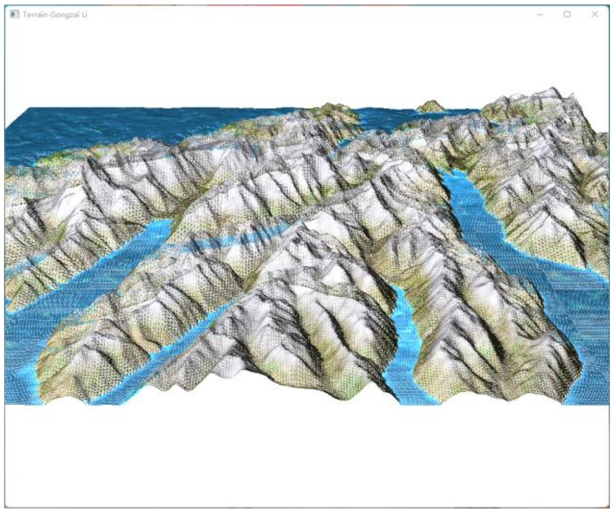


Figure 5 – Scene 2 with wireframe (Push Key 'space')

1. Figure 1 shows the dynamic level of detail. It can be seen in Figure 1 that dynamic level of detail algorithm which provides a higher tessellation for patches near the camera and a lower tessellation for patches located farther away.
2. Figure 2 and Figure 3 show the lighting (ambient and diffuse term). Figure 4 shows the shadow behind the mountain.
3. Figure 2 and Figure 3 show that it uses 'Water', 'Snow' and 'Grass' textures to map terrain by different heights. Water has flat surfaces. Mixing snow and grass textures at the interface of snow and grass.
4. Figure 2 shows scene 1 and Figure 3 shows scene 2. (You could push key '1' or '2' to switch scene).
5. Figure 5 shows the wireframe model.

Extra Features

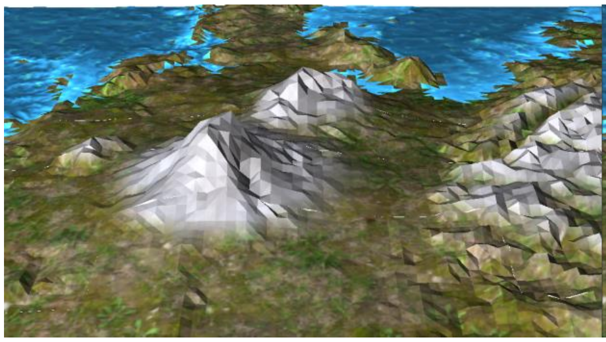


Figure 6 – Scene 2 with Cracking

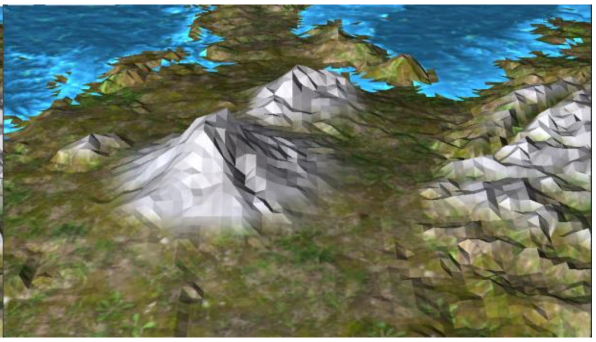
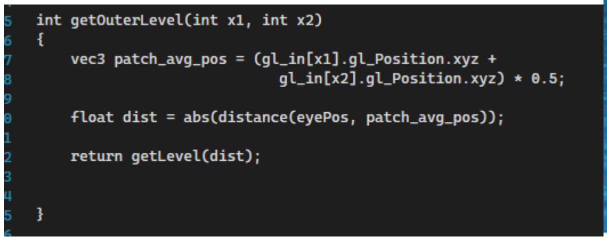


Figure 7 - Scene 2 with no Cracking



```
int getOuterLevel(int x1, int x2)
{
    vec3 patch_avg_pos = (gl_in[x1].gl_Position.xyz +
                         gl_in[x2].gl_Position.xyz) * 0.5;

    float dist = abs(distance(eyePos, patch_avg_pos));

    return getLevel(dist);


}
```

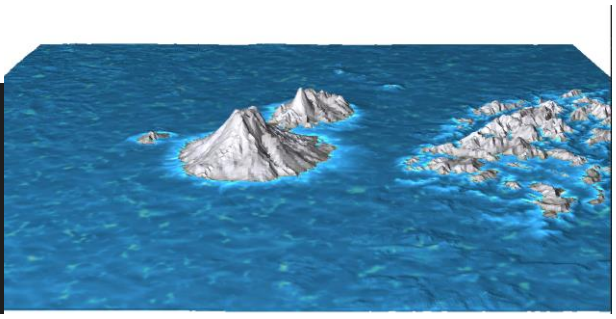
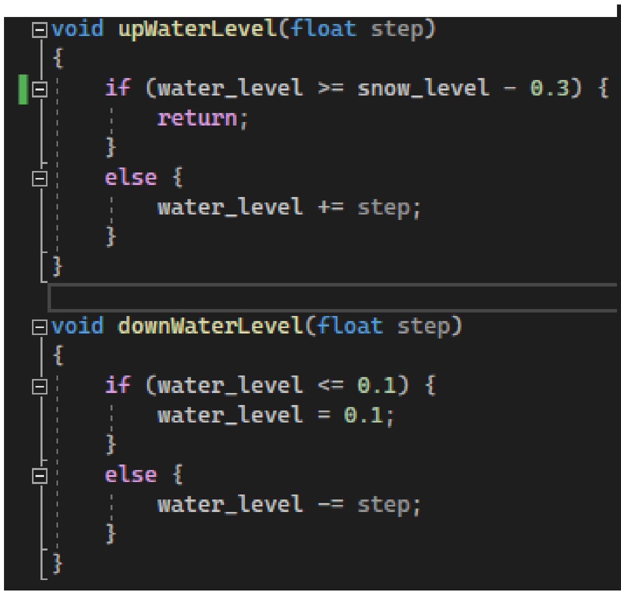


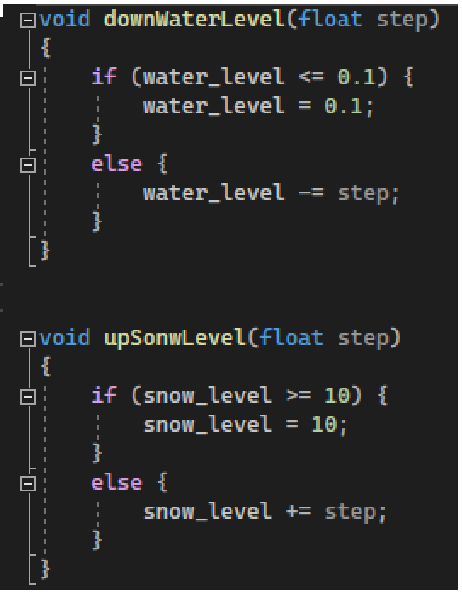
Figure 8 - Code for Fixing Cracking

Figure 9 – Scene 2 with Adjustable water levels



```
void upWaterLevel(float step)
{
    if (water_level >= snow_level - 0.3) {
        return;
    }
    else {
        water_level += step;
    }
}

void downWaterLevel(float step)
{
    if (water_level <= 0.1) {
        water_level = 0.1;
    }
    else {
        water_level -= step;
    }
}
```

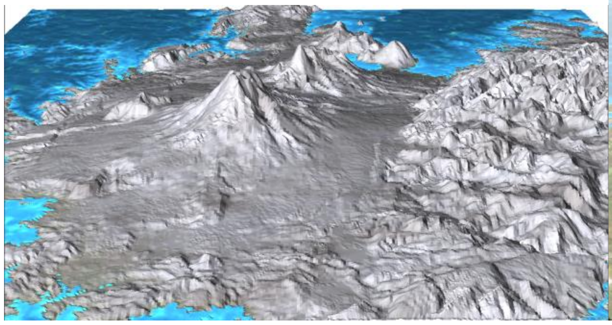


```
void downWaterLevel(float step)
{
    if (water_level <= 0.1) {
        water_level = 0.1;
    }
    else {
        water_level -= step;
    }
}


void upSonwLevel(float step)
{
    if (snow_level >= 10) {
        snow_level = 10;
    }
    else {
        snow_level += step;
    }
}
```

Figure 10 - Code for Adjustable water levels

Figure 11 – Code for Adjustable snow levels



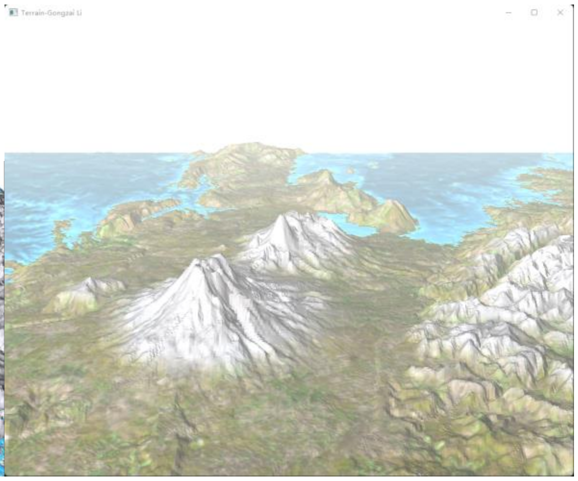
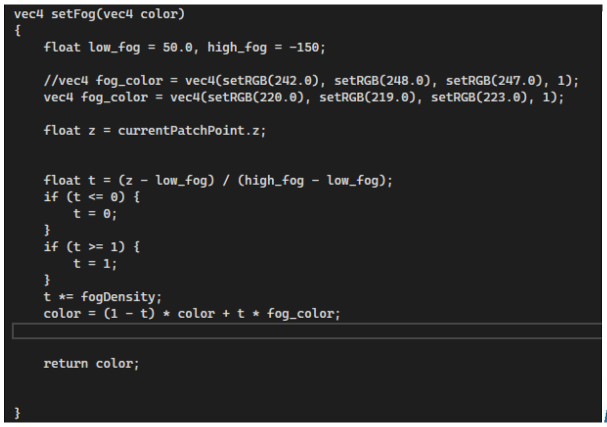


Figure 12 – Scene 2 with Adjustable snow levels

Figure 13 – Scene 2 with fog (Push Key 'f' to toggle fog)



```
vec4 setFog(vec4 color)
{
    float low_fog = 50.0, high_fog = -150;

    //vec4 fog_color = vec4(setRGB(242.0), setRGB(248.0), setRGB(247.0), 1);
    vec4 fog_color = vec4(setRGB(220.0), setRGB(219.0), setRGB(223.0), 1);

    float z = currentPatchPoint.z;


    float t = (z - low_fog) / (high_fog - low_fog);
    if (t <= 0) {
        t = 0;
    }
    if (t >= 1) {
        t = 1;
    }
    t *= fogDensity;
    color = (1 - t) * color + t * fog_color;


    return color;


}
```

Figure 14 – Code for creation of fog

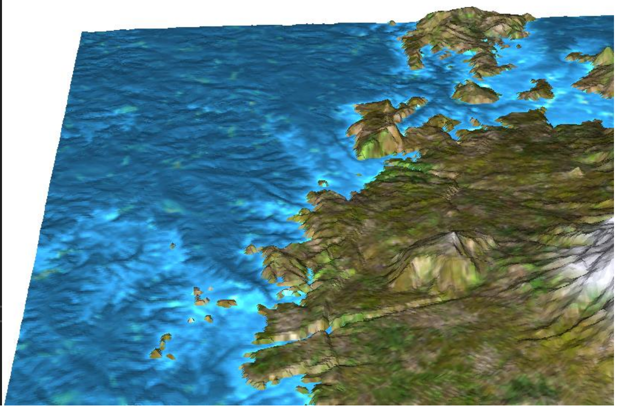
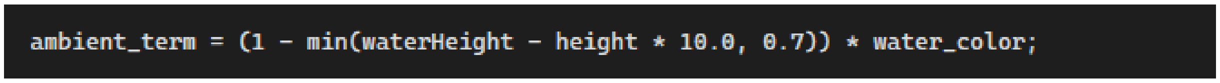
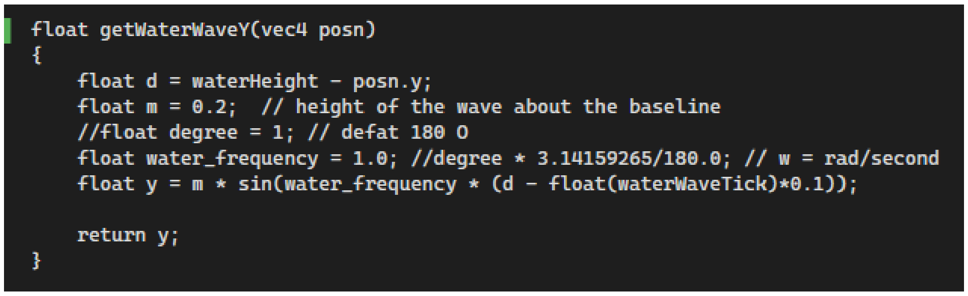


Figure 15 – Scene 2 with Water depth color and water wave



```
ambient_term = (1 - min(waterHeight - height * 10.0, 0.7)) * water_color;
```

Figure 16 – Code for creation of water depth color



```
float getWaterWaveY(vec4 posn)
{
    float d = waterHeight - posn.y;
    float m = 0.2;  // height of the wave about the baseline
    //float degree = 1; // defat 180 O
    float water_frequency = 1.0; //degree * 3.14159265/180.0; // w = rad/second
    float y = m * sin(water_frequency * (d - float(waterWaveTick)*0.1));

    return y;
}
```

Figure 17 – Code for creation of water waves

1. **Cracking:** I used the outer tessellation levels computed using the distance of the centre of the corresponding edge from the camera to fix the cracking problem (The code shows Figure 8). The ‘c’ key toggles cracking or non-cracking (Figure 6 and 7 shows cracking and non-cracking).

2. **Adjustable water levels:** Adjusting water level can be seen in Figure 9 and the code for implement can be seen in Figure 10. Where key ‘q’ increases and key ‘a’ decreases the level of water.

3. **Adjustable snow levels:** Adjusting snow level can be seen in Figure 12 and the code for implement can be seen in Figure 11. Where key ‘w’ increases and key ‘s’ decreases the level of snow.

4. **Fog:** Where key ‘f’ to tagger fog. Figure 13 shows up. Figure 14 shows code for creating fog. The fog will change according to the time its concentration will change。

5. **Water feature:** Figure 15 shows the water depth colour. Figure 16 shows the code for creating water depth colour. Figure 17 shows the code for making water waves (Hard to show water waves in an image).

Keyboard functions list

Key	Function
↑	Move forward
↓	Move backward
←	Turn left
→	Turn right
Space	Toggle between wireframe and solid fill
1	Display height map MtCook
2	Display height map MtRuapehu
q	Increasing water level
a	Decreasing water level
w	Increasing snow level
s	Decreasing snow level
f	Toggle fog
c	Toggle cracking
v	Toggle water wave