COSC 422 – Assignment 1 – Terrain Renderer

Maxwell Clarke – 72568156

Fri 24th Aug 2018

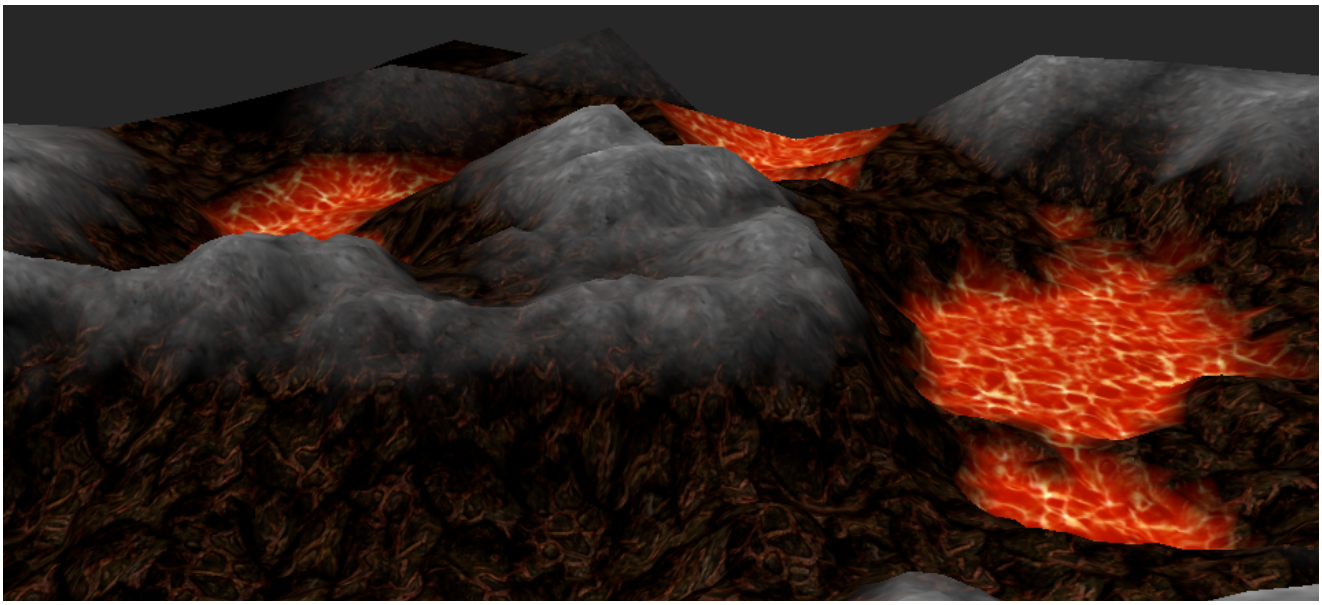

Illustration 1: Map 1

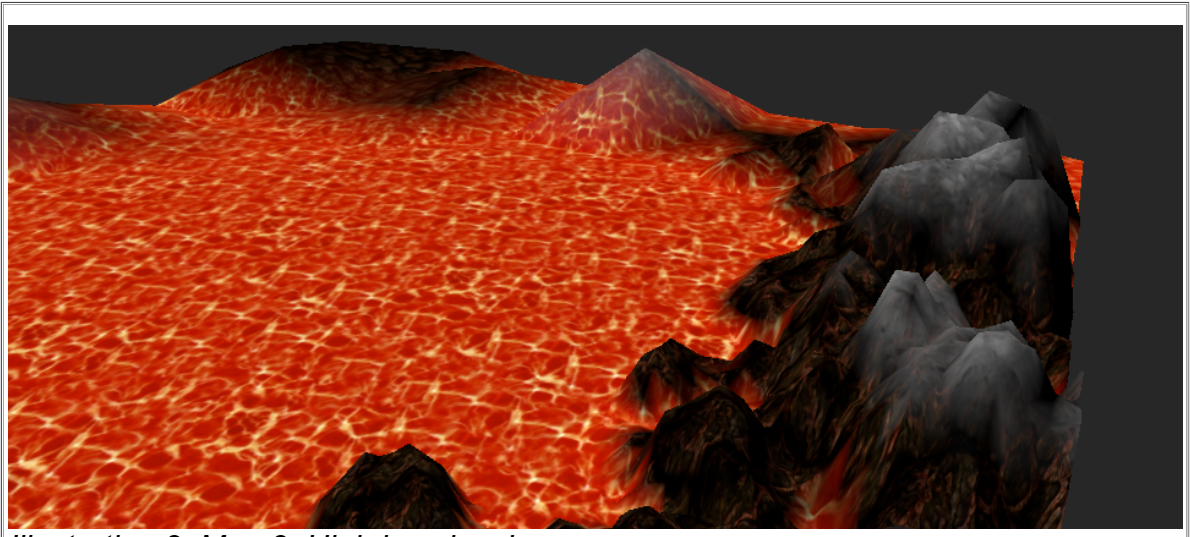

Illustration 2: Map 2: High lava level

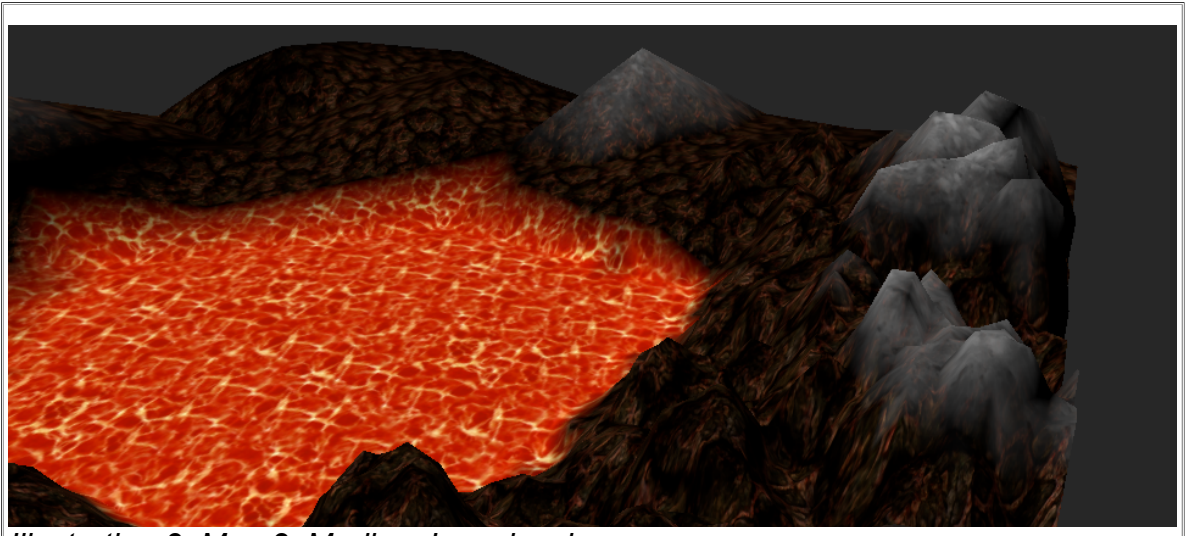

Illustration 3: Map 2: Medium Lava level

Illustration 4: Map 2: Low Lava level

1 Description

1.1 Terrain

Map 1 is created from a Perlin noise fractal image obtained online. Map 2 is created from a similar greyscale image which was created to be used as game heightmap, for Open Roller Coaster Tycoon 2.

1.2 Images

The terrain has 3 textured layers. The lowest is Lava, the middle is a dark volcanic Stone, and the highest is Ash.

Blending has been implemented between Lava and Stone, and between Stone and Ash. However blending has not been implemented between Lava and Ash, which causes an abrupt change when the Lava is raised to the Ash level.

1.3 Dynamic Level-of-Detail

Dynamic level of detail has been implemented. The tessellation level function was implemented as the following function, where dist is the distance from the camera in world coordinates. This equation is exaggerated for demonstration purposes – it reduces the level to 1 at distance 100 – in a real application a more shallow gradient would be used.

```
int level(float dist) {
   return int(max((100 - dist)/5, 1));
}
```

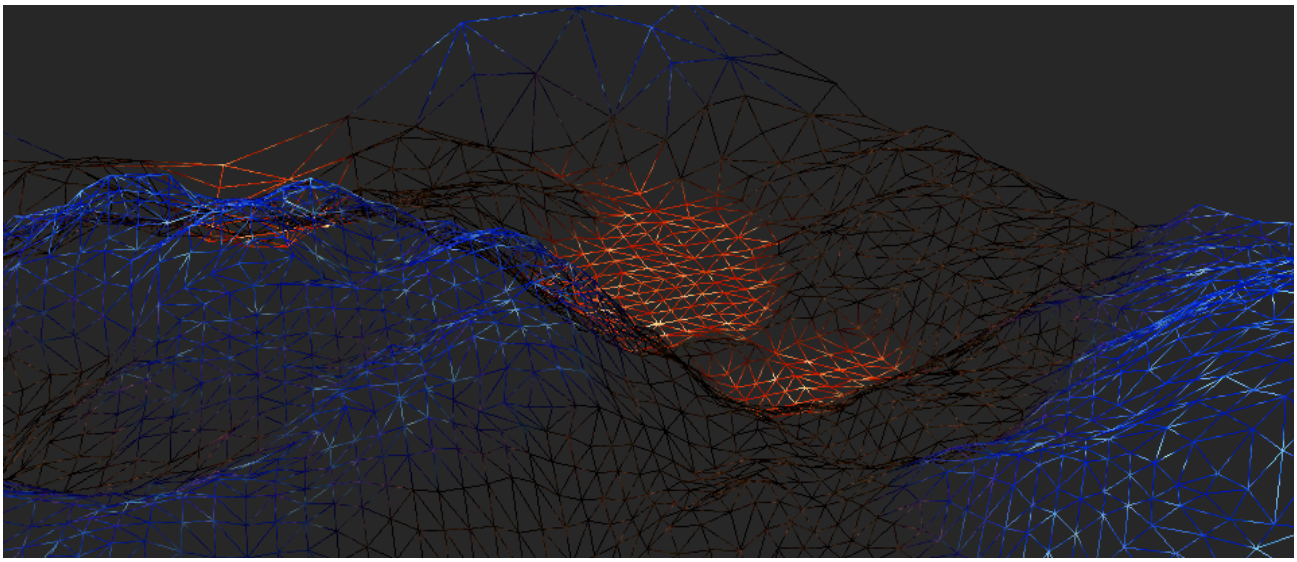

Illustration 5: Line mode, showing lower tesselation levels further from the camera.

1.4 Prevention of Cracking

Cracking artifacts are prevented by setting the gl_TessLevelOuter value differently for each edge of the patch. The average position of the edge is used in the equation from 1.3 to calculate the tessellation level. For the inner tessellation levels, the average position of all four patch vertices is used. The distance from the camera is determined by passing the camera position into the Tessellation Control Shader as a uniform.

1.5 Lighting

Lighting has been implemented. The lighting model considers only Diffuse and Ambient terms. A fixed lighting direction was used – this simulates a light source at an infinite distance.

The normals were calculated per-vertex, rather than per-primitive. This resulted in a more smooth appearance of the terrain, due to the normals being interpolated when passed to the fragment shader.

To achieve this, two extra points were calculated in the geometry shader when sampling the heightmap. For each vertex, a second and third point were created by adding a small delta to the X and Y coordinates respectively and sampling the heightmap based on these locations. Using these three points to calculate the vertex normal gave a more accurate normal than calculating the normal from the three vertices in the primitive.

Lastly lighting was disabled for the lava, as lighting artifacts looked undesirable around the edge of the lava. This was achieved by reducing the effect of the lighting based on the texture weight of the lava (in the fragment shader).

1.6 Rust

The program is implemented using the Rust programming language. I have submitted an x64 Linux binary along with the source code and report. If this does not work, you can contact me. It is also simple to set up a Rust development environment on a CSSE Lab machine by following the instructions here https://www.rust-lang.org/en-US/install.html. For convenience my source code can also be found here https://github.com/Maxeonyx/terrain-generator

2 Controls

I and O to raise and lower Ash level.

K and L to raise and lower Lava level.

H to toggle heightmap.

P to toggle GL_LINES mode.

Arrow keys to turn and move.

3 References

https://commons.wikimedia.org/wiki/File:PerlinNoise2d.png

https://openrct2.org/forums/uploads/monthly_2017_06/scene2.png.72fadcf00adf7c876f53f2fc0b4a85e6.png

https://opengameart.org/content/volcano-lava-floor

https://opengameart.org/content/golgotha-effects-textures-lavafulljpg

ash_brown.png was created and eaten by the author.