Computer Graphics

Project Report

Group members :

Mikael Morales Gonzalez 234747
Charles Parzy Turlat 250628
Volodymyr Loyko 271690

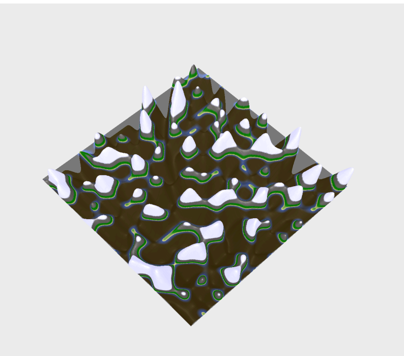
Part1

Introduction

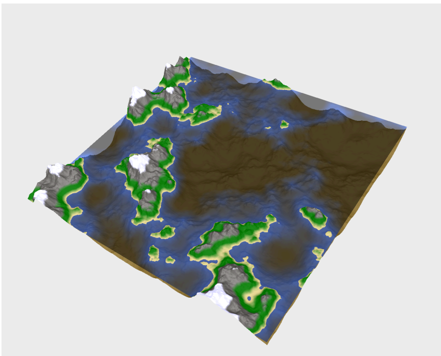
We generated a height map using a noise function. Then we passed the height map to the terrain, which is initially flat. After that, in the terrain shader, we set the height of every vertex using the value stored in the height map.

1) The Noise Function

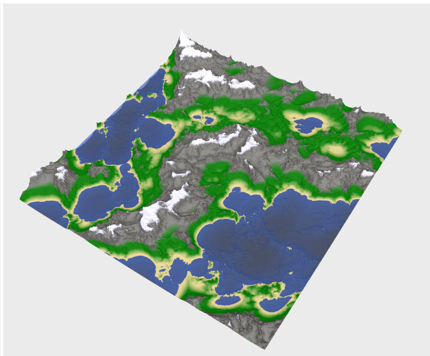
At first, we implemented the Perlin Noise function using the provided documentation. For now, the permutations and the gradients are hard coded in the shader. In order to improve the Noise function, we implemented the Fractal Brownian Motion function which use the Perlin Noise function. But to get a better result, we implemented two more functions: Hybrid Multifractal and Ridge Mutlifractal. To know about these methods, we read the paper about "procedural fractal terrains"[1]. To get the correct coefficient, we bound some keys that allow us to see realtime changes of the terrain. We decided to use the Ridge Mutlifractal function since it gives better results, as you can see below.



Fractal Brownian Motion



Hybrid Mutlifractal



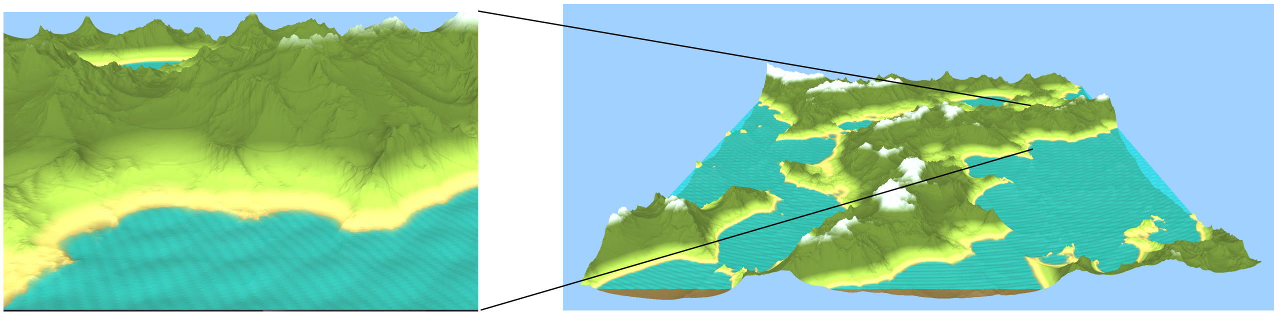
Ridged Multifractal

2) The Water

To generate the water, we draw a flat terrain at a fixed height, which is consistent with the transition from the sand to the seabed.
The terrain is rendered on top of this flat terrain, and to avoid having water under a mountain, we set the color to transparent if it is below a mountain.

---

[1] https://www.classes.cs.uchicago.edu/archive/2015/fall/23700-1/final-project/MusgraveTerrain00.pdf



Waves on the water quad

3) Waves

To make the initial flat water grid look like water we used the Gerstner wave function. We first initialized the attributes for 5 wave functions, 3 of them circular and 2 directional and then calculated the height and new x and y values. The only difference between the circular and directional waves apart from constant values was that the direction vectors for the circular waves depended on the x and y values. The difference between the Gerstner wave and the normal sine wave is that not only does the height value changes, but also the x and y values do too. This gives an illusion of water particles riding the wave and not just hovering up and down. The combination of multiple waves also adds to the immersion factor. The water waves are shaded in the same way as terrain. Where the water met the sand surface, the water plane matches the shape of the sand and is slightly above it. In this region the water blue is faded out, to look like a wave on the beach.

The formula for the height is as follows: a * sin(w * (x,y) • (d1,d2) + (time * o)) - a. Where a is the amplitude of the wave, w is wavelength, o is the speed, and (d1, d2) is the direction vector. The formulas for new x and y are almost identical to the height formula except for few key differences: a * q * d * cos(w * (x,y) . (d1,d2) + (time * o)). Where the new variable q is the roundness of the wave and is in range (0,1) though it is advised for it to be in the lower range to not add “loops” at the top of your waves. We also changed the basic sin(…) in the height formula to 2 * ((sin(…) + a)/2)^1.5. This change made the sides of each bump on the wave steeper.

4) Terrain colors

We chose five colors corresponding to the different materials. To render the colors, we used the techniques seen during the lab sessions. So, the colors are defined by three values : the ambient one, the diffuse one and the specular one.
To get the values we have now, we played with the values.

5) Water colors

To get a more realistic effect, we darkened the water depending on the depth of the seabed. To do that, we used the mix function explained during the lab sessions.

6) Workload

Mikael Morales Gonzalez - height map generation - 33.(3)% of total workload.
Volodymyr Loyko - water quad and wave generation - 33.(3)% of total workload.
Charles Parzy Turlat - terrain and water realistic colorization - 33.(3)% of total workload.

It is hard to determine more specific percentages of workload since usually we worked together on each part, having an equal input. The list above is more of an approximation of who contributed most to each part of the project, even though everyone had a hand in everything.