Computer Graphics

Project Report

Group members :

Mikael Morales Gonzalez	234747
Charles Parzy Turlat	250628
Volodymyr Loyko	271690

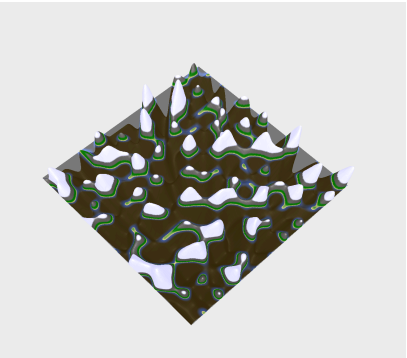
Part1

Introduction

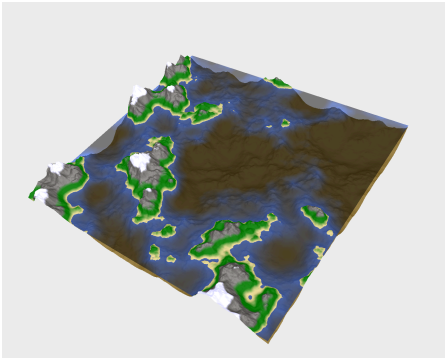
We generated a height map using a noise function. Then we passed the height map to the terrain, which is initially flat. After that, in the terrain shader, we set the height of every vertex using the value stored in the height map.

1) The Noise Function

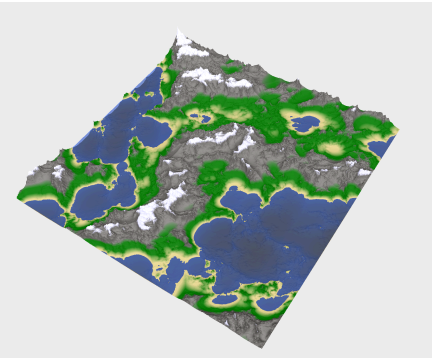
At first, we implemented the Perlin Noise function using the provided documentation. For now, the permutations and the gradients are hard coded in the shader. In order to improve the Noise function, we implemented the Fractal Brownian Motion function which use the Perlin Noise function. But to get a 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



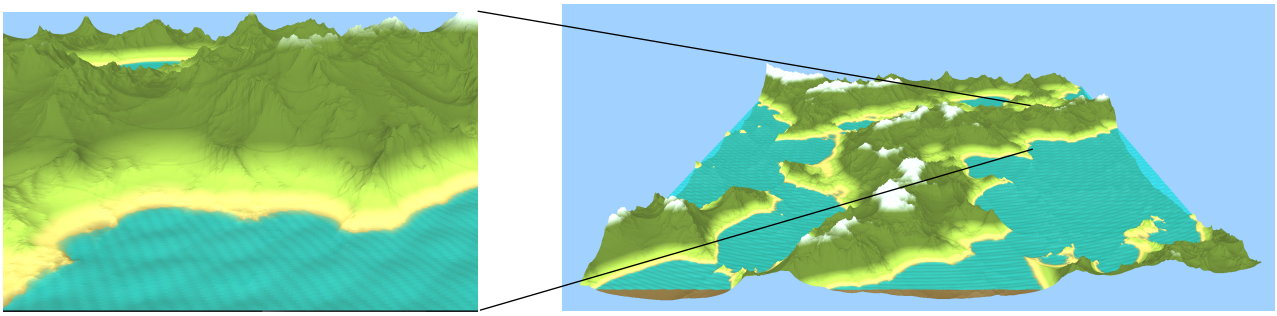
Ridged Multifractal

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

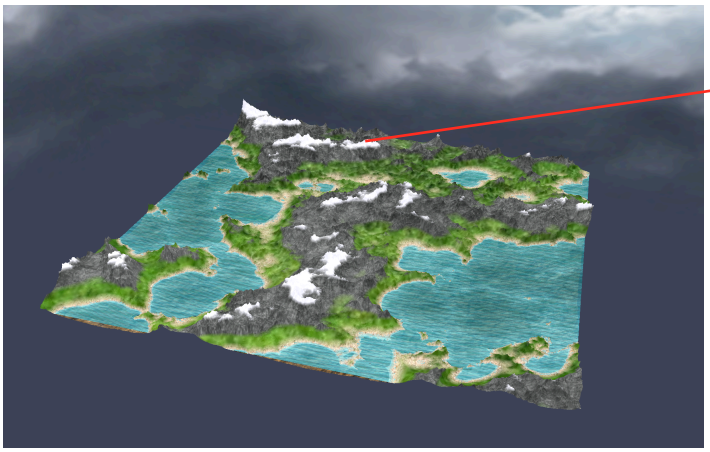
Mikael Morales Gonzalez - height map generation - 33.(3)% of total workload.
Volodymyr Loyko - water quad and wave generation - 33.(3)% of total workload.
Charles Parzy Turlat - terrain and water realistic colorization - 33.(3)% of total workload.

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.

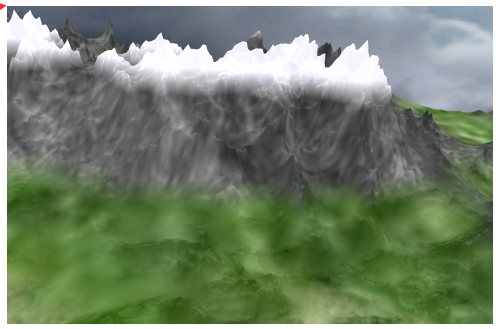
Part2

1) Texturing

We added six different textures to improve our previous result with colors. In order to have a better transition between the textures, a gradient is computed such that the textures disappear progressively. Our textures are: grass, rock, sand, seabed and water.



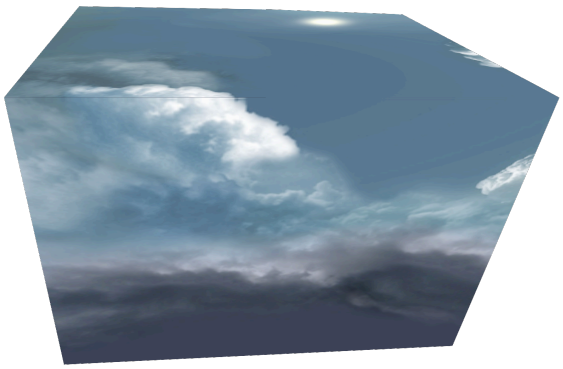
Current state of the terrain



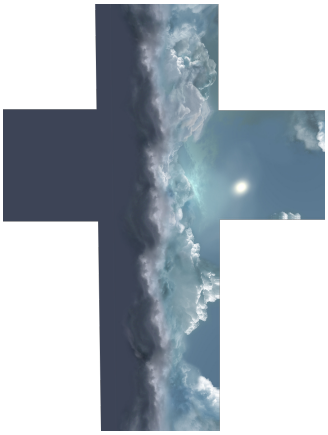
Transition between textures

2) Sky modeling

To model the sky, we defined a class called skybox, which represent a cube with a scale factor much bigger than the size of the terrain. Since the cube is bigger than the terrain, the terrain is inside the cube, giving the impression of a nice sky.
To model the sky, we used a texture that is directly put on the cube.



Cube containing the terrain



Texture of the sky

To model the sky correctly, we use the same techniques that were used in the homework The methods applied to model the sky are taken from the homework 4, but with some adjustments that were required for this specific part.

3) Terrain normals improvement

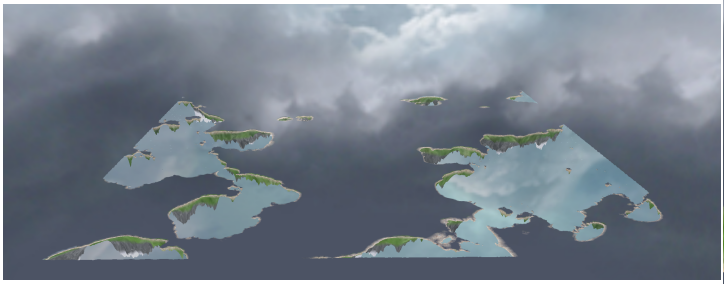
We improve the way we computed the normal when generating the terrain. Before, we used the two function dFdx & dFdy, but the result was not good, as we could see the triangles in our terrain.

To obtain a better result, we are now computing the normals by creating two vectors. These vectors are computed by taking, for every pixel, one pixel above in x (respectively y) direction and one pixel below in x (respectively y) direction. Then we compute the cross product of these two vectors, producing the normal at the pixel position. This technique produce a better result.

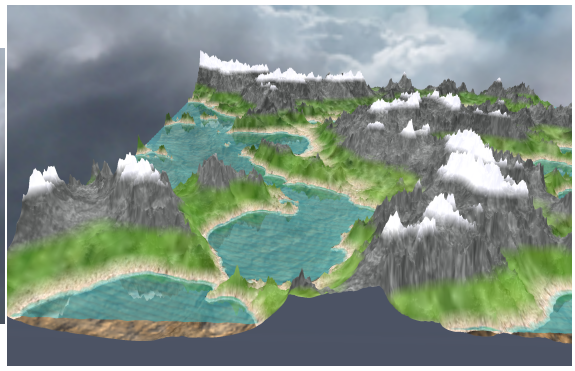
4) Water

This time we tackled the water reflections. Conceptually, the way we generate reflections is we look at the world if we were in the water and looking out. To do this we flip the terrain and sky and only render the parts that are visible through the water to a texture using a FBO. We then mix this texture in when we are rendering the water to simulate the reflection. This provides us with a simulation of a reflection of terrain and sky on perfectly still water. While this works well enough, the more realistic/immersive rendition of reflection can be achieved with some distortion of the textures. While this can be done relatively easily on smaller scale, computing the wave normals for the gerstner waves is expensive, and on a larger scale is not feasible. We have not implemented a way to deal with this yet, thus we just left the reflections absolutely perfect for now.

In the future we plan to generate a Simplex noise tillable cube and use FFT to extract the constants for our Gerstner wave equation. This will allow us to have predetermined wave pattern which in turn will allow us to generate a wave normal map on cpu and supply it to the shaders so it can distort the reflection texture in a correct way.



The reflection texture used as the sole texture for the water



The terrain + water + reflection + skybox render

5) Work in progress

We are currently working on adding a lighthouse on the terrain, in order to have a nice feature for the next part. What we would like to do, is to be able to turn off the main light source and turn on a rotating light source inside the lighthouse to simulate the behavior of a lighthouse at night. Unfortunately, this part is not finished yet, as we have some difficulty with the library to import the object.

6) Workload

Mikael Morales Gonzalez - Sky modeling & terrain normals - 33.(3)% of total workload.
Volodymyr Loyko - Water - 33.(3)% of total workload.
Charles Parzy Turlat - Texturing - 33.(3)% of total workload.

As in part 1, it is hard to determine more specific percentages of workload since we worked together on each part. 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.