Part 2 – Terrain Rendering

Éloi Alain Enguerrand Granoux Josselin Held

Thursday 4th May, 2017

Contents

1 Introduction **1**

2 Mandatory parts **1**
2.1 Realistic texture . . . 1
2.1.1 Mapping with interpolation, randomness and steep slopes . . . 1
2.1.2 Improving steep slopes and realistic beaches . . . 1
2.2 Skybox . . . 2

3 Optional parts **2**
3.1 Infinite terrain . . . 2
3.1.1 Terrain movement . . . 2
3.1.2 FPS camera . . . 2
3.2 Water reflection . . . 2

4 Fraction of the Workload **3**

5 Screenshot **3**

1 Introduction

2 Mandatory parts

2.1 Realistic texture

2.1.1 Mapping with interpolation, randomness and steep slopes

Main contributor: Enguerrand.

At first, we use the height to set up textures (sand, grass, rock and snow). Then we use some mix to have more realistic texture. The coefficient of the mix depends on the inclination and on the height. We also add a time dependence to represent the seasons and make the snow disappear.

2.1.2 Improving steep slopes and realistic beaches

Main contributor: Éloi.

The condition on the inclination – computed with `dFdx` and `dFdy` – only introduced pixel sparkling at the borders. In order to reduce the effect, I changed the condition structure to focus on water vs grass vs snow. The sand and rocks are then mixed with the grass – using custom interpolation curves – depending on the inclination of the slope (and altitude for the sand). Beaches do not occur on steep slopes.

2.2 Skybox

Main contributor: Éloi.

At first, I tried to map the cube pattern (presented during the class) onto the box using `glDrawElements`. Unfortunately, some vertices ought to be mapped to several points on the pattern.

Eventually, following the instructions on `learnopengl.com`, the result was better. The use of `GL_TEXTURE_CUBE_MAP` was determining.

3 Optional parts

3.1 Infinite terrain

3.1.1 Terrain movement

Main contributor: Éloi.

When pressing the W, A, S and D keys the terrain moves continuously with time in the view direction. The principle used is to compute the noise at a position `(x, y)` on the grid that changes with time. This offset is now computed in `main.cpp` where key actions are listened to. Note that the projection of the view direction onto the vertical axis is set to zero so the camera remains high and centered.

3.1.2 FPS camera

Main contributor: Éloi.

The user may dynamically (while pressing aforementioned movement keys) change the view direction with the mouse. The trackball mechanism has been changed completely. The zoom feature now moves the camera in the view direction (this time, the camera position in the world changes, so it is literally possible to go anywhere in the world: under the terrain, outside the skybox).

3.2 Water reflection

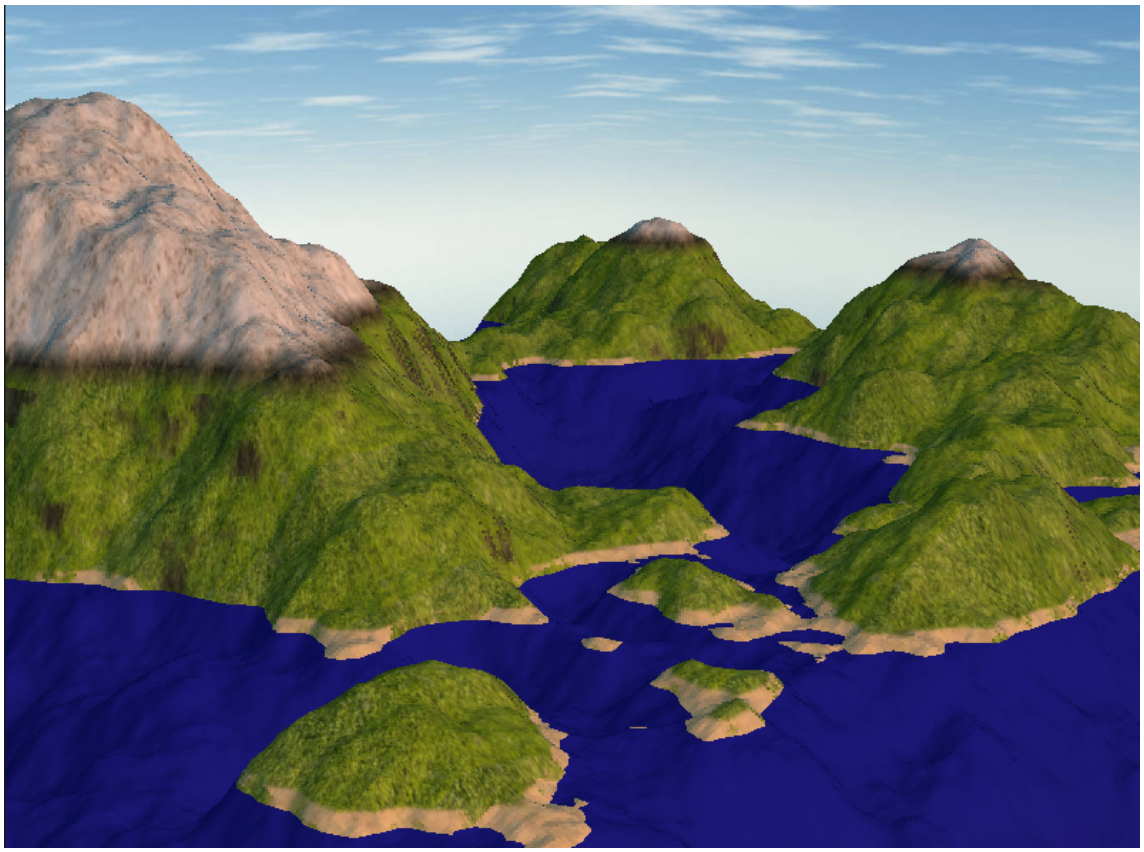
Main contributor: Josselin.

We are implementing water reflection. We used the same concept as in the Homework 4, namely reflecting the camera under the water. We then draw a second time the now reflected terrain in a framebuffer, which is sent to a `water.h` class, which draw a second plan at height zero representing the water with the reflection.

4 Fraction of the Workload

1. Éloi Alain: 40%
2. Enguerrand Granoux: 30%
3. Josselin Held: 30%

5 Screenshot



Waiting for water reflection