Part 1 – Procedural Terrain

Éloi Alain Enguerrand Granoux Josselin Held

Thursday 13th April, 2017

Contents

1 Introduction

The starting point of our project is the homework 2 (trackball). We deleted the cube and kept our work for the grid.

2 Mandatory parts

2.1 Noise

Main contributor: Éloi.

The Perlin noise was implemented following the instructions of the slides.

A fixed (prime) number of random gradients are generated by CPU once and for all. These are available to the shader as a uniform vec2 array variable. Gradients are then thoroughly selected in the array using an offset, the cell coordinates and modulo.

To create the fractal Brownian motion, we added the contributions of our noise function using distinct frequencies. To obtain a realistic terrain, we had to set a particular amplitude for each contribution. The higher the frequency, the more little the contribution.

2.2 Framebuffer

Main contributor: Josselin.

2.3 Applying altitude

Main contributor: Enguerrand.

2.4 Shading

Main contributor: Éloi.

The implementation is very similar to the homework on shading. The only difference that is worth mentioning is the way of computing the normal vectors as these are not directly available. In both following methods, we make use of `dFdx` and `dFdy`.

The first implementation used "real" position of the fragment, inherited from triangle interpolation. The issue is that this method requires many triangles to be drawn to prevent the user from visualizing them.

The second implementation reads the height of the fragment (given in the heightmap). From then on, to avoid aliasing, the linear interpolation must be activated when reading data from the heightmap.

2.5 Simple terrain texture

Main contributor: Enguerrand.

3 Optional parts

3.1 Infinite terrain (not completed yet)

To give the impression of infinite terrain, we would place the camera right above the center of the grid. To "move" in the terrain, the user would press the arrow keys. Technically, when pressing the arrow keys, the position of the cells (heightmap) change with time. The cell position change should depends of the view vector.