Generating Mountains Using Hydraulic Erosion

Mika Rebensburg* Mathilda Larsson†

Lund University
Sweden

Abstract

Generating mountain landscapes can for example be done to create a more immersive experience in video games or in other applications. One way of doing so is by approximating the real erosion of mountains due to rain by simulating thousands of raindrops and how they shape the landscape. In this project, hydraulic erosion has been implemented to create a realistic mountain landscape, enhanced by basic shading and normal mapping.

1 Introduction

The aim of this project was to create a mountain landscape using hydraulic erosion and to make the scene as realistic as possible. While creating mountains with more basic techniques (such as using noise to create the initial landscape, which was the first step of our project), the landscape can be improved upon by simulating how mountains are shaped in nature - by erosion from raindrops.

2 Frameworks

The algorithms were implemented using the OpenGL API together with "bonobo", an extension framework used in the course.

3 Algorithms

We first generate a simple quad shaped mesh of triangles. Then, we change the y-values of the vertices according to heights calculated using noise which we have based on similar calculations by [McGuire 2014]. To get a more realistic landscape, we have layered three noise maps with three different frequencies on top of each other. Using this technique, a good baseline for the mountain shapes were obtained, see fig. 2

3.1 Hydraulic Erosion

The next part is simulating raindrops eroding the landscape. The basic idea is that a raindrop is placed at random on the landscape with a capacity to carry sediment and a certain lifetime before it is evaporated [Lague 2019]. The droplet's height is calculated along with the direction in which it is flowing based on bilinear interpolation of the heights of the surrounding vertices. The raindrop is then moved 1 unit in its direction if the direction is larger than zero and if the new position is still within the defined bounds of the landscape. The direction is calculated according to eq. 1, where the previous direction of the raindrop dir_{new} is taken into account along with the gradient g and the factor $p_{inertia} \in [0, 1]$ [Beyer 2015].

$$dir_{new} = dir_{prev} \cdot p_{inertia} - g \cdot (1 - p_{inertia}) \quad (1)$$

The new position pos_{new} is calculated according to eq. 2 based on the old position pos_{old} and the calculated direction dir_{new}.

$$pos_{new} = pos_{old} + dir_{new} \quad (2)$$

If the drop is still viable to continue (i.e. within the defined boundaries of the landscape), the height for its new position is calculated. At this point, it is determined if the droplet is to deposit part of the sediment it is carrying or if it should erode. The droplet will deposit if the difference between its new height and old height is positive or if the sediment it is carrying is larger than its capacity. For example, the sediment capacity will be large if the droplet is moving down a steep slope and contains a lot of water. A deposit of sediment is made by decreasing the droplet's sediment and distributing it over the surrounding vertices (i.e. increasing the height of the vertices). Erosion is performed by decreasing the height of the surrounding vertices and adding to the sediment carried by the droplet. Finally, the droplet's speed is updated and a fraction of its water is evaporated according to eq. 3 and 4, where $p_{evaporation} \in [0, 1]$, h_{new} and h_{old} are the new and old heights and $gravity$ is an adjustable parameter [Beyer 2015].

$$speed_{new} = \sqrt{speed_{old}^2 + (h_{new} - h_{old}) \cdot gravity} \quad (3)$$

$$water_{new} = water_{old} \cdot (1 - p_{evaporation}) \quad (4)$$

This process is repeated for a set number of iterations by defining the number of raindrops wanted for the erosion.
In order to perform better erosion, the number of surrounding nodes affected is based on a parameter called the erosion radius. Before performing the hydraulic erosion described above, each vertex in the landscape is assigned a number of surrounding vertices based on this radius. These will be affected and to different degrees when a drop erodes around a certain vertex.

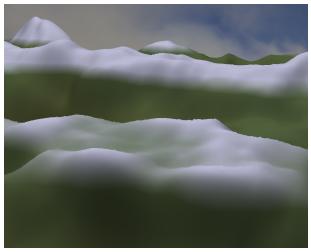
3.2 Texturing and Shading

Our shading was applied fully in the fragment shader. Diffuse lighting was used for illuminating the scene and different colours were blended to apply texture. For blending the colours of the mountains, we used the built in OpenGL method `mix`, that takes two colours and one value between 0 and 1 that determines the weight of each colour. First, we blend between a grey and a green, depending on the altitude and steepness of the fragment. A high steepness and a high altitude will result in a less green and more grey fragment colour, which simulates the growth behaviour of grass.

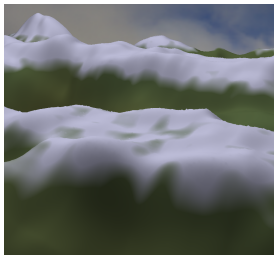
To make the scene more diverse and less uniform, we mix in a brown colour to parts of the grass, using noise again.

On the top of the highest peaks there is snow colouring, which is based on the height of the fragment. Above a certain height threshold, the white snow colour will be blended into the grey mountain colour. The height that determines whether to place snow and how to blend it is based on a noise map but lies between certain boundaries. This helps avoiding having the snow blend the same everywhere which looks unrealistic, see Figure 1.

*m.rebensburg@campus.tu-berlin.de
†ma7472la-s@student.lu.se



(a) Snow without noise



(b) Snow with noise

Figure 1: Snow generation

To make the surface look less smooth, we added a normal map that was made for shading a rocky surface. Finally, to place the mountain landscape within a context, a simple skybox was used based on the code written in the previous course EDAF80.

4 Results

For our final mountain picture, we used a mesh with a width and height of 1000 units and 1000x1000 vertices. The hydraulic erosion was implemented on the CPU. The parameters of the hydraulic erosion were manually adjusted to give a good looking result.

The calculation of 600,000 raindrops on a laptop with an Intel Core i7-9750H CPU with 2.60GHz takes 19.6 seconds.

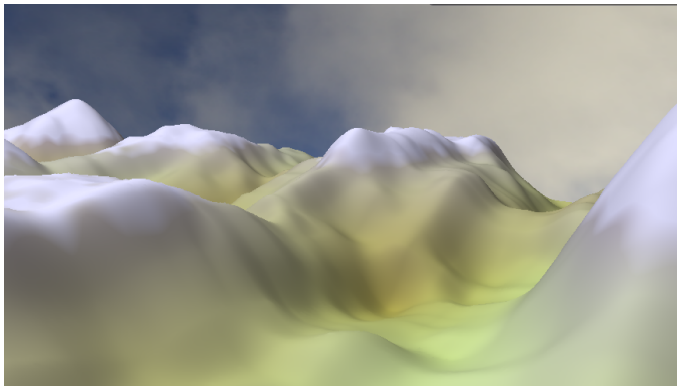


Figure 2: Mountain image without hydraulic erosion and normal map

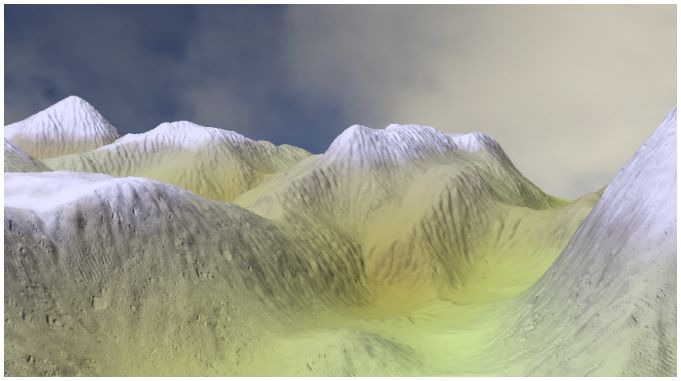


Figure 3: Final mountain image with all mentioned improvements

5 Discussion

The hydraulic erosion makes the surface more rough and therefore more realistic looking.

Real mountains consist of different materials that have different erosion properties, e.g. massive rocks will not erode nearly as much as soil. One could therefore think about adding different erosion properties to different parts of the map.

While the computation of the landscape takes almost 20 seconds, hydraulic erosion and landscape generation often only needs to be performed once during development when one wants to get a heightmap that is then saved for later use without computation. Instead of calculating the hydraulic erosion on the CPU, performing the calculations on a GPU using a compute shader could have a positive impact on the performance. Here, parallelization could be used to improve performance.

Concering the realism of the landscape, there is much room for improvement. To give a more realistic look, one can add fog, water, better textures or generate plants such as trees.

6 Conclusion

In this report we have explored the possibilities of a specific hydraulic erosion implementation and discussed some techniques to make the landscape look more realistic.

There is also an inherent challenge in a project like this where the objective measurements are restricted to how long the computations take while there is no clear answer to the subjective question ”Is this a good-looking mountain landscape?”. There are many ways to tune parameters in this project, such as the number of raindrops and the gravity for the hydraulic erosion, but also which colours and which normal maps to use for the shading.

References

BEYER, H. T. 2015. Implementation of a method for hydraulic erosion. *Bachelor's Thesis in Informatics: Games Engineering, The Technical University of Munich.*

LAGUE, S., 2019. Coding adventure: Hydraulic erosion. https://www.youtube.com/watch?v=eaXk97ujbPQ. [Accessed 2020-12-12].

MCGUIRE, M., 2014. 1d, 2d 3d value noise. https://www.shadertoy.com/view/4dS3Wd. [Accessed 2020-12-12].