# Progress Report

Bruce Alcock

A Procedural, Minimal Input, Natural Terrain Plug-in for Blender

Supervisors: Kevin Glass and Shaun Bangay

Date: 19 April 2007

## 1 Previous Short Term Objectives

- Create a method to erode the terrain as detailed in [2].
- Create a way of specifying areas to have flatter ground.
- Create a method to roll river particles down the terrain to create river systems as detailed in [1].

## 2 Progress

### 2.1 Comparison of Renderings

In an effort to visually understand better how the fBm generating function works, the variables were tweaked in a controlled manner to show the effect each has. The variables mentioned here are defined in Algorithm 1: v.co[0,1,2] are the x,y,z coordinates of the current vertex in the mesh. The findings as qualitatively evaluated are as follows:

- XYZ division: higher values effectively zoom in on the fractal, so it tends more and more toward flat ground with higher numbers. See Figure 1.
- Fractal increment (H): seems to have to opposite effect to octaves. Lower values produce more detailed terrain. See Figure 2.

**Algorithm 1** fBm generation

```
val = v.co[0]/xyzdiv, v.co[1], v.co[2]/xyzdiv
newN = Blender.Noise.fBm(val, H, lacunarity, octaves, 2)*10.0
v.co[1] = newN
```

- Lacunarity: is a measure of how the fractal fills space, and higher values seem to produce smoother terrain. See Figure 3.
- Octaves: higher values for octaves predictably makes the terrain more detailed, since it's sampling other frequencies and interpolating them. See Figure 4.

## 2.2 Flattening Ground

In order to have scenes where action can happen, generally there will need to be flat(ter) areas in which to place cities, fields, or just general action to take place. As such a method to flatten areas needs to be produced. Two methods were tried, both interpolating the result of a function with the fBm values. A cosine function trace in the range [0, 1] generated on distance from a center point as shown in Figure 6 multiplied with the fBm to produce output as shown in Figure 5. Similarly an exponential function trace in the range [0, 1] generated in the same way as shown in Figure 6 multiplied with the fBm to produce output as shown in Figure 5.

Although the exponential function does a better job of flattening the terrain, perhaps the cosine function is more suitable, because it flattens a smaller area and then fairly slowly merges back in with the fBm and looks more natural. The exponential function could however be tweaked to a lower base for the exponent (the one shown in Figures 5, 6 has a base of 8) to achieve a similar effect.

Because the flattening is actually too harsh, flattening the terrain to an unrealistic degree, other methods will need to be considered, perhaps adding on small bits of fBm to maintain a slightly uneven appearance. Also, areas should be able to be raised as well as flattenend.

## 2.3 Related Work

From the Blender website there is a link to Blender World Forge (BWF), which "is a Python script designed to build worlds. Its main purpose is the generation of fractal terrains, whether flat or spherical, on which the user can later on add water, clouds etc." After downloading it, looking at screenshots and reading the documentation: it generates fBm terrains and can add in craters, but any work making it look more realistic: adding lakes, flatter areas and so forth is all done in a post-processing step: the user has to go and edit the mesh created. It seems to be more concerned with generating fractal planets, even though the author makes mention of flat terrains. BWF web link: `http://www.selleri.org/Blender/scripts/BWF-0.1.0.zip`

# 3 Problems

Needed to learn how to use both arrays (lists) and tuples properly, which prevented me from doing erosion [2] and river particles [1].

# 4 Objectives for Next Week

- Implement erosion [2].
- Create bigger terrain
- Implement multiple flattening locations.
- Consider and implement different flattening methods.

# References

[1] Fares Belhadj and Pierre Audibert. Modeling landscapes with ridges and rivers: bottom up approach. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 447–450, New York, NY, USA, 2005. ACM Press.

[2] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 41–50. ACM Press, 1989.

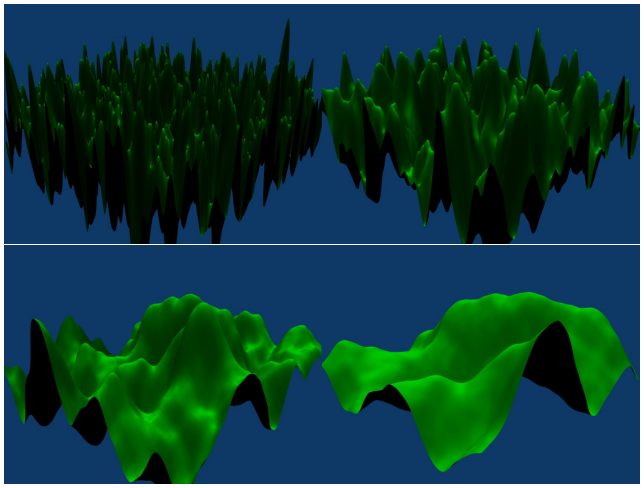

Figure 1: Changing xyz division values (1, 2, 5, 9)

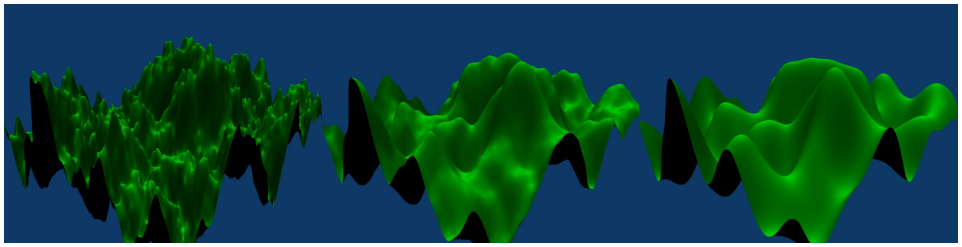

Figure 2: Changing fractal increment: H (1, 2, 4)

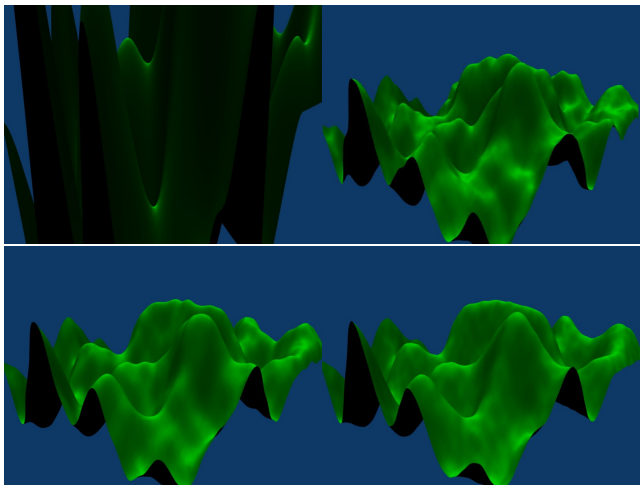

Figure 3: Changing lacunarity (1, 2, 4, 8)

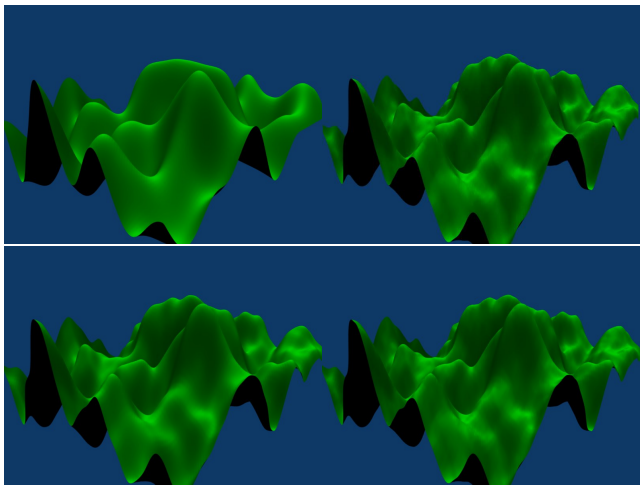

Figure 4: Changing octaves (1, 2, 3, 10)

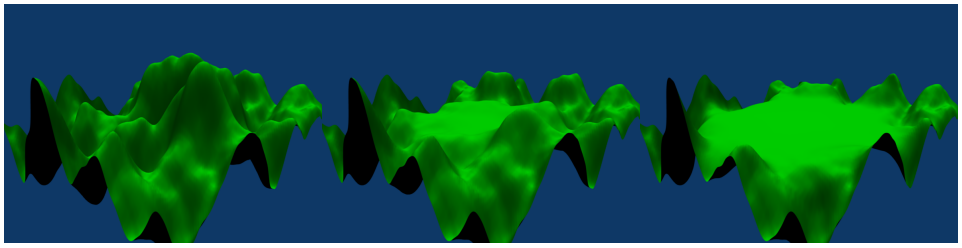

Figure 5: Flattening: none, cosine, exponential

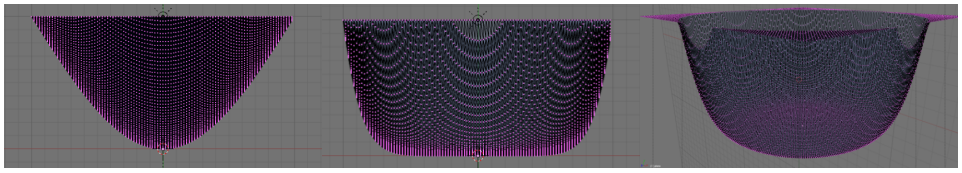

Figure 6: Flattening function: cosine, exponential, exponential (perspective)