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

Bruce Alcock*
Rhodes University

Abstract

This paper uses methods of terrain representation, creation and realism described in literature. In particular a combination of terrain synthesis using Fractional Brownian Motion, and erosion procedures is examined, as well as the procedural formation of rivers via squig curves. Other techniques are examined, such as Triangulated Irregular Networks and ridges generation are discarded in terms of this project for reasons detailed. We find that using a combination of Fractional Brownian Motion, procedural formation of rivers via squig curves to form initial terrain, and hydraulic erosion for post processing, we have full control over the style of terrain: from jagged, mountainous regions to flat regions; and the phase of river from tightly rock controlled to flood plain regions.

1 Introduction

1.1 Problem Statement

In an effort to increase the richness of virtual worlds, the goal of this project is to create realistic, natural terrain with minimal input, but yet to be as configurable as possible so as to allow for a multitude of scenes to be created.

1.2 Background

A Text-to-Scene converter is being developed as another project in this Computer Science department, which takes text from a book for example as an input and can then derive a three dimensional world from this, with the ultimate goal of being able to fully automate movie production from scripts of text. This project aims to address the needs of the Text-to-Scene converter for terrain, by being able to configure areas for city placement and action areas while being able to fill in the rest.

1.3 Overview

Procedural methods have been chosen for use with this project to fit with the minimal input paradigm, which means that no pre-existing data needs to exist for the terrain generation. The configurability of the procedural methods lies in the control through parameters, which will be expanded on in Section 4.

*Supervised by Kevin Glass, Shaun Bangay and Hannah Slay

2 Related Work

2.1 Representation

2.1.1 Heightmaps

A heightmap is a two dimensional matrix of coordinates in which the height of the terrain is represented as a number at each coordinate. Each is associated with the height above the base level (hence the name heightmaps). This method of representation has the shortcoming that terrain features such as overhangs and caves cannot be represented [Benes and Forsbach 2001a]. However it is often used because of the regular interval spacing, which makes erosion calculations quicker for example, because the distances are constant; as well as the easy storage into two dimensional arrays [Benes and Forsbach 2001a; Benes and Forsbach 2002; Musgrave et al. 1989].

2.1.2 Digital Elevation Models (DEMs)

DEMs are grid-based measurements of real world data, and different areas of the world are available in different gaps between the grid locations [Zhou et al. 2007]. Since this method relies on predefined data it is discarded for the purposes of this project.

2.1.3 Voxels

Voxels divide an object (in this case terrain) into three dimensional cubes of a set size. Although this gives more control over the terrain and provides a mechanism for creating caves and other features, it requires substantially more memory than a heightfield [Benes and Forsbach 2001a]. It is computationally expensive since it requires surfaces to be created from each voxel to its neighbours [Parker and Udeshi 2003]. It is discarded for this project because it would require many algorithms such as erosion to be completely re-thought and has not been used much for terrain modelling.

2.1.4 Triangulated Irregular Nets (TINs)

Another representation is a TIN, which represents a surface as a set of irregular size, non-overlapping, yet contiguous triangles [Fowler and Little 1979]. The reason this is not considered is because it is a method to speed up rendering in real-time, as demonstrated in [Duchaineau et al. 1997]. TINs therefore offer no benefits to realism.

2.2 Synthesis

2.2.1 Subdivision

The original purpose of subdivision was to divide a surface into small enough regions so as to be displayable as a pixel on screen [Catmull 1974]. It works by finding the midpoint of a curve segment and creating new curves between the originals, therefore enhancing its definition by creating seven vertices where previously

only three existed, or more importantly: four surfaces where previously one existed [Catmull 1974]. A recent method called $\sqrt{3}$ subdivision has been created where faces are divided into three, and new vertices adaptively adjusted to apply more subdivision to areas that have bigger curvature, thereby smoothing rough edges [Kobbelt 2000]. Via usage of weights, sharp edges can still be produced at specifiable locations on models [Kobbelt 2000]. $\sqrt{3}$ subdivision is now a popular choice to use for subdivision.

A use for this in creating terrain, is that at each stage of subdivision the new vertices are adjusted on the vertical axis by random amounts, forming initial terrain [Prusinkiewicz and Hammel 1993]. Two older methods and a proposed one for creating terrain via recursive subdivision are discussed in [Miller 1986]. These methods of creating terrain seem to have lost popularity to noise sampling (see Section 2.2.2) because of their reproducibility and the infinite detail of fractals.

Subdivision is usually used for smoothing models and increasing resolution when approximating surfaces, where a regular subdivision hierarchy is used to update a mesh dependent on view [Lindstrom and Pascucci 2001]. Another example is to adaptively subdivide recursively: subdivide to increase resolution where the model dictates it is necessary [Kajiya 1983].

2.2.2 Noise Sampling

Noise sampling methods use a base noise function (like Perlin Noise) to generate a terrain by sampling it in some way. Fractional Brownian Motion, is presented to create an initial heightmap for the terrain, generated by sampling Perlin Noise at multiple frequencies and applying translation and scaling onto the values [Musgrave et al. 1989]. Because the basis (Perlin) function is seeded, the entire fBm generation process is entirely reproducible.

Work has been done since to improve the resolution of fBm and stochastic models in general. Due to sampling problems the model produced by sampling fBm is very crude, and by path generation a better approximation can be attained [Fournier et al. 1982]. Another method is using random displacements: generating random numbers and using this as the heightmap [Fournier et al. 1982].

2.2.3 Feature generation

A method of terrain generation suggested, entails generating ridges and rivers and then interpolating between them using an extension to the midpoint-displacement method to fill in the rest of the terrain [Belhadj and Audibert 2005]. Another method is to create rivers via the use of squig curves and interpolating it into the base terrain [Prusinkiewicz and Hammel 1993]. A new method using a sketchpad has also been developed, which involves drawing the main features on a sketchpad and accesses stored digital elevation maps to match features to build a terrain model from the matchings [Zhou et al. 2007].

2.3 Realism

2.3.1 Erosion: Hydraulic and thermal

Hydraulic erosion involves dripping water onto the heightfield and distributing the water and sediment accumulated to neighbouring vertices. The erosive power at the current position is calculated based on the volume of water at that point as well as the sediment already in the water [Musgrave et al. 1989].

Thermal weathering encompasses any natural process that knocks material off ridges and deposits them at the feet of the mountain. If the slope of a vertex exceeds a talus angle, material is simply distributed to the neighbours, which softens ridges and valleys alike [Musgrave et al. 1989].

2.3.2 Multilayer heightmaps

Multi-layer heightmaps are a tool for realism, because they afford the mechanism to have different layers having different attributes. When combined with a method like erosion they can help to strive toward realism due to different layers having different hardnesses and as such eroding at different rates, as happens in nature [Benes and Forsbach 2001a]. The only usage of multi-layer heightmaps so far has been by the same authors that created it, to simulate erosion on the surface of Mars is parallel [Benes and Forsbach 2001b].

3 Procedural Methods for Terrain Generation

3.1 Representation and Synthesis

3.1.1 Heightmaps

This method of representation is used for this project due to the simplicity. However it has been extended to include the three erosion constants (see section 3.2.3) for each position for enhanced configurability: it means each location can have its own set of erosion constants and therefore regions can be harder than others, more willing to take on sediment, and have the ability to hold more water at that location.

3.1.2 Fractional Brownian Motion

Perlin Noise is used as a function to create pseudo-random values. Perlin noise works by calculating values for $[x,y,z]$ points by assigning pseudo-random, uncorrelated gradients $[a,b,c]$ to them and a value d which forms a linear equation for $[x,y,z]$ and can be retrieved using pre-calculated hash tables [Perlin 1985].

Fractional Brownian Motion (fBm) uses a base noise function and samples it at multiple frequencies. The base frequency is calculated using the equation:

$$a_0 = (N(p_0) + c_t)\, c_s + c_0$$

where a is the heightmap point, N is the noise generating function, p is the initial grid point, c_t is a constant translation transformation, c_s is a constant scaling transformation. Successive frequencies are calculated by:

$$a_i = a_{i-1} + a_{i-1}\,(N(p_i) + c_t)\, c_s w^i$$

where w^i is a frequency increment constant. Since the underlying noise function can be two dimensional, the generated points are continuous and produce a fractal terrain, which is useful for its endless level-of-detail and reproducibility, but even with the same seed for the noise function, the end terrain can appear very different due to the scaling and translation constants [Musgrave et al. 1989].

3.2 Realism

3.2.1 Squig Curves

The chosen method of river creation is via squig curves, which are an L-system of sorts and a subdivision scheme (see Figure 3), involving each triangle having an entry, exit and neutral edge [Prusinkiewicz and Hammel 1993]. The technique works by randomly assigning edges (entry, exit and neutral) to an initial triangle. When subdivided the original edges are split into two (entry edges becoming entry and neutral, exit becoming exit and neutral, and neutral becoming two neutrals), and the three new edges between them being assigned accordingly. This process is repeated recursively. By tracing from the midpoint of entry to exit edges, it gives a fairly reasonable representation of a winding river (see Figure 4), and this is then plotted onto the main terrain heightmap and vertices simply lowered to a base level where the rivers occur [Prusinkiewicz and Hammel 1993].

3.2.2 Flattening

In order to integrate with the Text-To-Scene and city creation project, areas need to be flattened for city placement and action areas in the scene (presumably activities in the script will not be performed on hugely jagged terrain). If a point is within a specified radius, a cosine function is used to create a flattened area:

$$value = 1 - cos\left(\frac{d}{D} \times \frac{\pi}{2}\right)$$

where d is the distance from the center and D is the desired radius of the flattened area. The values produced are between zero and one, which can then be used to smoothly interpolate flat areas with the rest of the terrain.

3.2.3 Erosion

Hydraulic erosion works by assigning each vertex v at time t an altitude a_t^v , a volume of water $w_t^v$and an amount of sediment s_t^v . Iterating through time each vertex v passes excess water and sediment to each neighbouring vertex u where the amount of water passed is defined as:

$$\Delta w = min\left(w_t^v, (w_t^v + a_t^v) - (w_t^u + a_t^u)\right)$$

Based on the amount of water passed, it is determined which direction the water is flowing (see Algorithm 1). If water is being passed to neighbours, the water at the current and neighbour vertices are calculated as well as the sediment capacity c_s to calculate sediment transfers with (see Algorithm 1), then sediment movement is calculated (see Algorithm 2). K_c is the sediment capacity which specifies the max sediment that may be suspended in a unit of water, K_d is the deposition constant which specifies the rate at which sediment settles out of a unit of water and is added to a vertex, and K_s is the soil softness constant, which specifies the softness of the soil and is used to control the rate at which soil is converted to sediment. What this does is to erode the terrain to an approximation of what happens in reality when rain falls on ground and redistributes soil [Musgrave et al. 1989].

Algorithm 1 Water Change

if $\Delta w \leq 0$
  $a_{t+1}^v = a_t^v + K_d s_t^v$
  $s_{t+1}^v = (1 - K_d)\, s_t^v$
else
  $w_{t+1}^v = w_t^v - \Delta w$
  $w_{t+1}^u = w_t^u - \Delta w$
  $c_s = K_c \Delta w$

Algorithm 2 Sediment Movement

if $s_t^v \geq c_s$
  $s_{t+1}^u = s_t^u + c_s$
  $a_{t+1}^v = a_t^v + K_d\,(s_t^v - c_s)$
  $s_{t+1}^v = (1 - K_d)\,(s_t^v - c_s)$
else
  $s_{t+1}^u = s_t^u + s_t^v + K_s\,(c_s - s_t^v)$
  $a_{t+1}^v = a_t^v - K_s\,(c_s - s_t^v)$
  $s_{t+1}^v = 0$

4 Implementation in Blender

The techniques are implemented as Python scripts for Blender.

4.1 Representation and Synthesis

4.1.1 Heightmaps

The heightmap is represented as a two dimensional array containing four values for each element: the height, and the three erosion constants (see Section 3.2.3). Having each location have its own set of erosion constants means that areas may be harder than others for instance, which affords the possibility to model very tough rocks in the middle of a hill and erode this for instance (see Figure 1).

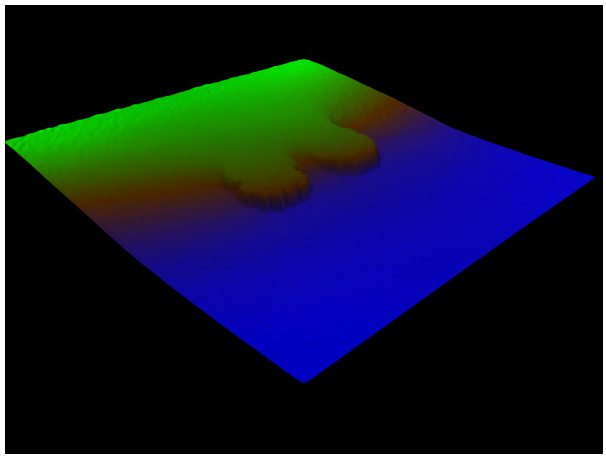


Figure 1: Natural phenomena replication

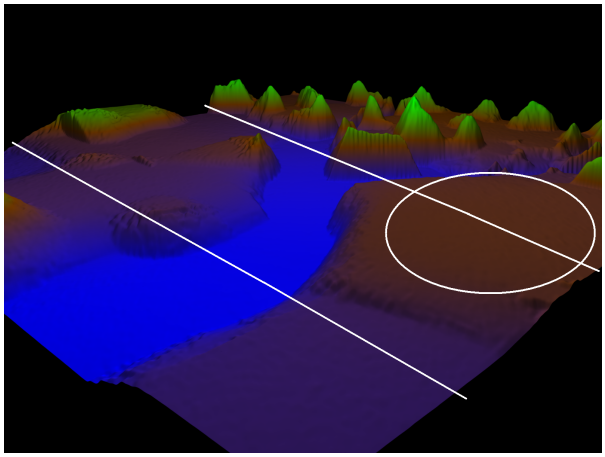


Figure 2: Terrain formation control

4.1.2 Fractional Brownian Motion

Blender provides a method to generate fBm, with parameters for the vertex position, the fractal increment, the lacunarity or gap between successive frequencies, the number of octaves to use, and the base noise function to use. We find that New Perlin noise is the most appropriate type of noise for the base function. Two additional values that can be tweaked are: the input vertex position, which can be scaled to create more rapidly or slowly changing terrain; and the scaling on the output value, which can make for higher, more jagged terrain with higher values. By altering parameters, we can control the formation of the terrain, which means sections of the terrain can be made to vary from others: having tightly rock controlled, jagged areas to flood plains, as can be seen in Figure 2: the far area being tightly rock controlled, the middle area a medium region and the close area a low flood plain.

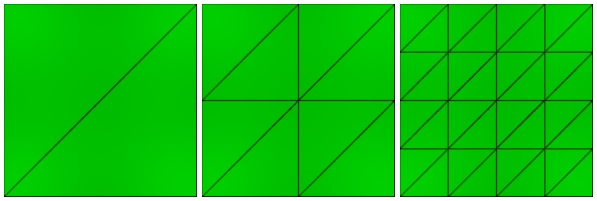
4.2 Realism

4.2.1 Squig Curves

As an observation on the method proposed, edges can be shared rather than creating two separate edges and assigning one to be an entrance and the other an exit. Also it is unnecessary to assign a label of entrance and exit, rather we use a system where an edge can be neutral, marked or unassigned (during subdivision only) [Prusinkiewicz and Hammel 1993].

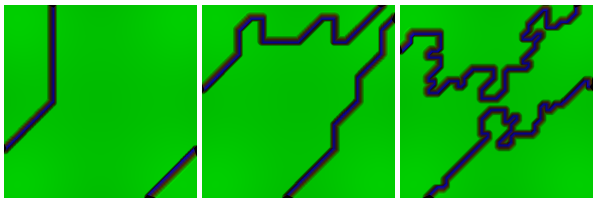
To thicken the river paths, as well as provide a gradual fade from river to normal base terrain, we use a radius of influence calculation, assigning heightmap values based on how far they are from a river path, and this radius can be altered which can assist in interpolating with the terrain to create jagged areas to flood plain areas as can be seen in Figure 2 where the far area has narrow rivers and the close area wider ones.

Another configurable option would be to select the recursive depth of subdivision for each triangle so as to provide meandering sections of river and swiftly changing ones. This is still to be implemented.



(a) n=0 (b) n=1 (c) n=2

Figure 3: Squig Curves: Grid after *n* levels of subdivision



(a) n=1 (b) n=3 (c) n=5

Figure 4: Squig Curves: Path after *n* levels of subdivision

4.2.2 Flattening

This is implemented as per the theory, using a cosine graph of configurable radius and can be seen in Figure 2 shown as the circled area. The method takes as input the center of the circle and the radius so as to be of use to the Text-To-Scene converter. If a more rectangular region is needed to be flattened this can be achieved by flattening a few smaller circles in a row.

4.2.3 Erosion

A first attempt to duplicate the results detailed in the paper failed, with wave-like erosion images being formed [Musgrave et al. 1989]. This problem was diagnosed to be unfair distribution of water and sediment: because the neighbours are checked in the same order every time, the first few being checked were getting preference. This problem is clearly visible in the two renders in Figure 5. The solution was to distribute the water proportionally looking at each neighbour as one of the eight [Benes and Forsbach 2002].

The authors recommend two heightmaps: one from a previous step and the currently worked on one which means that apart from the memory usage being twice as much, calculations are being repeated many times [Musgrave et al. 1989]. We find that using a precalculation step to calculate the proportion each neighbour will receive, we can eliminate the need for a second array, replacing it

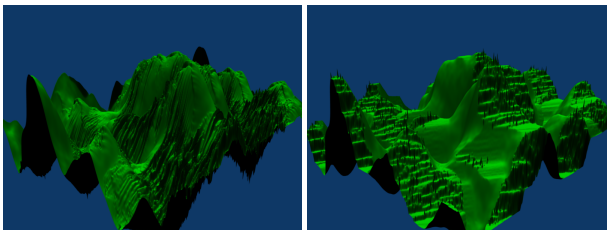


Figure 5: Wave erosion

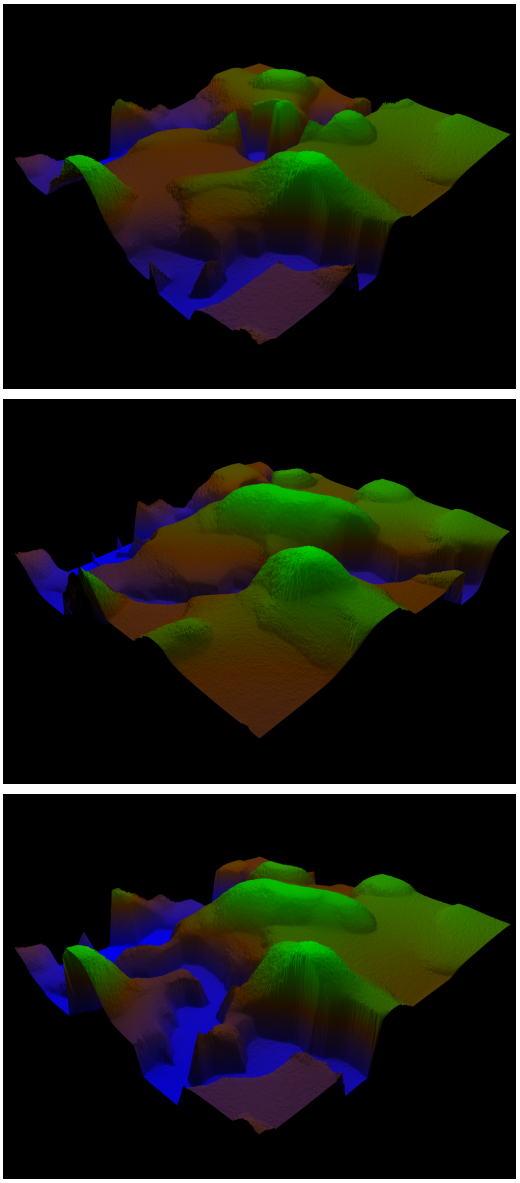


Figure 6: Results

with a variable length array for neighbours that will receive sediment and water and the proportions thereof. The erosion method also implements evaporation with an exponential method to prevent pools of stagnant water building up and creating jagged areas of terrain due to skewed erosion calculations [Benes and Forsbach 2002]. The method will continue to run after the amount of timesteps specified, until all the water has evaporated, as this gives the sediment contained in the water time to settle. The rain occurs every set number of timesteps and stops once the desired number of timesteps has passed.

Since the erosion algorithm is a global procedure the implemented method takes as input the desired number of timesteps to erode by, and if certain areas need to be eroded more than others to mimic softness or hardness of areas this can be achieved because each location on the heightmap has its own set of erosion constants.

5 Results

The three renders in Figure 6 show common results produced by this project, what can be emphasized here is that the only difference between the images is the squig river paths, other than that the images have the same base terrain and it is the erosion procedure that makes each image unique. The render in Figure 2 shows the configurability of the methods, with differing radius of influence for the river based on the section, as well as modified fBm output scaling to create the jagged area to floodplain transition, as well as the flattened region (refer to Section 4.1.2).

6 Conclusions

The project is capable of producing terrain with minimal input, and is very configurable yet very simple as has been mentioned in each relevant section. A multitude of scenes can be created by altering parameters, in particular offsetting the fBm would create completely different regions, but also the squig curve river generation which is completely random within its ruleset. With erosion procedures as well as configurable control of all parameters the project is capable of producing realistic, natural terrain and has so achieved the goals of the project.

References

BELHADJ, F., AND AUDIBERT, P. 2005. 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*, ACM Press, New York, NY, USA, 447–450.

BENES, B., AND FORSBACH, R. 2001. Layered data representation for visual simulation of terrain erosion. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, IEEE Computer Society, Washington, DC, USA, 80.

BENES, B., AND FORSBACH, R. 2001. Parallel implementation of terrain erosion applied to the surface of mars. In *AFRIGRAPH '01: Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation*, ACM Press, New York, NY, USA, 53–57.

BENES, B., AND FORSBACH, R. 2002. Visual simulation of hydraulic erosion. In *Journal of WSCG 2002: Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2002*, WSCG, 79.

CATMULL, E. E. 1974. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Department of Computer Science, University of Utah.

DUCHAINEAU, M., WOLINSKY, M., SIGETI, D. E., MILLER, M. C., ALDRICH, C., AND MINEEV-WEINSTEIN, M. B. 1997. Roaming terrain: real-time optimally adapting meshes. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, USA, 81–88.

FOURNIER, A., FUSSELL, D., AND CARPENTER, L. 1982. Computer rendering of stochastic models. *Commun. ACM 25*, 6, 371–384.

FOWLER, R. J., AND LITTLE, J. J. 1979. Automatic extraction of irregular network digital terrain models. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics*

and interactive techniques, ACM Press, New York, NY, USA, 199–207.

KAJIYA, J. T. 1983. New techniques for ray tracing procedurally defined objects. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 91–102.

KOBBELT, L. 2000. *sqrt*3-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 103–112.

LINDSTROM, P., AND PASCUCCI, V. 2001. Visualization of large terrains made easy. In *VIS '01: Proceedings of the conference on Visualization '01*, IEEE Computer Society, Washington, DC, USA, 363–371.

MILLER, G. S. P. 1986. The definition and rendering of terrain maps. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 39–48.

MUSGRAVE, F. K., KOLB, C. E., AND MACE, R. S. 1989. The synthesis and rendering of eroded fractal terrains. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM Press, 41–50.

PARKER, E., AND UDESHI, T. 2003. Exploiting self-similarity in geometry for voxel based solid modeling. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, ACM Press, New York, NY, USA, 157–166.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 287–296.

PRUSINKIEWICZ, P., AND HAMMEL, M. 1993. A fractal model of mountains with rivers. In *Proceeding of Graphics Interface '93*, 174–180.

ZHOU, H., SUN, J., TURK, G., AND REHG, J. M. 2007. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (July/August), to appear.