Literature Review

Bruce Alcock

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

Supervisors: Kevin Glass and Shaun Bangay

Abstract

This review examines 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. Other techniques as examined, such as Triangulated Irregular Networks and ridges generation and discarded in terms of this project for reasons detailed. In conclusion it is decided that a combination of Fractional Brownian Motion, procedural formation of rivers via squig curves and a physically based model of river formation will be used as initial terrain, and then eroded to achieve the goals of this project.

1 Introduction

To create natural terrain procedurally there are three areas that need to be examined. The first is the data representation: how to represent terrain on a computer in an efficient form. Secondly the initial terrain synthesis method: whether to use some form of pseudo-random noise or a model based on some physical method. And lastly the method of increasing realism, since the initial terrain methods available all have flaws which make them unsatisfactory. This literature review summarizes the methods currently proposed in the field of terrain generation on these topics.

2 Overview

We define three categories of techniques useful for terrain generation: data representation (see Section 3), that is, how to represent terrain on a computer in a manageable form; the initial terrain synthesis method (see Section 4) that describes how initial terrain-like geometry is generated; realism (see Section 5), which improves upon the coarse initial terrain synthesis methods.

3 Representation

The data structures used to represent the terrain affects both the realism of the produced terrain, as well as the complexity of the terrain synthesis algorithms. Popular approaches include: heightmaps which are a two-dimensional representation; and voxels which are three-dimensional, as well as TINs which represent the model as a set of non uniform triangles.

3.1 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 may be associated with information such as the height above the base level (hence the name height fields), as well as auxiliary information such as the amount of water at that point and the sediment contained in the water [4, 13]. Other information stored includes squig curves to represent river paths and river paths created by simulations [1, 16].

Multi-layer heightmaps are an extension to the idea of heightmaps, based on real world geological core samples. They encompass more than one layer of material, each of which can have different hardness and erosion constants for instance [2].

This method of representation has the shortcoming that terrain features such as overhangs and caves cannot be represented [2]. 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 [2, 4, 13].

3.2 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, because the space requirement is now $n \times gridsize^3$ rather than $n \times gridsize^2$, where n represents the space requirement for each element [2]. It is also computationally expensive to visit each element, and to render since surfaces need to be created for each voxel to its neighbours[14]. For these reasons voxels have not been used to model terrains much and algorithms such as erosion would have to be entirely re-thought to work with this representation.

3.3 TINs

Another representation is a Triangular Irregular Network (TIN), which represents a surface as a set of irregular size, non-overlapping, yet contiguous triangles [8]. The reason this is not considered is because it is a method to speed up rendering in realtime, as demonstrated in [6]. TINs therefore offer no benefits to realism.

3.4 Summary

The method chosen is heightmaps since they provide a regular method of representing terrain, useful for erosion and similar algorithms, with a possible extension of multi-layer heightmaps to make an attempt at caves and overhangs. TINs are a method to increase rendering speeds and as such offer no benefit to this project, and voxels complicate calculations, increase time and complexity and would need intense rewrites of algorithms.

4 Synthesis

Generating the initial values for heightmaps may be done using various different schemes, including subdivision, noise sampling, and feature generation, including rivers and ridges.

4.1 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 [5]. 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 [5]. 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 [10]. Via usage of weights, sharp edges can still be produced at specifiable locations on models [10]. $\sqrt{3}$ 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 context sensitive initial terrain [16]. Context sensitivity refers to the fact that two adjacent triangles share a common edge, and when subdivision occurs the new vertex in the center of that edge for each triangle is really only one vertex and must be treated as common and only adjusted once. Two older methods and a proposed one for creating terrain via recursive subdivision are discussed in [12]. These methods of creating terrain seem to have lost popularity to noise sampling (see Section 4.2) 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 [11]. Another example is to adaptively subdivide recursively: subdivide to increase resolution where the model dictates it is necessary [9].

4.2 Noise Sampling

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 [15].

A sampled form of Perlin Noise: Fractional Brownian Motion (fBm), 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 [13]. Because the basis (Perlin) function is seeded, the entire fBm generation process is entirely reproducible. 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 [13].

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 [7].

Another method is using random displacements: generating random numbers and using this as the heightmap [7]. An extension to this being random successive additions: generating many random numbers for each grid location and successively summing them to form the heightmap [17].

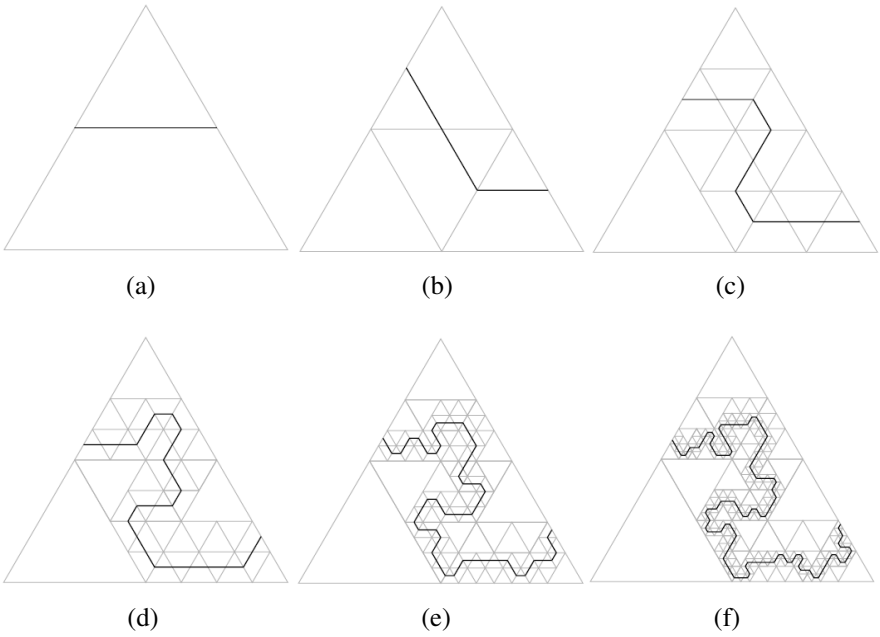

Figure 1: Squig Curves

4.3 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 [1]. Another method is to create rivers via the use of squig curves and interpolating it into the base terrain [16]. A new method using a sketchpad has also been developed [18].

4.3.1 Ridge Generation

Ridges are constructed by randomly scattering pairs of "ridge particles" over the map. Each pair of ridge particles initially has the same position and altitude, but opposite orientations. At each iteration and step the particle is impacted by Fractional Brownian Motion on the horizontal plane to make the ridges non linear, and altitude is decreased according to distance from the original location in a Gaussian distribution centered at the original location with a standard deviation of $\sigma = \frac{1}{4} \times width\,(heightmap)$. Each point modifies the heightmap's altitudes between the previous point and the current one, and then each successive point also has a Gaussian curve $\left(\sigma\prime = \frac{1}{4} \times \sigma\right)$ drawn perpendicular to it [1].

4.3.2 River Generation

One method of river creation is via squig curves, which are an L-system of sorts and a subdivision scheme, involving each triangle having an entry, exit and neutral edge [16]. Referring to Figure

1(a) as the original triangle, with the left side assigned to be an entry entry, right side an exit and bottom a neutral edge. When subdivided the original edges are divided randomly 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 (see Figure 1(b)) being assigned accordingly. This process is repeated recursively (see the last four images in Figure 1(c,d,e,f)). This gives a fairly reasonable representation of a winding river, and this is then plotted onto the main terrain heightmap and vertices simply lowered to a base level where the rivers occur [16].

Another method is via river particles, which are randomly dispatched over the map to create rivers. They are mobile spheres with a mass (which determines the resulting river's width and depth), and are subject to gravity. The river particles are placed at top top of ridge lines where a physically based model is used to describe the particle's motion, where negative y-axis acceleration is more weighted and frictional forces are added. The process stops when all particles are static or have exited the grid. The only case worth noting is when a river particle (r_i) intersects another's (r_j) path: then r_j is backtracked until the intersection points and its properties modified as follows [1]:

$$\begin{aligned} position\,(r_j) &= position\,(r_i) \\ \overrightarrow{velocity\,(r_j)} &= \overrightarrow{velocity\,(r_i)} + \overrightarrow{velocity\,(r_j)} \\ mass\,(r_j) &= mass\,(r_i) + mass\,(r_j) \end{aligned}$$

4.3.3 Other Areas

The final stage of the process is to fill in the areas that are not marked as river or ridge tops [1]. This cannot be done with subdivision schemes alone, because with little data they produce artifacts and discontinuities. Instead, the non-computed grid is filled using a subdivision scheme (either triangle-edge or diamond-square subdivision) using pre-fill data, calculated by processing the grid in scan-line order in increasingly big squares: each square looks at the four corners and if at least one corner has a value, the others are filled by interpolating with the other available corners. If at least one corner of the square has a value, the subdivision method is called for the square which fills in more coordinates [1].

4.3.4 Sketch Pad

A recent method involves drawing the main terrain features on a sketchpad. The system then accesses stored digital elevation map data and tries to match features to the drawn terrain and builds a terrain model from the matchings. This method will not be used due to it relying on a database and not being entirely procedural [18].

4.4 Summary

The chosen methods for this project are a combination of noise synthesis (Fractional Brownian Motion) to create the base terrain due to the results looking visually better than other methods,

Algorithm 1 Water change

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

and river generation and interpolation with the initial terrain using squig curves due to their ease of construction and effectiveness [13, 16].

5 Realism

Initial terrain synthesis alone is not sufficient for realistic terrain generation. The combination of midpoint displacement and squig curves generate unrealistic artifacts as a result of rivers cutting into high areas of the heightmaps [16]. Fractional Brownian Motion does not include mountains and rivers as they would exist naturally, because the landscape is statistically identical about the horizontal axis [13]. Ridge and river creation is insufficient because all areas between the ridge and river lines are only interpolated, producing unrealistic artifacts [1]. More needs to be done to pursue realistic terrain generation, and this is achieved using simulations of real world phenomena such as wind and water erosion [4, 13]. In addition, the use of multi-layer heightmaps may also be exploited.

5.1 Erosion

To give the raw heightfield a more natural looking appearance with proper mountains and riverbeds, two approximations of natural processes are suggested for eroding the terrain [13]:

5.1.1 Hydraulic Erosion

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. This 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:

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

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

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

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 [13].

5.1.2 Thermal Erosion

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 T, material is simply distributed to the neighbours, which softens ridges and valleys alike. So if the difference in slope exceeds the talus angle: $a_t^v - a_t^u > T$ then $a_{t+1}^u = a_t^u + c_t\,(a_t^v - a_t^u - T)$, where c_t is a constant percentage of the difference to move [13].

5.2 Multi-layer 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 [2]. 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 [3].

5.3 Summary

In an effort to create as realistic a terrain as possible, both hydraulic and thermal erosion will be tried and perhaps used in combination with multi-layer heightmaps [2, 4, 13].

6 Conclusion

In conclusion, the techniques that will be used in this project will be to generate Fractional Brownian Motion as a base terrain [13]. This will then use a method based on the Squig curves and ridges and rivers approaches to add in specific river regions [1, 16]. Finally erosion procedures will be applied onto the terrain, with reference to whether the vertex is part of a river or not [4, 13].

The literature is lacking a realistic method of creating a terrain, relying on post-creation methods to fix the terrain in an effort toward realism. Using the combination of methods mentioned, this should produce satisfactory renderings of natural looking terrain, with the only user input being a translation from the origin so as to allow for reproducible, yet movable and different terrain. However, there is still work to be done in the field of terrain before terrain is realistic enough to pass human standards on observation.

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] Bedrich Benes and Rafael Forsbach. Layered data representation for visual simulation of terrain erosion. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, page 80, Washington, DC, USA, 2001. IEEE Computer Society.

[3] Bedrich Benes and Rafael Forsbach. 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*, pages 53–57, New York, NY, USA, 2001. ACM Press.

[4] Bedrich Benes and Rafael Forsbach. 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*, page 79. WSCG, 2002.

[5] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Department of Computer Science, University of Utah, December 1974.

[6] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. Roaming terrain: real-time optimally adapting meshes. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 81–88, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[7] Alain Fournier, Don Fussell, and Loren Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, 1982.

[8] Robert J. Fowler and James J. Little. Automatic extraction of irregular network digital terrain models. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 199–207, New York, NY, USA, 1979. ACM Press.

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

[10] Leif Kobbelt. $sqrt3$-subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112. ACM Press/Addison-Wesley Publishing Co., 2000.

[11] Peter Lindstrom and Valerio Pascucci. Visualization of large terrains made easy. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 363–371, Washington, DC, USA, 2001. IEEE Computer Society.

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

[13] 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.

[14] Eric Parker and Tushar Udeshi. Exploiting self-similarity in geometry for voxel based solid modeling. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 157–166, New York, NY, USA, 2003. ACM Press.

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

[16] Przemyslaw Prusinkiewicz and Mark Hammel. A fractal model of mountains with rivers. In *Proceeding of Graphics Interface '93*, pages 174–180, Toronto, Ontario, May 1993.

[17] Dietmar Saupe. Algorithms for random fractals. pages 71–113, 1988.

[18] Howard Zhou, Jie Sun, Greg Turk, and James M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):to appear, July/August 2007.