THE UPLIFT MODEL TERRAIN GENERATOR

J R Rankin

La Trobe University
Bundoora, Australia

Abstract

Terrain generation finds many applications such as in CGI movies, animations and video games. This paper describes a new and simple-to-implement terrain generator called the Uplift Model. It is based on the theory of crustal deformations by uplifts in Geology. When a number of uplifts are made on the Earth's surface, the final net effect is an average of the influence of each uplift at each point on the terrain. The result of applying this model from Nature is a very realistic looking effect in the generated terrains. The model uses 6 parameters which allow for a great variety in landscape types produced. Comparisons are made with other existing terrain generation algorithms. Averaging causes erosion of the surface whereas fractal surfaces tend to be very jagged and more suited to alien worlds.

Keywords

Terrain, height map, height field, 3D rendering, fractals

Introduction

Terrain generation [1,3] is the creation of virtual land formations in Computer Graphics. The objective is to create realistic looking land formations without too much time-consuming processing especially for interactive CG applications and such as to be compatible with low resolution devices. Following the ideas introduced by Mandelbrot [6] many fractal algorithms were explored for creating the shapes of mountains. This has been the best in realism for terrain generation in the past but it is too computationally intensive for interactive games and in any case the terrains generated are too rough and alien to suit many modern 3D games and simulations.In [3] we reviewed 3 main fractal terrain generation algorithms called the Fault Formation algorithm [8], the Midpoint Displacement algorithm [2,9] and the Particle Deposition algorithm [4,10]. While the construction of fractals involves elegant recursive programming recent terrain generation algorithms [7] have reverted to the benefits of the iterative approaches. More recent work has involved using parameters to gain some control of the terrain appearance or using Evolutionary Algorithms to evolve a terrain into a more suitable form [11]. Thus there is still a need for a simpler non-recursive terrain generation algorithm that produces believable 3D outdoors hill or mountain scenes. This paper answers this need.

A 3D terrain is represented by a height field y_{ij} where i = 0 to N_x and j = 0 to N_z.The x-z plane is the horizontal ground level. The base of the terrain is a rectangle which lies in the x-z plane andhas width $N_x*\Delta_x$ and breadth $N_z*\Delta_z$. where Δ_x is the distance between grid lines along the x-axis and Δ_y is the distance between base grid lines along the z-axis. The Uplift Model is applicable to any sized base grid. Unlike many fractal algorithms such as the Diamond-Square algorithm [7] the base does not have to be square or a power of 2 with $N_x = N_z = 2^n+1$. The Uplift Model has more parameters for controlling the appearance of the terrain than most other terrain generators have. For example thepopular Diamond-Square fractal terrain generation algorithm (and also the similar Midpoint Displacement method) has only two parameters H (scaling exponent) and A(amplitude) [2] and Brownian surfaces [1,5] have only one parameter (amplitude).

Mountain Skyline Algorithm

A 2D version of the terrain generator algorithm was created and tested first. For a mountain skyline we only need the one-dimensional array y(x) giving the height y of the far mountain ridge at position x from left to right across the screen. The horizontal axis from left to right on the screen is divided into N_x equal intervals each of width Δ_x. so therefore we will compute y_i for i = 0 to N_x. First we must initialize the horizon as $y_i = 0.0$ for all i = 0 to N_x. Now consider a single peak of random height $y_p> 0$ at a random horizontal grid position i_p across the screen where $0 \leq i_p\leq N_x$. This is an uplift from below the surface of the Earth having an influence on all other points of the terrain. Since we will use the linear approximation, one may at first think that this influence should be approximated as the inverted V shape in Figure 1 below with (for initial considerations) unit slope to the left and right.

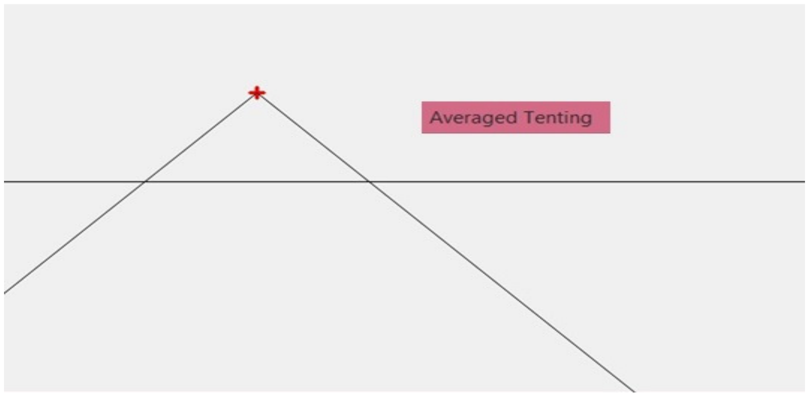

Figure 1. The inverted V shape representing uplift influence shown with the horizontal ground line.

If we now superimpose two random uplifts with this type of influence curve on the terrain we can compute the new terrain as the average of the heights of the two uplifts. This is shown as the red line in Figure 2. It is clear from Figure 2 that the increased height of the peak on the right hand side has no effect on the averaged curve. Furthermore, the averaged curve looks like nothing more than another uplift but with the peak cut off to form a plateau. Averaging influence curves of this type evidently is not going to give us new shapes to form novel and interesting landscapes.

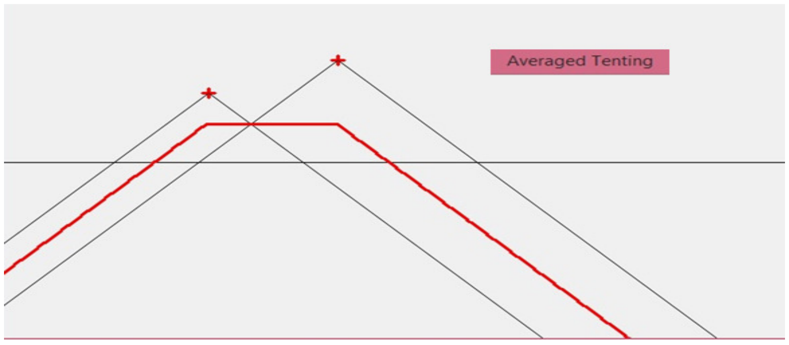

Figure 2. The result of averaging two uplifts based on the inverted V influence curve.

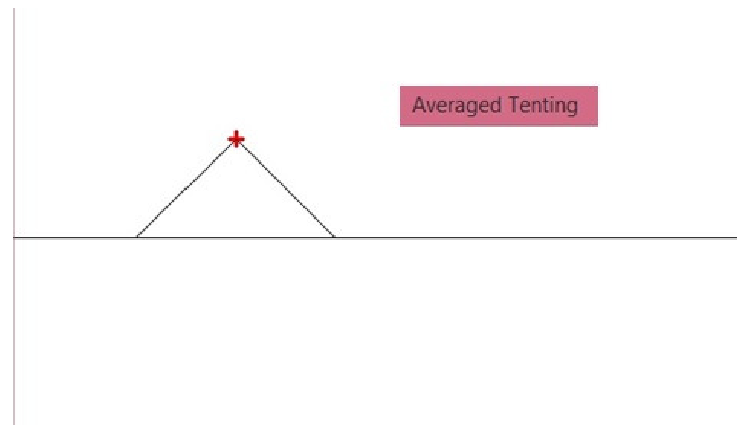

Figure 3. The influence curve of a tented uplift superimposed on the horizontal ground line.

However, a better approach is to use “tenting” wherein the influence is represented by the tent (or chinaman’s hat) shape shown in Figure 3. It is important to note that the tent shape is clamped flat at the sides at ground level so that this curve doesn’t go below zero.

To see why tenting is a better approach consider again the simple case of averaging two uplifts but now using tenting instead of inverted V shapes for the influence curves as in Figure 4 below:

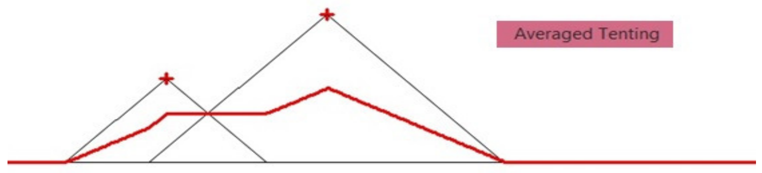

Figure 4. Averaging two uplifts represented as tents.

Comparing Figure 4 with Figure 2 it is clear that tenting provides the better approach to simulating terra formation with uplifts. In Figure 2 the average effect is a polyline with only 3 line segments whereas in Figure 4 the polyline is more complicated and has 7 line segments. In Figure 2 the taller uplift did not have an effect on raising the land mass closer to that uplift whereas in Figure 4 the higher uplift did exert greater influence nearer to its peak.

A feature of this terra formation algorithm is erosion: the averaged curve erodes the sharp corners of the tent curves.The averaged curve also lies well below the tent curves. It was also observed that the more peaks used in the Uplift Model the more erosion occurred so that the resulting land profile is lower (closer to the ground level). This erosion is a significant difference between the Uplift Model and fractal terra formation algorithms. For example a typical Brownian curve [5] looks as in Figure 5 below. The Brownian curves are reminiscent of craggy mountain ranges. By contrast the 2D Uplift Model produces smoother, worn-down mountain ranges as shown in Figure 6 below.

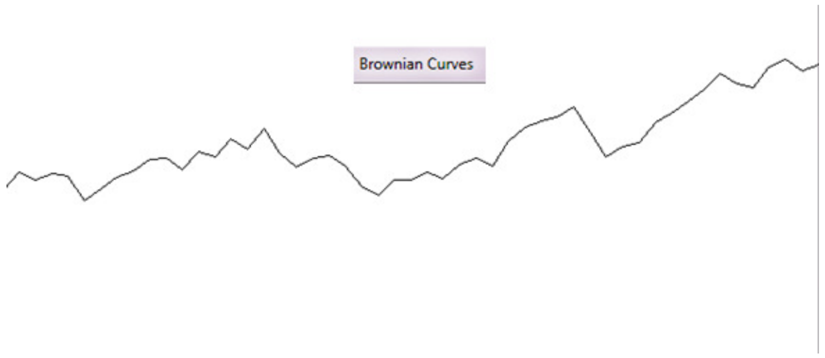

Figure 5. A Brownian curve used as a mountain ridge skyline.

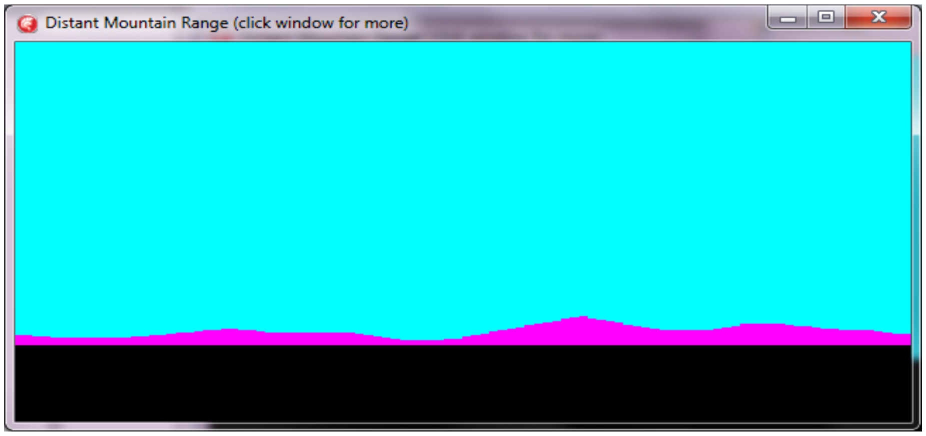

Figure 6. A distant mountain range using the 2D Uplift Model.

We further note that overlap of the tents is necessary for realistic outputs. If there are places with no overlap then isolated triangles appear on the horizon and these are obviously artificial. To have overlap we just need to have a sufficient number of uplifts generated. By having 10 or more random uplifts this algorithm of averaging their tent influences produces believable mountain range horizons such as shown in Figure 6above. With the horizon computed and displayed, other aspects of the view can be added to achieve the desired scene effect such as showing the sky changing tone from light blue to dark blue based on height y above the sea level (y = 0) in the picture, and such as adding stars or clouds and foreground scene features like trees, rivers and buildings.

If we now allow for slopes other than unity on the left and right sides of the uplifts we can generate skylines ranging from gentle hills to high rugged mountains. The 2D Uplift Model algorithm then uses these 5 parameters: the number of divisions N_x along the x-axis, the number of peaks (N_p), the range of values (i.e. minimum and maximum Y_{min} and Y_{max}) of the peak heights and the steepness R of the tent sides. While Figure 6 uses R = 1, Figure 7 below uses R = 20 resulting in mountains reminiscent of the Andes in Peru.

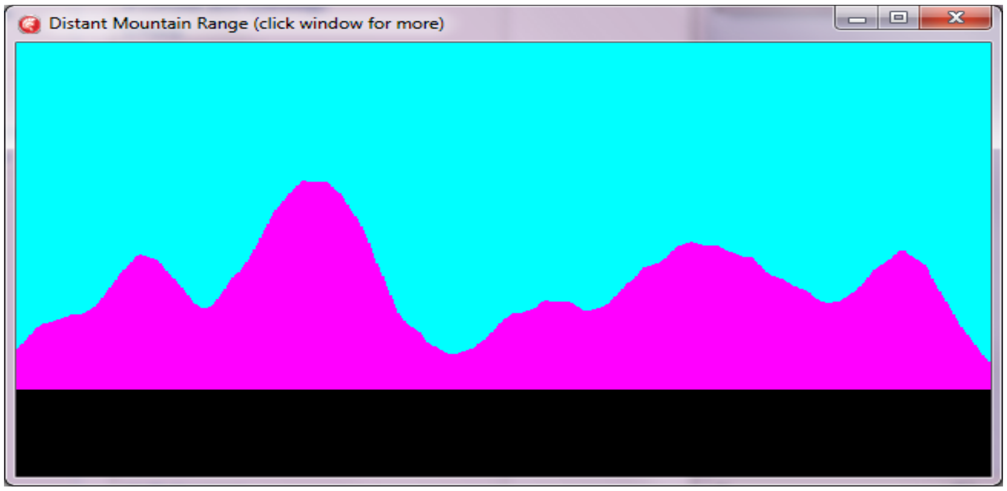

Figure 7. The 2D Uplift Model with ruggedness parameter R = 20.

The 3D Uplift Model

The same observations of the 2D Uplift Model of the preference for tenting, the requirement for overlap and erosion and the wide variety of scene types covered apply to the 3D Uplift Model. Start with choices for the 6 Uplift Model parameters N_x (the number of grid divisions along the x-axis),N_z(the number of grid divisions along the z-axis), N_p (the number of peaks), Y_{min} and Y_{max}, (the range in peak heights) and R (the ruggedness factor) and start with a perfectly flat height array $y_{ij} = 0.0$. In a loop for 1 to N_p, create each peak which requires a random grid position (i_p, j_p) and a random height y_p between Y_{min} and Y_{max}. The grid position (i_p,j_p) where $0 \leq i_p \leq N_x$, $0 \leq j_p \leq N_z$ is the centre for the peak. Given a peak with values i_p, j_p and $y_p$we next compute the influence y of this peak at all grid points (i,j). For simplicity this influence is supposed to drop off linearly from y_p at (i_p,j_p) according to the 2DEuclidean distance d of (i,j) from (i_p,j_p) in the x-z plane. The 3D uplift influence surface is therefore a cone sitting on the horizontal ground plane generalizing the tent curves of the 2D Uplift Model. Thus we compute:

$$d = \sqrt{[(i_p - i)\Delta_x]^2 + [(j_p - j)\Delta_z]^2}$$

from which we calculate the influence y as:

$$y = y_p - Rd$$

where R is the desired ruggedness factor for the terrain. If y < 0 then it is reset to y = 0. The cone at (i_p,j_p) therefore has height y_p and base radius y_p/R. The influence y of this peak at grid position (i,j) is then added on to y_{ij} and then we can move on to the next grid position. After the influences of this peak at all the grid positions have been computed and added into the height fields at those grid positions, then the algorithm does not need to save the i_p, j_p and y_p values and we can continue in the loop to do the same thing for the next peak. At the end of looping through all peaks we will divide the resulting y_{ij} values by N_p, (the number of peaks) to give the average influence at each grid site of all the peaks and this is the final height field value used for plotting the new terrain.

In rendering the terrain, first the water table [1] is rendered as the height map $y_{ij} = 0$ in blue. Then each grid cell is rendered as two shaded triangles in Open GL with their vertex coordinates (x_{ij}, y_{ij}, z_{ij}) computed by:

$$x_{ij} = i\Delta_x$$
$$z_{ij} = j\Delta_z$$

and y_{ij} given as the computed height field.

Results and Comparisons

The figure below,Figure 8, shows a typical highland terrain generated by the 3D Uplift Model. When we compare a typical highland scene generated by the Uplift Model as in Figure 8 with a typical Brownian surface scene shown in Figure 9, it is immediately apparent that the Brownian surface is far more jagged and alien than the Uplift Model surface. As the Brownian surface (like the Brownian curve) is based on Brownian Motion the correlation between heights is highest for neighbouring grid points and decreases linearly with increasing grid distance between the points. This increasing non-correlation results in the extreme ruggedness of the fractal mountains on all scales. It also results in the well-known lack of control that designers have on the shape of the output fractal surfaces [11]. These features of fractal terrains have meant that few fractal terrain generators have been used in 3D video games where realistic ground motion is required as pointed out in [11]. This ruggedness down to the smallest visible scales has made it difficult to accurately render animations and their shadows moving over the fractal surfaces. In contrast the Uplift Model has a strong erosion effect sharply reducing the jagged appearance of the landscape. Additionally we must consider that on average 50% of neighbouring height changes in Brownian surfaces must be reversals and this means that small jagged peaks will arise across the whole landscape in the fractal approach. As we have seen, this eroding effect of averaging in the Uplift Model prevents such small scale jagged formations from occurring.

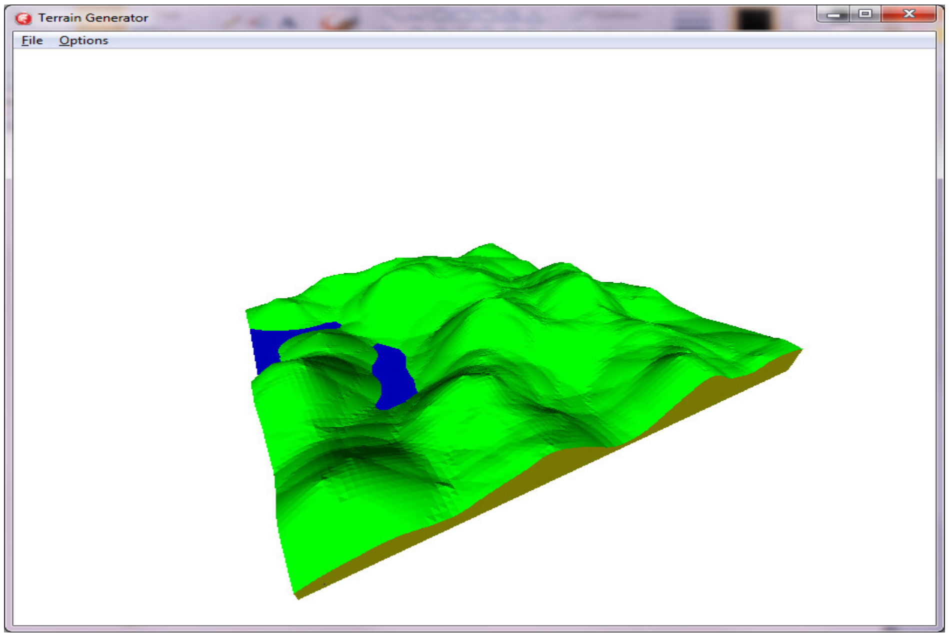

Figure 8. A typical highland landscape produced by the Uplift Model.

The view in the program for which Figure 8 is a snapshot can be turned left or right to reveal a very pleasing mountainous scene suitable for use in any outside 3D game. In comparison fractal mountain surfaces look very much like crinkled paper with jagged spurs on smaller and smaller scales. Fractal mountains don't give a feeling of size because fractals by definition are self-similar on all scales ad infinitum.

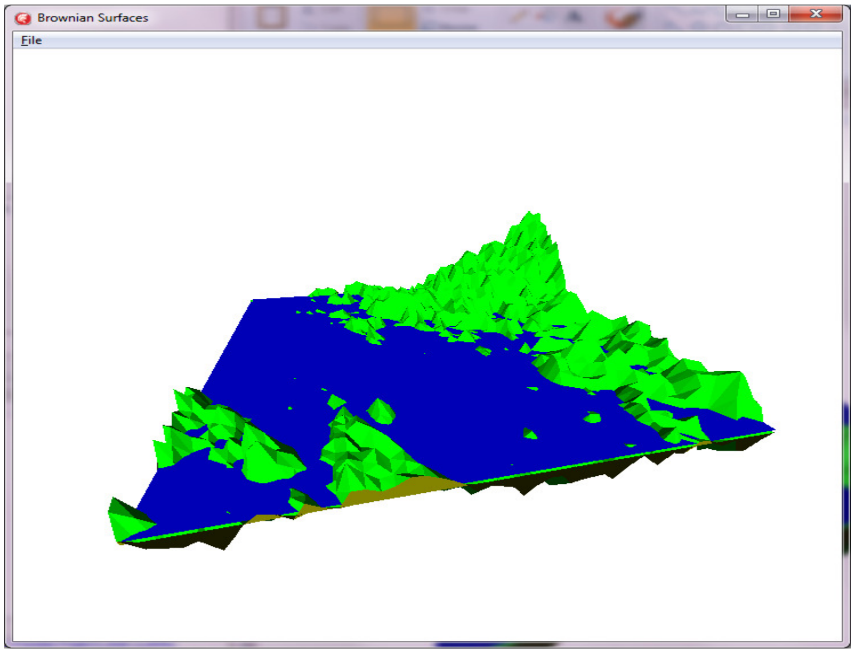

Figure 9. A typical Brownian surface to the same resolution $N_x x N_z$ as in Figure 8.

CONCLUSIONS

The Uplift Model is very easy to implement and generates realistic terrains ranging from, small islands, to marshlands, to lowlands with rolling hills to highlands with tall peaks, to mountainous zones and rugged mountain zones with almost vertical slopes as seen in the Peruvian Andes mountains. It does not suffer from the excessive jaggedness found in fractal landscapes that are perhaps more appropriate to alien worlds such as planets without atmospheres or for generating useful texture maps and height maps for representing hazy cloud cover. The Uplift Model should therefore find more applications in 3D video games and simulations of natural terrestrial weathered landscapes.

References

[1] Bessant, C "Computers and Chaos", Sigma Press, Wilmslow, UK, 1992, pp 131-144.
[2] Bird A.K., Dickerson B.T and George C.J."Techniques For Fractal Terrain Generation", Dickerson_Terrain.pdf available at http://web.williams.edu/Mathematics/sjmiller/public_html/hudson/Dickerson_Terrain.pdf
[3] Boyapati M and Rankin J.R., "Fractal Terrain Generation for SIMD Architectures"Rankin, J.R.|Boyapati, M International Journal of Computer Applications in Technology, 2009, Vol 34, No 4, pp 298-302
[4] Habrador, "How to Generate A Random Terrain" August 2014 at http://blog.habrador.com/2013/02/how-to-generate-random-terrain.html
[5] Lauwerier, H "Fractals, Images of Chaos", Penguin Books, 1991, pp114-116.
[6] Mandelbrot, BB "The Fractal Geometry of Nature", WH Freeman and Company, New York, USA, 1983, pp 232-243 & 256-264.

[7] Martz, P, "Generating Random Fractal Terrain", August 2014 at http://www.gameprogrammer.com/fractal.html

[8] Shankel J. "Fractal Terrain Generation – Fault Formation", Game Programming Gems 8, vol 1, pp 499-502, 2000.

[9] Shankel J. "Fractal Terrain Generation – Midpoint Displacement", Game Programming Gems 10, vol 1, pp 503-507, 2000.

[10] S hankel J. "Fractal Terrain Generation – Particle Deposition", Game Programming Gems 12, vol 1, pp 508-511, 2000.

[11] Affe W.L., Zambetta F. and Xiaodong L. "A Survey of Procedural Terrain Generation using Evolutionary Algorithms", IEEE World Congress on Computational Intelligence, WCC! June 2012, Brisbane, Australia