Modeling Landscapes with Ridges and Rivers

Belhadj Farès
A.I. Laboratory
Paris 8 University
2 rue de la Liberté
93526 Saint-Denis Cedex, France
amsi@ai.univ-paris8.fr

Audibert Pierre
A.I. Laboratory
Paris 8 University
2 rue de la Liberté
93526 Saint-Denis Cedex, France
audibert@ai.univ-paris8.fr

ABSTRACT
Generating realistic models of landscapes with drainage network is a major field in computer graphics. In this paper, we present a fractal based method which generates natural terrains using ridges and rivers information. As opposed to methods that compute water erosion for a given terrain model, our terrain mesh is generated constrained by a pre-defined set of ridge lines and rivers network. A new extension of the midpoint displacement model is used in this context. The resulting landscapes meshes lead to realistic rendering and our method seems to be very promising.

Categories and Subject Descriptors

General Terms
Algorithms

Keywords
terrain models, terrain erosion, fractals, midpoint displacement.

1. INTRODUCTION
Modeling realistic approximations of natural landscapes is a long standing problem in computer graphics. Well known interpolation methods [3, 5] are enough powerful for generating plausible models of non-eroded mountains. In 1988, Kelley et al. [4] overcame this limitation by generating a water drainage network to modify the initial terrain surface. Alternative methods based on the simulation of hydraulic and thermal erosion were proposed by Musgrave et al. [6, 2] and extended — using layered structures — by Nagshima [7]. In general, these last methods process an initial terrain in order to produce an eroded terrain model. However, despite simplifications, these methods still remain too strict, complex to manipulate and time consuming.

In A Fractal Model of Mountains with Rivers Prusinkiewicz and Hammel [8] described a novel approach that models — in a single integrated process — a landscape with a river using context-sensitive rewriting mechanisms. This approach still remains original and promising but suffers from either artifacts or the generation of only one river.

In this paper we introduce a bi-process method that models realistic landscapes with ridge lines and drainage network. Our algorithm generates a terrain model around a precomputed set of ridge lines and rivers network. First, a simple and rapid method that generates the ridges and rivers network is used. Then, we describe an extension of the basic midpoint displacement method that allows generating fractal terrain model around a pre-filled ridges and rivers network. This extension provides good interpolations without either artifacts or discontinuities.

2. MODELING RIDGES AND RIVERS
The first part of our modeling process focuses on the problem of generating an initial Digital Elevation Map — D.E.M. — that represents the ridges and rivers network. We describe in this section one simple way of obtaining this set of ridge lines and the associated drainage network.

2.1 Ridge Lines
Starting with an empty Digital Elevation Map, couples of Ridge Particles are randomly dispatched on the map. These Ridge Particles are mobile points with map coordinates, an altitude and an orientation. Within one couple, particles have the same initial position $P_i$ with relatively high altitude $y_0$ and opposite initial orientation angles. For a given step $\delta$ and at each iteration $i$, particles are subject to side impacts in way to describe a fractional Brownian motion ($fBm$) [3] on the $xz$ plane. The particle’s altitude decreases according to the covered distance $i \times \delta$ on a Gaussian distribution $G_i$ centered at the initial position and with a standard deviation $\sigma = \frac{1}{4} \times width(D.E.M.)$. Thus during the particle displacement from $P_i$ to $P_{i+1}$, we modify the map altitudes according to the successive particle’s altitudes on the $[P_i, P_{i+1}]$ segment. For each point $p$ on this segment, we draw a Gaussian curve $G_{p}(p, \sigma' = \frac{1}{4} \times \sigma)$ perpendicular to $[P_i, P_{i+1}]$ and with an amplitude equal to $altitude(p)$ (see Fig.1).
The Particle motion is stopped when its altitude became null or when it collides with another particle’s route.

Finally, by modifying the initial number of Ridge Particles, $G_1$, $G_2$, and the $fBm$ parameters we can obtain different kinds of ridged terrains.

![Figure 1: The route of a Ridge Particle couple $(r_i, r_{i+1})$ modifies the terrain mesh according to two Gaussian functions. At the initialization, $r_i$ starts at $P_0$ and moves toward $P_1$, $r_{i+1}$ takes the opposite direction from $P_0$ and moves toward $P'_1$.](image1)

2.2 Rivers Network

After the deformation of the Digital Elevation Map with the ridge lines, a similar method is used to draw the rivers network. Now we consider River Particles, they are mobile balls with a mass $m$ subject to gravity acceleration — $m$ serves in weighting the river’s width and its deepness. River Particles are randomly dispatched at the top of the ridge lines. A simple physically based method is used to model this particles motion (in the manner of “obtaining velocity field of water flow” in [1]); here the negative $y$ axis acceleration is more weighted and a friction power is added — thus intersections between the river’s route and ridge lines are suppressed. When a particle $r_i$ intersects another particle’s route $path(r_j)$, $r_i$ is stopped and $path(r_j)$ is backtracked until the intersection point. $r_j$ properties are modified such as:

\[
\begin{align*}
\text{position}(r_j) &= \text{position}(r_i) \\
\text{velocity}(r_j) &= \text{velocity}(r_j) + \text{velocity}(r_i) \\
\text{mass}(r_j) &= \text{mass}(r_j) + \text{mass}(r_i)
\end{align*}
\]

The process is stopped when all the particles either become static or exit from the terrain’s limits. In the next process, we only consider particles that exit from the terrain’s limits (see Fig. 2).

![Figure 2: A basic ridges and rivers network. Ridge lines are blended in order to compute the corresponding rivers network.](image2)

3. MIDPOINT-DISPLACEMENT

At this point, a plausible ridges and rivers network is generated and stored on the map; the ridges on Fig. 2 were blended using a Gaussian distribution to allow us to compute the rivers network. For the second part of the terrain generation process, we just keep the skeleton of this network and reinitialize all other points of the map (see Fig. 3).

The skeleton of the ridges and rivers network is stored in the Digital Elevation Map. We define four possible states for each map coordinate:

- **NULL** : the coordinate is not yet computed;
- **RIDGE** : the coordinate is on a Ridge Particle’s path, its value is equal to the ridge altitude at that position;
- **RIVER** : the coordinate is on a River Particle’s path, its value is equal to the ridge altitude at that position;
- **DONE** : the coordinate is computed.

Thus, a Digital Elevation Map containing NULL, RIDGE and RIVER states is obtained. We aim to get only the
3.1 The Reverse Midpoint Displacement

Many midpoint displacement methods [5] can be useful to fill the remaining NULL state coordinates but the resulting terrain will either contains artifacts and discontinuities (ex.: The Triangle-Edge Subdivision, The Diamond-Square Subdivision) or modifies the already computed coordinates (ex.: The Square-Square Subdivision). By pre-filling a part of the remaining NULL state coordinates we find that these two first methods, The Triangle-Edge and The Diamond-Square give better interpolation results.

We choose one of Triangle-Edge or Diamond-Square subdivision methods and use it, as described in [5], with our midpoint displacement function fct(square). This function recursively interpolates all NULL state coordinates in the square square; it uses the chosen subdivision method.

The Reverse Midpoint Displacement processes the Digital Elevation Map array DEM[N][N] with a scan-line order and a step s varying from 1 to 2^n with N = 2^n + 1; so s takes at least n + 1 different values. For each s, the map is considered as the minimal covering set of s-side squares. For each set of s-side squares, a partial interpolating mesh is computed in two steps:

- If one (respectively two or three) corner of each s-side square has a non-NUL state, then the other three (respectively two or one) corners are computed by interpolating first mentioned corner’s altitude;
- If one corner of each s-side square has a non-NULL state, the the md_fct is called for this square.

Figure 4 and 5 show respectively the stage 4 (s = 2^3 and N = 513) and the last stage (s = 2^9) of the Reverse Midpoint Displacement process.

We declare DEM[N][N] as a new empty Digital Elevation Map, ur() a uniform random number generator function in the [-1,1] interval and fct(integer s) as our Reverse Midpoint Displacement function. Here is the body of this function:

1. copy DEM[N][N] in C DEM[N][N];
2. for i ← 0 to N - 1 step s;
   (a) for j ← 0 to N - 1 step s;
      i. square ← the square with top-left corner on (i, j) and side equal to s;
      ii. nn_s ← for each corner of square in DEM: get the number of non-NUL coordinates;
3. if nn_s > 0 then;
   • a ← for each corner of square in DEM: the altitude average of all non-NUL coordinates;
   • for each corner (1, m) of square with a NULL coordinate:
     − altitude(C DEM[i][m]) ← a + s * ur();
     − state(C DEM[i][m]) ← DONE;
4. if s = 1 then goto 6;
5. for i ← 0 to N - 1 step s;
   (a) for j ← 0 to N - 1 step s;
      i. square ← the square with top-left corner on (i, j) and side equal to s;
      ii. nn_s ← for each corner of square in DEM: get the number of non-NUL coordinates;
6. s ← 2 * s;
7. if s < N then rmd_fct(s)

The function rmd_fct(1) is used to process the Digital Elevation Map shown on figure 3. The resulting landscape is shown on figure 6.

Figure 4: Stage 4 of 10 of the Reverse Midpoint Displacement process (s ← 2^i, i ∈ \{0,1,2,3\}). Points either in blue or in pink are already computed. The green points still remain to compute.

4. CONCLUSIONS

This paper has presented a novel technique for automatic modeling of landscapes with ridges and rivers. We have demonstrated an extension of the basic midpoint displacement algorithm that allows generating realistic terrains constrained by a predefined ridges and rivers network. Thus, this technique is efficient enough to model high resolution landscapes in reasonable time. Computing the model shown on figure 5 took 0.45 seconds on a personal computer.
Figure 5: Final result (stage 10 of 10) of the Reverse Midpoint Displacement process \((s \leftarrow 2^i, i \in \{0, \ldots, 9\})\).

Pentium IV / 3Ghz / 512Mo — 0.25 seconds to compute the ridges and rivers network and 0.20 seconds to compute the Reverse Midpoint Displacement.

Future works are aimed toward two main goals:

- Creating a toolbox in order to design ridge lines, valleys and actually the corresponding rivers network. Thus, the user can interactively model and generate his own landscape;

- Reproducing real landscapes by either retrieving or detecting ridge lines and rivers network on satellite data. By using controllable noise generators, our extension could actually serve in data compression.

5. REFERENCES


