Modeling landscapes with ridges and rivers: bottom up approach

Conference Paper · December 2005

DOI: 10.1145/1101389 · Source: DBLP

2 authors, including:

Farès Belhadj
Université de Vincennes - Paris 8
16 PUBLICATIONS **124** CITATIONS

SEE PROFILE

Modeling Landscapes with Ridges and Rivers: bottom up approach

Farès Belhadj* Université Paris 8

Pierre Audibert† Université Paris 8

Abstract

We present a new fractal based method of generating realistic models of natural landscapes. In this paper, we describe a three-steps method that produces a terrain mesh using ridge lines and rivers drainage network. As opposed to previous methods that compute water erosion for a given terrain model — Digital Elevation Map —, our terrain mesh is generated constrained by a precomputed set of ridge lines and rivers network. A skeleton of a *ridges and rivers network* is computed and stored in a Digital Elevation Map (**D.E.M.**) as initial values. Then, the elevations data set is enriched using a novel interpolation method based on a *Midpoint Displacement's Inverse* process. Finally, a midpoint displacement subdivision is applied to interpolate the remaining elevation coordinates. The resulting terrain meshes lead to naturalistic landscape models and our method seems to be very promising. Images produced by our model are presented.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation— [I.3.5]: Computer Graphics—Computational Geometry and Object ModelingPhysically based modeling — Curve, surface, solid, and object representationsI.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Fractals

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

1 Introduction

Modeling realistic approximations of natural landscapes is a long standing problem in computer graphics. Well known interpolation methods [Fournier et al. 1982; Miller 1986] are powerful enough to generate plausible models of non-eroded mountains. In 1988, Kelley et al. [Kelley et al. 1988] overcame this limitation by generating a water drainage network to modify the initial terrain surface. An Alternative method based on the simulation of hydraulic and thermal erosion was proposed by Musgrave et al. [Musgrave et al. 1989]. In "MOJOWORLD: *Building Procedural Planets*" [Ebert et al. 2003] Musgrave et al. gave a review of methods used to generate *procedural planets*; interesting *ridged monofractal* and *multifractal* terrain models are studied and various image results are shown.

In general, the methods based on an erosion model process a precomputed terrain to produce an eroded terrain model. Despite simplifications, these methods still remain too strict and also complex to manipulate. In[Prusinkiewicz and Hammel 1993] Prusinkiewicz and Hammel described a method — in a single integrated process — that models mountains with rivers using context-sensitive rewriting mechanisms. Their approach still remains original and promising but the obtained terrain models suffer from either artifacts or lack of water erosion.

In this paper we introduce a new 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. This method involves three main stages. First, a simple and rapid process that generates the skeleton of a *ridges and rivers network* (**skeleton—D.E.M.**) is employed to initialize the terrain **D.E.M.**. Then, an extension of the basic Midpoint Displacement (**M.D.**) model is used to enrich the elevations data set. Finally, by using this basic **M.D.** subdivision model, we interpolate the remaining elevation coordinates. Related to a *Midpoint Displacement's Inverse* (**M.D.I.**) process, our **M.D.** extension computes new elevations coordinates in order to avoid discontinuities that appear while a single **M.D.** process is applied on a pre-filled **D.E.M.** (see Figure 3).

Thus, our novel algorithm — combination of the **M.D.I.** and the **M.D.** processes — allows generating artifact-less landscape models around a pre-filled **D.E.M.** (a naturalistic landscape model is shown on Figure 7 and a surrealistic landscape model is shown on Figure 8).

2 Modeling Ridges and Rivers Network

The first part of our modeling process focuses on the problem of generating the initial **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.

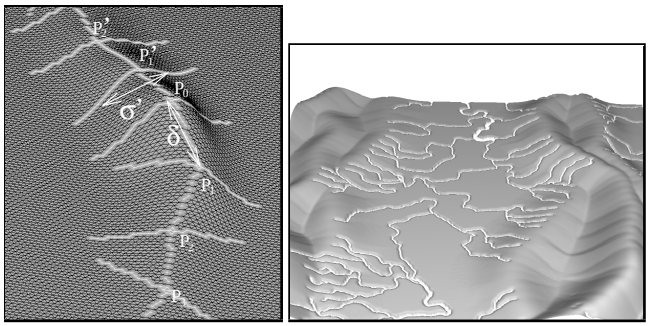
2.1 Ridge Lines

Starting with an *empty* **D.E.M.** (all elevations still remain to compute), 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_0 with relatively high altitude y_0 and opposite initial orientation angles. For a given step δ and at each iteration i, particles are subject to side impacts in way to describe a Fractional Brownian Motion [Fournier et al. 1982] on the xz plane. The particle altitude decreases according to the covered distance $i \times \delta$ on a Gaussian distribution $\mathscr{G}_1$ centered at the initial position and with a standard deviation $\sigma = \frac{1}{4} \times width(DEM)$. Thus during the particle displacement from P_i to P_{i+1}, we modify the map elevations according to the successive particle altitudes on the $[P_i, P_{i+1}]$ segment. For each point p on this segment, we draw a Gaussian curve $\mathscr{G}_2(p, \sigma' = \frac{1}{4} \times \sigma)$ perpendicular to $[P_i, P_{i+1}]$ and with an amplitude equal to $altitude(p)$. As shown on Figure 1a, for two opposite *Ridge Particles*, $r_i(P_0 \rightarrow P_1 \rightarrow ...)$ and $r_{i+1}(P_0 \rightarrow P'_1 \rightarrow ...)$ we obtain one ridge line by the deformation of the terrain mesh (modifying the **D.E.M.** elevations) across r_i and r_{i+1} routes.

Particle motion is stopped when its altitude became null or when it collides with an other particle route.

*e-mail: amsi@ai.univ-paris8.fr

†e-mail: audibert@ai.univ-paris8.fr



(a) (b)

Figure 1: (a) 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 to P'_1. (b) A randomly generated ridges and rivers network. The ridges are blended in order to compute the rivers network.

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 a gravity acceleration — *m* serves in weighting the river width and its depth. *Rivers Particles* are randomly dispatched at the top of the ridge lines. A simple physically based method is used to model these particles motion (in the manner of *"obtaining velocity field of water flow"* in [Chiba et al. 1998]); here the negative *y* axis acceleration is more weighted and a friction power is added. During a particle motion, each of its discrete positions is stored in a coordinates list; for a *River Particle* r_i, its successive discrete positions are stored in $path(r_i)$. These information are also stored in the map. Thus, by using a unique identifier for each *River Particle*, intersections of particles paths can be easily detected. Then, when a particle r_i intersects an other particle route $path(r_j)$, r_i is stopped and $path(r_j)$ is backtracked until the intersection point and the map is updated. r_j properties are modified such as[1] :

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

The process is stopped when all the particles either become static or exit from the terrain limits. In the next process, we only consider particles that exit from the terrain limits (see Figure 1b).

2.3 The skeleton of the ridges and rivers network

At this point, a plausible ridges and rivers network is generated and stored on the map; the ridges on Figure 1b are blended using a Gaussian distribution to allow us to compute the rivers network. For the next part of the terrain generation process, we just keep the skeleton of this network and reinitialize all other points of the map (see Figure 2).

Thus, only 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;

[1] At the intersection point, the particle r_i becomes a source for the river path described by r_j.

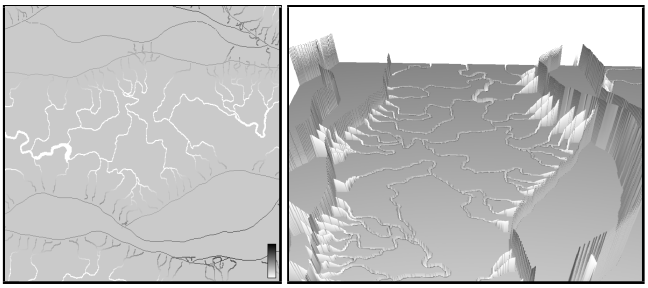


Figure 2: The skeleton of the ridges and rivers network: the **skeleton—D.E.M.** is shown on the left image; the right image shows a tridimensional view where rivers are rendered in white.

- *RIDGE* : the coordinate is on a *Ridge Particle* path, its value is equal to the ridge altitude at that position;
- *RIVER* : the coordinate is on a *River Particle* path, its value is equal to the river altitude at that position;
- *DONE* : the coordinate is computed.

Finally, a Digital Elevation Map containing *NULL*, *RIDGE* and *RIVER* states is obtained. We aim to get only the coordinates with a *NULL* state computed[2].

3 The Interpolation Process

The previous section describes a simple method that generates ridge lines and computes the associated water flow network. This information is now stored in the Digital Elevation Map (the **skeleton—D.E.M.**). Thus, new elevations should be interpolated in order to obtain a realistic fractal based terrain model. This interpolation method involves two steps (stage two and three of the main method): 1) the **M.D.I.** process enrich the elevations data set in order to produce enough data for 2) the **M.D.** process. The **M.D.I.** method avoids appearance of discontinuities on the resulting model. Figure 3 shows the resulting **D.E.M.** for a single **M.D.** process; figure 5 shows the resulting **D.E.M.** for a single **M.D.I.** process and Figure 6 shows the final result obtained with our two-process algorithm.

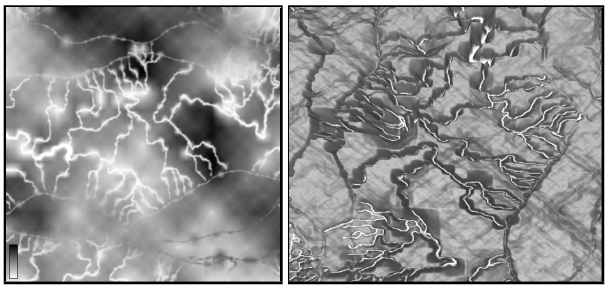


Figure 3: Resulting image after processing a basic midpoint displacement method on the **skeleton—D.E.M.**. Discontinuities are visible at the neighborhood of the ridges and rivers network.

[2] compute a coordinate remains on interpolating its map elevation, thus its new state is *DONE*.

3.1 The Midpoint Displacement's Inverse process

The **M.D.I.** process is based on a simple but powerful observation. When we process a midpoint displacement subdivision on a given square, the square midpoints (children) are interpolated using the elevations of its corners. So points (parents) that serve to interpolate a midpoint can be stored in a list. Thus, for a given square side `S` (the root square is assimilated to the **D.E.M.**), we pre-compute recursively for each midpoint its parent-list. We call `parent(x,z)` the parent-list of the `(x,z)` point, it contains the triplets `(px,pz,d)` where `(px,pz)` is a parent coordinate and `d` the euclidean distance from `(px,pz)` to `(x,z)`. Thus, in the inverse process, we can interpolate (if the state is *NULL*) a point elevation by using its available children elevations. Figure 4 shows **M.D.I.** process for the triangle-edge subdivision.

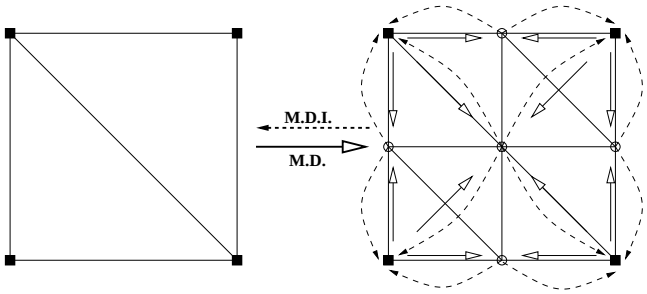


Figure 4: The Triangle-Edge Subdivision and its inverse process.

3.2 Choosing a Midpoint Displacement Method

Many Midpoint Displacement methods [Miller 1986] can be useful to fill the remaining *NULL* state coordinates but, as shown on Figure 3, the resulting terrain either contains artifacts or discontinuities (ex.: *The Triangle-Edge Subdivision*, *The Diamond-Square Subdivision*) or modifies the already computed coordinates (ex.: *The Square-Square Subdivision*). By pre-filling, with the **M.D.I.** process, 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 [Miller 1986], with our midpoint displacement function `md_fct(square_t square)`. This function recursively interpolates all *NULL* state coordinates in the square `square`.

3.3 The Algorithm

For a given **skeleton—D.E.M.**: `DEM[N][N]`, we declare:

- `urand()` a uniform random number generator function in the $[-1,1]$ interval;
- `avElevations(x,z)`: the function that returns the list `avList[x][z]` of available children information for the `(x,z)` coordinates. A doublet `(child_elevation, distance)` is given for each available child. Children with *NULL* states do not appear in this list;
- `addElevation(x,z,cx,cz)` the function that adds in `avList[x][z]` the elevation `DEM[cx][cz]` and the euclidean distance from `(x,z)` to `(cx,cz)`;

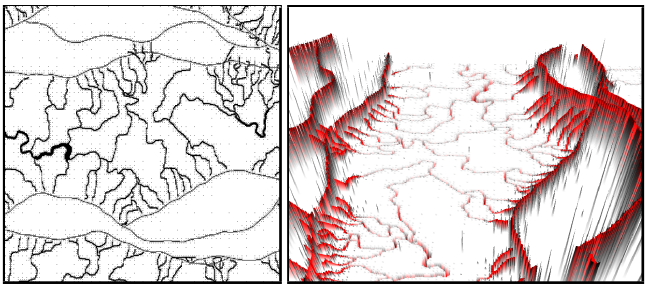


Figure 5: The Midpoint Displacement's Inverse process. The left image shows: in black color the computed elevations and in white color the elevations that still have to be computed. The right image shows the tridimensional view of the **D.E.M.**: in black/gray color elevations computed with the **M.D.I.** process (the peaks around the ridges and rivers network (in red)) and in white color the elevations that still have to be computed.

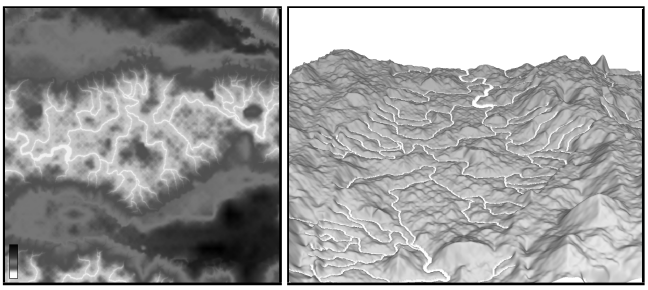


Figure 6: The final result obtained after the interpolation process. By pre-filling (enriching) the **skeleton—D.E.M.** using the **M.D.I.** process, the midpoint displacement interpolation becomes more efficient; discontinuities and artifacts that appear on Figure 3 are completely avoided.

- we use a FIFO queue to store a coordinates list; so additional functions are declared:
  - `sizeQueue()` the function that returns the number of elements in the queue;
  - `putQueue(x,z)` the function that adds the coordinates `(x,z)` at the end of the FIFO queue. Only one occurrence is added;
  - `getQueue()` the function that extracts the first element `(x,z)` from the FIFO queue. This function deletes the extracted element from the queue;
  - `eltQueue(n)` the function that returns a copy of the n^{th} element `(x,z)` of the FIFO queue. This function leaves the element in the queue;

Here is the main loop of our interpolation process:

1. for `i` $\leftarrow$ 0 to `N-1`;
   (a) for `j` $\leftarrow$ 0 to `N-1`;
      i. if `state(DEM[i][j])` <> *NULL* then `putQueue(i,j)`;
2. while `sizeQueue()` > 0 ;
   (a) `sq` $\leftarrow$ `sizeQueue()`;
   (b) for `i` $\leftarrow$ 1 to `sq`;
      i. `C` $\leftarrow$ `getQueue()`;
      ii. for `P` in `parent(C.x,C.z)`;
         A. if `state(DEM[P.x][P.z])` <> *NULL* then continue;
         B. `addElevation(P.x,P.z,C.x,C.z)`;

```
            C. putQueue(P.x,P.z);
      (c) sq ← sizeQueue();
      (d) for i ← 1 to sq;
            i. E ← eltQueue(i);
           ii. v ← 0 , d ← 0 , n ← 0;
          iii. for V in avElevations(E.x,E.z);
                 A. v ← v + V.child_elevation;
                 B. d ← d + V.distance;
                 C. n ← n + 1;
           iv. elv ← v/n + d/n × urand();
            v. elvation(DEM[E.x][E.z]) ← elv;
           vi. state(DEM[E.x][E.z]) ← DONE;
  3. md_func(square(DEM));
```

4 Conclusions

This paper has presented a novel method for automatic modeling of landscapes with ridges and rivers. We have demonstrated an extension of the basic midpoint displacement algorithm that allows generating realistic (as shown on Figure 7) and surrealistic (as shown on Figure 8) terrains constrained by a predefined ridges and rivers network. Our method is efficient enough to model high resolution landscapes in a reasonable time — generating a 1024×1024 terrain mesh takes less than one second on a personal computer.

Topics for future study aimed toward reproducing real landscapes starting from small (example: too low resolution **D.E.M.**) or partial (example: only ridges and/or rivers information are available) terrain data sets; these data can be either retrieved or detected on satellite data. Thus, by using controllable noise generators, our algorithm could actually serve in data compression.

References

CHIBA, N., MURAOKA, K., AND FUJITA, K. 1998. An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation 9*, 4, 185–194.

EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. 2003. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, ch. 20 – MOJOWORLD: Building Procedural Planets, 565–615.

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

KELLEY, A. D., MALIN, M. C., AND NIELSON, G. M. 1988. Terrain simulation using a model of stream erosion. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 263–268.

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. *Computer Graphics 23*, 3, 41–50.

PRUSINKIEWICZ, P., AND HAMMEL, M. 1993. A fractal model of mountains with rivers. 174–180.

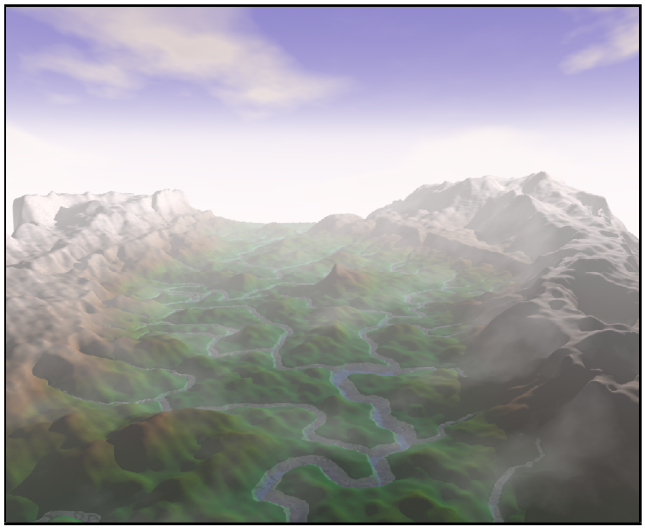


Figure 7: Generating a naturalistic landscape model: the textured rendering of our example model.

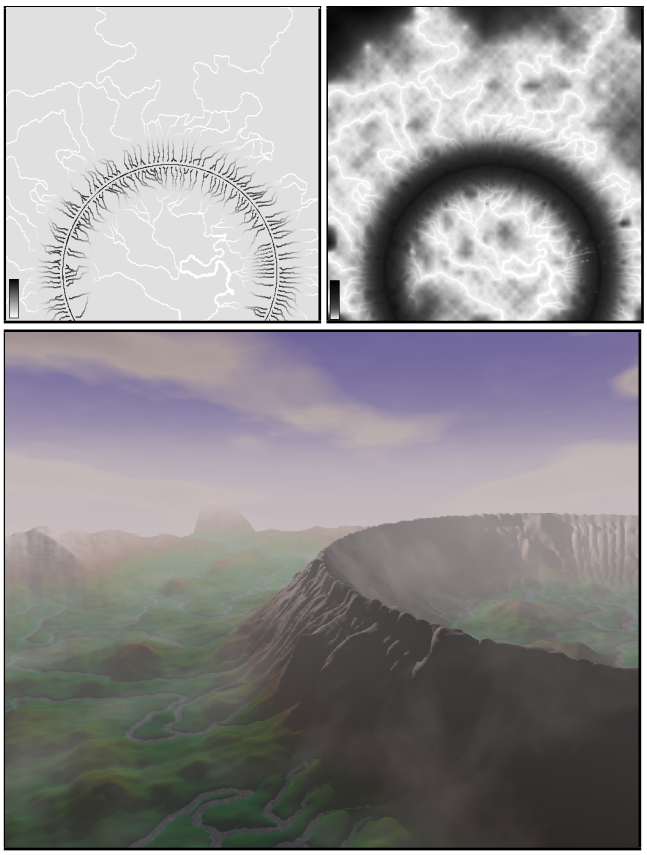


Figure 8: A surrealistic landscape model: the top left image shows the used **skeleton—D.E.M.** — one *Ridge Particle* is used to draw a partial circle and the rivers network is computed around it; the top right image shows the resulting image after the interpolation process ; the final result is shown on the bottom image.