A Fractal Model of Mountains with Rivers

Przemyslaw Prusinkiewicz and Mark Hammel

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail addresses: pwp@cpsc.ucalgary.ca, hammel@cpsc.ucalgary.ca

ABSTRACT

This paper addresses the long-standing problem of generating fractal mountains with rivers, and presents a partial solution that incorporates a squig-curve model of a river's course into the midpoint-displacement model for mountains. The method is based on the observation that both models can be expressed by similar context-sensitive rewriting mechanisms. As a result, a mountain landscape with a river can be generated using a single integrated process.

KEYWORDS: modeling of natural phenomena, terrain models, midpoint displacement, squig curve, context-sensitive geometric rewriting.

INTRODUCTION

In 1988, Mandelbrot pointed out "the most basic defect of past fractal forgeries of landscape" — the fact that each of them fails to include river networks [12, pages 243–260]. Since then, Kelley, Malin and Nielson [5] overcame this limitation by generating fractal terrain around a predefined drainage system. Pursuing an alternative approach, Musgrave, Kolb, and Mace [11] created river channels by simulating water erosion in fractal mountains. Both methods require separate processes to define the mountain and the river system. A different technique was introduced by Bardeen, whose program *Panorama* [1] combines mountain and river generation into a single process. The details of his algorithm have not yet been published.

In this paper we introduce a method that — like Bardeen's — creates the mountain and the river system simultaneously. Specifically, we combine the midpoint-displacement method for mountain generation given by Fournier, Fussell, and Carpenter [3] with the squig-curve model of a non-branching river originated by Mandelbrot [7, 8] (see also [9, Chapter 24]). Our method employs a context-sensitive rewriting system operating on geometric objects. Theoretical interest in such systems has been spawned by Smith [14], but few examples have been investigated to date. Consequently, a part of our paper is devoted to the discussion of the context-sensitive aspects of the constructions under consideration.

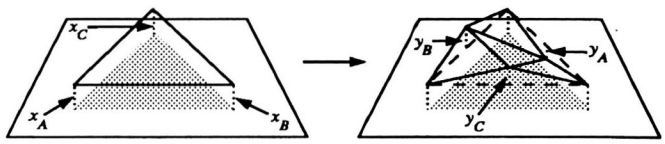

Figure 1: A production for fractal mountain generation using the midpoint displacement method. The initial altitudes x_A, x_B and x_C of the vertices of a subdivided triangle, and the displacement values y_A, y_B and y_C, vary between instances of production application.

We begin with a review of the basic midpoint-displacement construction. A description of the squig-curve construction follows. The two constructions are then related as different facets of the same context-sensitive process of triangle subdivision, and combined into a model of mountains with rivers. The paper is concluded by a list of topics open for further research.

MIDPOINT-DISPLACEMENT METHOD REVISITED

In the simplest version of the midpoint-displacement method, an initial horizontal triangle is subdivided into four smaller triangles, and the newly created vertices are displaced vertically by random values. A similar process is repeated for each of the smaller triangles, then for each of their descendants, until a given recursion limit is reached. Smith [14] presented midpoint displacement as a rewriting process governed by the class of productions depicted in Figure 1. This characterization related fractal mountain generation to formal language theory, and raised a question regarding the nature of the rewriting process in hand: Is it context-free or context-sensitive? Smith wrote: "In formal language theory, as Loren Carpenter has pointed out to me, the problem with his language is that it is context-free. Information internal to an original database triangle is never passed to neighboring triangles." Although this view has been supported in the literature [10], [12, page 244], it disregards a form of information transfer that *does* take place between neighboring triangles. As shown in Figure 2, we cannot apply the production of Figure 1 independently to triangles P and Q sharing a common edge l, since the displacement of the midpoint of l must

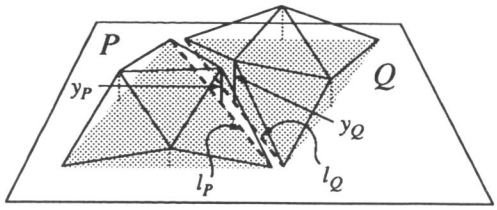

Figure 2: The midpoint-displacement method is context-sensitive. After the subdivision of triangles P and Q, the midpoints of the coinciding edges l_P and l_Q are displaced by vectors of equal lengths, $y_P = y_Q$. In practical implementations, lines l_P and l_Q collapse to a single edge l shared between triangles P and Q.

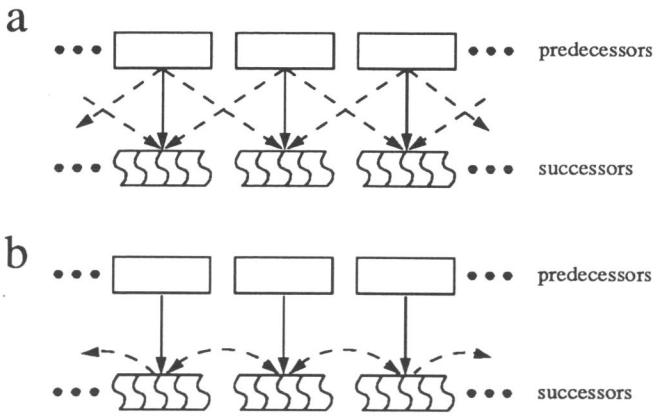

Figure 3: Information transfer in context-sensitive parallel grammars (a) and in the midpoint-displacement method (b). Continuous arrows represent the flow of information from the parent object to its replacement. Dashed arrows represent the flow of contextual information.

be the same for both P and Q [3]. Thus, the production instance applied to triangle P depends on that applied to triangle Q, and *vice versa*. This implies information transfer between P and Q, although its nature is different from that usually considered in formal language theory (specifically, in the theory of L-systems, which deals with parallel rewriting [6]). Traditionally, the set of applicable productions is constrained by the neighbors of the *strict predecessor* (the symbol being replaced). On the other hand, in midpoint displacement each production is constrained by the *successors* of the productions applied concurrently to the neighboring triangles (Figure 3). Nonetheless, in both cases the outcome of a production depends on its neighbors, and in this sense both production types are context-sensitive.

The information transfer associated with context-sensitive productions used in the midpoint-displacement method is further illustrated in Figure 4. The productions (a) are assumed to raise midpoints of the subdivided edges by 1, 2, and 3 units, counting counterclockwise. Distinguishing between possible orientations of the subdivided edges, we obtain six instances of production (a), as shown in (b). During the construction of a mountain, the production instance applied, say, to the bottom left triangle determines production instances appropriate for subdividing all other triangles (c, d). Thus, information is passed between the bottom left triangle and all other triangles in the mesh.

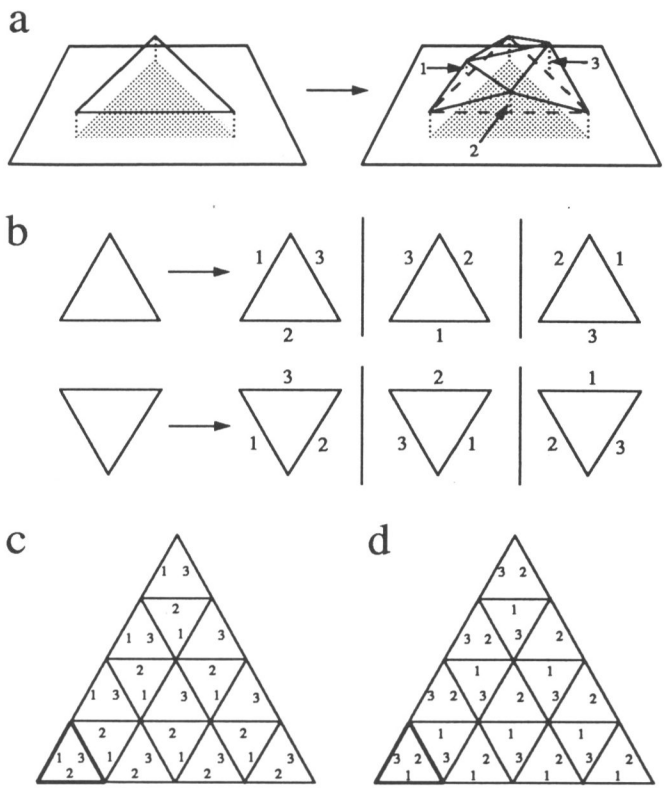

Figure 4: Information transfer in the midpoint-displacement method. Production (a) has six instances (b). The subdivision pattern of the bottom left triangle determines the subdivision patterns for all remaining triangles, as shown here for the third level of recursion (c, d).

Note that Figures 4 (c) and (d) can also be viewed as tilings using tiles with labeled edges. Coinciding edges must have the same label. A square-grid counterpart of such tilings was proposed and studied by Wang [15, 16] (see also [4, Chapter 11]), who showed that the operation of any Turing machine can be simulated using a set of appropriately labeled tiles. Clearly, a context-sensitive information-passing mechanism is needed to achieve this computational power.

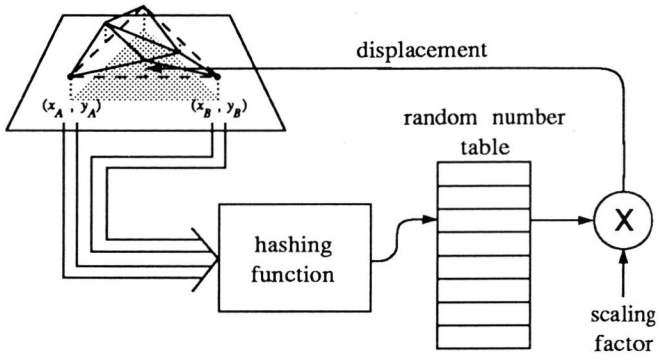

Figure 5: Deterministic calculation of a psuedorandom midpoint-displacement value.

The context-sensitive nature of the midpoint-displacement method may be hidden by programming techniques used to implement it. For example, Fournier *et al.* wrote [3]: "An obvious requirement is that the same random displacements must be generated on each boundary, which can be accomplished by tying the seeds of the random number generator to identifiers of points on the boundary, making certain that the same identifiers are assigned to the corresponding points in the representation of each polygon's boundary." A deterministic variant of this technique, suggested by Smith [14], is illustrated in Figure 5. When an edge is subdivided, the coordinates of its endpoints determine, via a hashing function, an index into a prestored table of random numbers that represent possible displacement values. Thus, if the displacement of the midpoint of edge l in Figure 2 is calculated separately for both triangles P and Q that share l, the returned values y_P and y_Q will be the same. This technique replaces the explicit context-sensitivity with the dependence of production parameters on the position of the subdivided triangle in the underlying coordinate system. Consequently, the deterministic midpoint-displacement method can be implemented in a simple, recursive manner.

SQUIG CURVES REVISITED

Mandelbrot introduced squig curves as "a model of a river's course, patterned after the well-known pictures in geology or geography that show the successive stages of a river that burrows into a valley, defining its course with increasing precision" [9, page 255]. Peyrière [13] (see also [2]) proposed to consider squig curve construction as a random rewriting process, governed by the set of productions depicted in Figure 6. The production predecessor is a triangle with the edges labeled *entry*, *exit*, and *neutral*. The entry edge represents the set of possible sites where the curve may enter this triangle, and the exit edge represents the set of possible sites where the curve may leave it. The neutral edge is not intersected

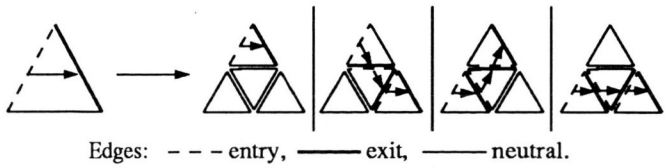

Figure 6: Productions for generating a squig curve. Before production application, the predecessor and the successor may be translated, rotated, reflected, and scaled. Arrows indicate the direction of the curve (river flow). Vertical bars separate alternative production successors.

by the curve. Each production subdivides the predecessor triangle into four smaller triangles, satisfying the following constraints:

- The entry edge of the predecessor is subdivided into an entry edge and a neutral edge;
- The exit edge of the predecessor is subdivided into an exit edge and a neutral edge;
- The neutral edge of the predecessor is subdivided into two neutral edges;
- Each pair of coinciding edges inside the subdivided triangle consists either of an entry and an exit edge, or of two neutral edges.

Figure 6 shows a set of four productions satisfying these criteria. The squig curve construction begins with a triangle that has one entry and one exit edge. A production application partitions these edges into two equal segments while subdividing the triangle. For each original edge crossed by the river, one of the new segments is selected as the next approximation of the crossing site. Once the entering and exit segments have been chosen, the path of the river through the new triangles is uniquely defined, assuming that a river may go through each triangle at most once. The subdivision process is repeated recursively until the desired level of detail is reached.

As shown in Figure 7, the segments crossed by the river must be chosen consistently for each pair of adjacent triangles, so that the exit segment from one triangle matches the entry segment of the neighboring triangle. Thus, triangle subdivision during squig curve construction is a context-sensitive process similar in nature to midpoint displacement. In both cases, a consistent decision regarding the edge shared by two triangles must be reached, whether it determines the altitude to which the midpoint will be raised, or the edge segment through which the squig curve will pass. Consequently, a

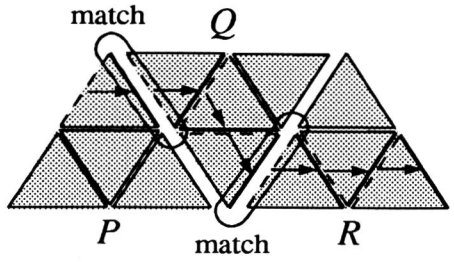

Figure 7: Consistent subdivision of three triangles P, Q, and R during a squig curve construction. After the subdivision, the exit edge from triangle P matches the entry edge to triangle Q and, in a similar way, the exit edge from triangle Q matches the entry edge to triangle R.

squig curve can be constructed in a manner analogous to the deterministic implementation of the midpoint displacement method (Figure 5). The location of the vertices of an edge crossed by the river is used as a key into a hash table of random numbers. The sign of the returned number determines which segment will be crossed in the next approximation of the riverbed.

The above algorithm guarantees that the river will run continuously through the mesh of triangles and will never intersect itself. A sample squig curve construction is illustrated in Figure 8.

INTEGRATION OF A RIVER AND A MOUNTAIN

The previous two sections demonstrate that the midpoint-displacement method and the construction of a squig curve can be viewed as variants of the same context-sensitive subdivision of a triangle. This suggests the combination of both constructions into a single algorithm. At each subdivision step, the path of the river and the shape of the mountain are specified with increased accuracy. When a triangle is subdivided to the n-th level of recursion, the midpoints of the edges crossed by the squig curve are assigned the minimum altitude $\mathrm{alt}(n)$, calculated as the sum of negative displacement limits d_i in the previous and current subdivision steps:

$$\mathrm{alt}(n) = \sum_{i=1}^{n} d_i \ .$$

The remaining midpoints are not affected by the river's course and are displaced in the usual pseudorandom way. The resulting algorithm for generating a mountain traversed by a river is illustrated in Figure 9. Note that if the top view of a mountain is regarded as a planar graph, at each level of recursion there is a path from its initial entry edge to the final exit edge, formed by the chain of vertices assigned the minimum altitude. For example, in Figure 9(c) this path runs through

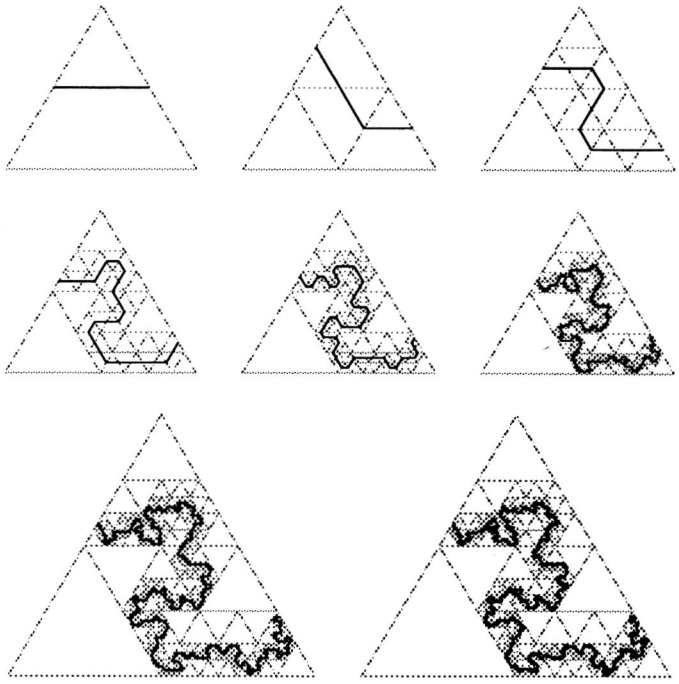

Figure 8: Example of a squig curve construction (recursion levels 0–7).

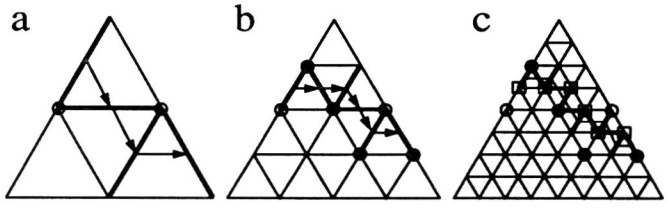

Figure 9: Vertices affected by a river's course at the first, second, and third level of the recursive construction of a fractal mountain with a river.

the vertices marked by a square box. Since these vertices have the same altitude, and are guaranteed to be the lowest of all points in the landscape, we can interpret the path that connects them as a riverbed. The river will never run upwards and will always lie lower than the surrounding terrain, thus satisfying two obvious constraints that a real river must meet.

A mountain landscape with a clearly defined river channel can be convincingly approximated using six or seven recursion levels. At nine or ten levels the results are more realistic, but the number of polygons is much larger (over 1 million for ten levels). Two sample landscapes created using the proposed method at ten levels of recursion are shown in Plates 1 and 2. The colors of the vertices, including the river, were determined by their altitudes serving as indices into an

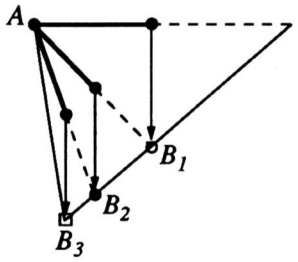

Figure 10: The emergence of an asymmetric valley, shown in cross-section. Consecutive approximations of the riverbed may move it horizontally towards a high-altitude point A, creating a succession of increasingly steep walls AB_1, AB_2, and AB_3. The arrows indicate the vertical displacements of the midpoints of the selected edges.

appropriate color map; the triangles were filled using Gouraud shading.

CONCLUSIONS

We have presented a technique for generating fractal mountains with rivers that combines the midpoint-displacement method for mountain generation with the squig-curve model of a river's course. Using this technique, we were able to achieve some degree of realism in the synthesized landscapes. Nevertheless, three key problems remain open:

- *The river flows at a constant altitude.* This assumption, although physically incorrect, could be viewed as an approximation for a river with a small slope, such as one flowing in the plains. However, in mountain landscapes the slope should not be neglected — for example, to make waterfalls possible.

- *The river flows in an asymmetric valley.* The algorithm tends to produce asymmetric river valleys in the shape of an italicized letter V - with one side almost vertical. This phenomenon, clearly visible in Plates 1 and 2, results from the river approaching a mountain vertex placed at a high altitude (Figure 10). In nature, both sides tend to be more symmetric and less steep.

- *The river has no tributaries.* The squig-curve construction can be extended with productions that introduce branching points and subdivide triangles that already include such points (Figure 11). Sample planar curves generated this way are shown in Figure 12. Unfortunately, it is not immediately apparent how these curves could be incorporated into fractal mountains. A model of a river source would be necessary, since the tributaries usually originate within the visualized area.

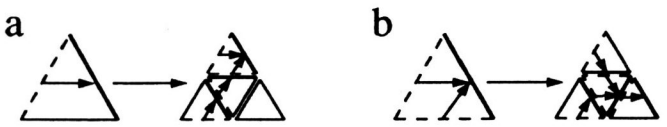

Figure 11: Sample productions for generating branching extensions of squig curves. Production (a) creates a tributary. Production (b) subdivides a triangle that already includes a branching point.

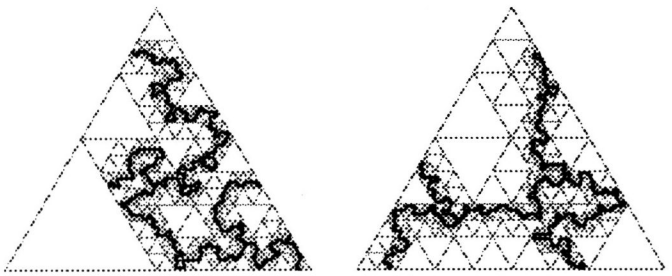

Figure 12: Extended squig curves with branches.

In addition, the images could be improved using more sophisticated rendering techniques.

ACKNOWLEDGMENTS

We would like to thank Ken Musgrave for outlining to us the state of the art in the modeling of landscapes with rivers, James Bardeen for a description of his software for landscape synthesis, and Benoit Mandelbrot, as well as Jules Bloomenthal, for helpful comments on our initial manuscript. We also gratefully acknowledge the support from the Natural Sciences and Engineering Research Council of Canada in the form of a research grant, a graduate scholarship, and equipment grants.

REFERENCES

[1] J. M. Bardeen. Panorama User's Manual. Manuscript, 1992.

[2] F. M. Dekking. Substitutions, branching processes and fractal sets. In J. Bélair and S. Dubuc, editors, *Fractal Geometry and Analysis*, NATO ASI Series, pages 99–119. Kluwer Academic Publishers, Dordrecht, 1991.

[3] A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–383, 1982.

[4] B. Grünbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman and Company, New York, 1987.

[5] A. D. Kelley, M. C. Malin, and G. M. Nielson. Terrain simulation using a model of stream erosion. Proceedings

of SIGGRAPH '88, in *Computer Graphics* 22, 4, pages 263–268, ACM SIGGRAPH, New York, 1988.

[6] A. Lindenmayer. Developmental algorithms: Lineage versus interactive control mechanisms. In S. Subtelny and P. B. Green, editors, *Developmental order: Its origin and regulation*, pages 219–245. Alan R. Liss, New York, 1982.

[7] B. B. Mandelbrot. Les objets fractals. *La Recherche*, 9:1–13, 1978.

[8] B. B. Mandelbrot. Colliers alléatoires et une alternative aux promenades aux hasard sans boucle: les cordonnets discrets et fractals. *Comptes Rendus (Paris)*, 286A:933–936, 1979.

[9] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Company, San Francisco, 1982.

[10] G. S. P. Miller. The definition and rendering of terrain maps. Proceedings of SIGGRAPH '86, in *Computer Graphics* 20, 4, pages 39–48, ACM SIGGRAPH, New York, 1986.

[11] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrain. Proceedings of SIGGRAPH '89, in *Computer Graphics* 23, 3, pages 41–50, ACM SIGGRAPH, New York, 1989.

[12] H.-O. Peitgen and D. Saupe, editors. *The Science of Fractal Images*. Springer-Verlag, New York, 1988.

[13] J. Peyrière. Processus de naissance avec interaction des voisins, évolution des graphes. *Annales de l'Institut Fourier*, 31:187–218, 1981.

[14] A. R. Smith. Plants, fractals, and formal languages. Proceedings of SIGGRAPH '84, in *Computer Graphics*, 18, 3, pages 1–10, ACM SIGGRAPH, New York, 1984.

[15] H. Wang. Proving theorems by pattern recognition. *Bell System Technical Journal*, 40:1–42, 1961.

[16] H. Wang. Games, logic, and computers. *Scientific American*, 11:98–106, 1965.

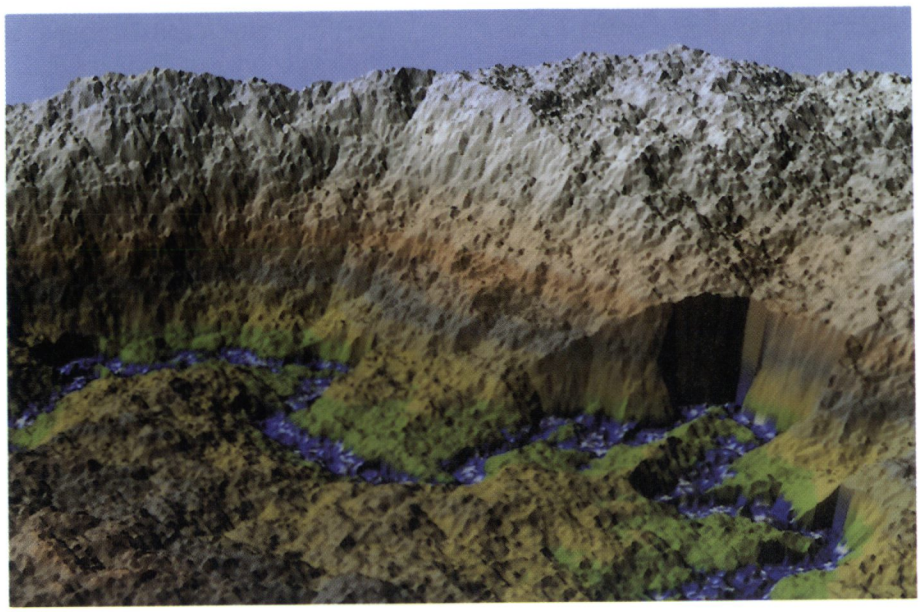

Plate 1: A fractal landscape with a river. This image was generated on a Silicon Graphics VGX 3D/310 workstation at 10 levels of recursion in approximately 8 minutes.

Plate 2: Another fractal landscape with a river.