Terrain Erosion Model Based on Rewriting of Matrices

Ivo Marák, Bedřich Beneš and Pavel Slavík
Department of Computer Science, Czech Technical University,
Karlovo nám. 13, Prague, Czech Rep.
e–mail: {marak|benes}@sgi.felk.cvut.cz
slavik@cslab.felk.cvut.cz

January 15, 1997

Abstract

This paper introduces a new approach to erosions of models of terrains. This approach is based on two dimensional generalization of the rewriting process - array grammars. A height field representing the terrain is considered as the matrix and context sensitive rewriting of this matrix, represents the erosion process. This approach is quite general and allows us to use an arbitrary kind of erosion which can be expressed by means of rewriting productions.

1 Introduction

Mandelbrot mentioned one of the curiosities of fractal terrains: fractal terrains look like fractal terrains even when they are turn upside down [5]. This property is typical for geologically new terrains in nature. Indeed, the fractal techniques provide terrains which appear as though they were "just created". The terrains we meet in our life are mostly quite different. They are maggeted by water, attrited by particles of sand and dust flown in the wind, and crannied from influence of temperature amplitude. Most of the terrains are also influenced by human factors. Most of the natural influences are reflected as a kind of *erosion*. This problem has been addressed, *e.g.* in [3, 4, 6, 8]. Mandelbrot also pointed out that most of the fractal methods fail to include river networks [5]. The rivers represent another kind of erosion [4, 11].

We introduce general concept of erosion using array grammars. The erosion is represented as a sequence of rewriting of the initial matrix (height field). The erosion is described as set of production rules. These production rules are applied to the square shaped part of matrices. This method is also advantageous, because it is inherently parallel.

2 Previous work

The previous work can be divided into two major groups. The algorithms belonging to the first group of the methods *generate* already eroded terrains [3, 4, 9, 11].

The second group describes erosions of *any* terrain [6], regardless if the terrain was obtained with some artificial technique or if measured data was used.

2.1 Traditional fractal terrain modeling techniques

Several techniques for fractal terrain synthesis exist. The original method for generating fractional Brownian surface (*fBm*) uses Poisson faulting [5]. This method involves application of Gaussian random displacements resulting in the statistically undistinguishable surfaces (planes or spheres). The main drawback to this approach is $\mathcal{O}(n^3)$ time complexity.

The random midpoint displacement method introduced by Fouriner *et al.* [3] represents a de facto standard method in fractal terrains generation. This well known recursive method is controlled by fractal dimension D of resulting fractal terrain.

Prusinkiewicz and Hammel [11] use the context sensitive rewriting process based on random midpoint displacement method on triangular grid. Their algorithm creates landscape and river simultaneously. This method creates one nonbranching river as result of the context sensitive rewriting system operating on geometric objects (in [11] on triangles).

The spectral synthesis method [8] (also called Fourier transform filtering) is another representative of these kinds of fractal modeling methods. The Fourier coefficients with spectral density proportional to $1/f^{\beta}$ are generated and inverse Fourier transform forms fractional Brownian motion surface. The exponent β controls the fractal dimension of the final object. Using only several first coefficients of Fourier spectrum causes generation of smoother surfaces than using coefficients corresponding to higher frequencies. A detailed description of the method can be found in [8].

Pickover [9] uses logistic function for generating "extraterrestrial terrains". This method has ability to create ridges, craters and streaks. Its advantage is that it can be used for tuning of already created terrain.

Kelley, Malin and Nielson [4] published algorithm consisting of two distinct steps. The first step represents the creation of a drainage network of rivers according to geologically known data and behavior. In the second step, the terrain is modeled as a set of surfaces under tension fitting previously generated data.

2.2 Terrain erosion models

Musgrave, Kolb and Mace [6] introduced techniques which are independent to the terrain creation algorithm and can be applied to already generated data represented as a for example height field. They introduce two methods: the *hydraulic erosion* and *thermal weathering*. Hydraulic erosion is caused by the presence of water (rain). The basic scheme of transport of the material is as follows: part of the material is dissolved in the water, the water runs down the hill and after its evaporation the material stands on the new place. This approach can be easily extended to any representation introduced in [2].

Thermal weathering is caused by temperature amplitude. Small elements of the material are crumbled and they pile up on the bottom of an incline. Both methods are controlled with several coefficients.

3 Erosion of terrain using array grammars

3.1 Formal means for the description of dynamic processes

In the previous part we described some methods for simulation of terrains erosion. These methods were mostly developed for this particular purpose.

However, some classes of general methods for description of such geometric operations exist. One of these methods is based on the theory of formal languages. The method defines a set of production rules by means of which the generative process will run. During this process some substrings are rewritten to other substrings. It would be possible to assign certain interpretation with geometric meaning. A good example of this kind are L-systems [10]. Here the dynamics of plant growing could be described in a formal way. The theory of L-systems has been extensively and intensively developed by now (*cf.* [7, 12, 13] for latest results).

A simple two-dimensional generalization of string rewriting systems is to extend grammars for one dimensional strings to two-dimensional arrays [1]. The primitives are the array elements and the relation between primitives is the two-dimensional concatenation. Each production rule rewrites one subarray by another, rather than one substring by another.

Array grammar is an ordered triple

$$G = \langle V, \omega, P \rangle \tag{1}$$

where:

- $V = \{\mathbf{M}\}$ denotes an alphabet consisting of matrices $\mathbf{M}$ of real numbers of type $N \times N$; $N > 0$,
- $\omega \in V$ is an *axiom* (starting symbol of rewriting) and finally
- P is a finite set of productions $P \subset V \times V$ of the form

$$\mathbf{A} \to \mathbf{B};\ \mathbf{A}, \mathbf{B} \in V. \tag{2}$$

The *production* (rewriting) ($\to$) is one application of a production rule $p \in P$ to an element $\mathbf{A} \in V$.

The *derivation* ($\Rightarrow$) $\mathbf{A}_0$ to $\mathbf{A}_n$ of length n is a sequence of n rewritings

$$\mathbf{A}_0 \to \mathbf{A}_1 \to \ldots \to \mathbf{A}_{n-1} \to \mathbf{A}_n,$$

where $\mathbf{A}_i \to \mathbf{A}_{i+1} \in P; i = 0, \ldots, n-1$. We write

$$\mathbf{A}_0 \Rightarrow \mathbf{A}_n. \tag{3}$$

In our interpretation a two-dimensional array can represent a segment of terrain that can be modified. The production rules should be constructed in such a way that their use will correspond to the terrain erosion process.

3.2 The production rules

The rewriting process (2) is context sensitive. We must construct the set of production rules P which will cause the physical simulation of erosion. These productions depend on the profile of the terrain. If we want to raise some terrain for example we must construct the productions which will curtail local disparity, and if we want to generate slope - we do not care about local saliencies - more wide-ranging production should be used.

3.2.1 Right side of the production.

Right side of the production (2) has the form

$$\mathbf{B} = \mathbf{X} * \mathbf{A}, \tag{4}$$

where $\mathbf{A}$ is the original matrix and $\mathbf{X}$ is a matrix of multipliers. Matrix $\mathbf{X}$ is of the same type $(N \times N)$ as matrix $\mathbf{A}$ and operator $*$ denotes that values b_{ij} of the matrix $\mathbf{B}$ are computed as

$$b_{ij} = \begin{cases} (1 + x_{ij})\ a_{ij}; & x_{ij} \leq 0 \\ a_{ij} + x_{ij}\ D; & x_{ij} > 0 \end{cases}$$

where

$$D = \sum (a_{ij} - (1 + x_{ij}) a_{ij}) = \sum -x_{ij} a_{ij}; \quad \forall x_{ij} \leq 0.$$

The number D is the total amount of material to be moved. This calculation causes no material to be added and no material to disappear from the tested matrix if

$$\sum x_{ij} = 1; \quad \forall x_{ij} > 0.$$

For example the matrix of multipliers $\mathbf{X}$ from equation (4) defined as

$$\begin{bmatrix} 0.125 & 0.125 & 0.125 \\ 0.125 & -0.2 & 0.125 \\ 0.125 & 0.125 & 0.125 \end{bmatrix}$$

caused material to be moved from the center to the sides.

3.2.2 Where the production can be applied.

The production according to the definition (2) should replace whole matrix $\mathbf{A}$. We use small matrices (4×4, or 5×5) and we divide the two dimensional projection of the height field into small squared subregions of the same type. This can be done in two ways in our algorithm as a *"chess-board" partitioning* or as a *floating array.*

In the first case the height field is divided into small regions as follows

$$\left[\begin{array}{cc|cc|cc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array}\right].$$

The left hand sides of productions are tested one by one with every subarray for matching. If it matches - the subarray is rewritten. One disadvantage to this method is that the size of $\mathbf{A}$ should be an integer divider of the size of the input height field, or the margin must be treated as a special case.

Second, and a more important disadvantage, is that this approach does not match the elements lying on the border of two regions.

The floating array approach uses the following scheme

$$\left[\begin{array}{|cc|cccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array}\right], \left[\begin{array}{c|cc|ccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array}\right], \ldots$$

This is implemented as follows: So called *shadow matrix* is initialized with zeros. After every step, the changes are summed in the shadow matrix. At the end these values are added to the height field.

Size of production matrix plays a significant role for the algorithm. Small ones have a better chance of matchings while large patterns can define more complicated appearance. The size of the matrix in the floating array approach influences the execution time of the simulation.

3.2.3 What the productions match.

We use the *range matching*; the rule matches if the values from the tested matrix belong to some range denoted by $[min, max]$. We consider three kinds of matching:

1. absolute matching,
2. matching with reference point and
3. matching with local ordering.

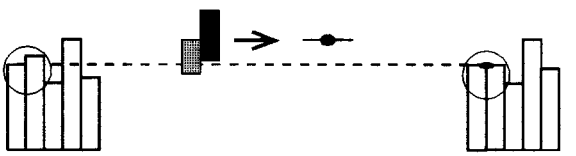

Figure 1: Left hand side of production (in the middle) matches one dimensional matrix (leftmost) only in one case, because absolute matching of z-coordinates was used.

In the case of absolute matching (see Figure 1) elements of the pattern array match if all of the tested elements belong to the ranges of the left side of production rules (2). The pattern of size 2×2 has form

$$\left[\begin{array}{cc} [a_{11}, b_{11}] & [a_{12}, b_{12}] \\ [a_{21}, b_{21}] & [a_{22}, b_{22}] \end{array}\right]$$

where $[a_{ij}, b_{ij}]$ are the ranges in which the tested value must lie within.

We can simulate for example erosion during very long time in one altitude (caused for example by river or by sea level) with this method (see Figure 2).

Matching with reference point (see Figure 3) takes one (*reference*) point r from the tested area and the differences in the z coordinate of the others are calculated. The rule matches if these differences from the point r lies within the ranges in the pattern matrix. The pattern of size 2×2 would have *e.g.* form

$$\left[\begin{array}{cc} r & [a_{12}, b_{12}] \\ [a_{21}, b_{21}] & [a_{22}, b_{22}] \end{array}\right],$$

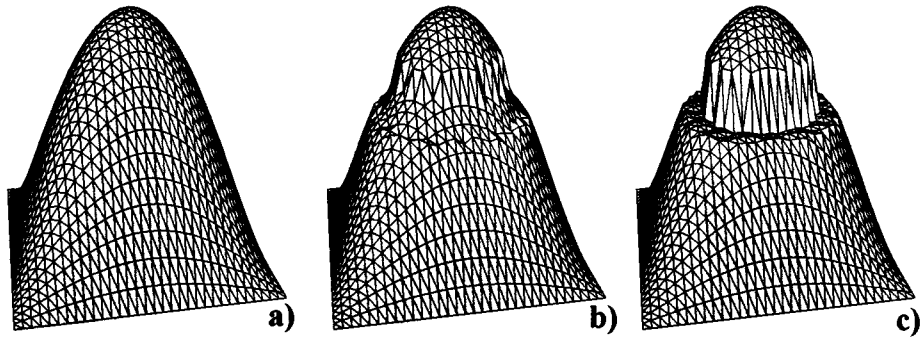

Figure 2: This erosion was generated by absolute matching floating array. Pattern matrix consists of the range $[[0.6, 0.8]]$ and maximum of altitude is 1.0. Matrix of multipliers $\mathbf{X}$ has the form $[-0.05]$. It was applied two times on the Figure b) and ten times on the c) to every elements. This production affects only the elements with absolute altitude within the interval [0.6 0.8].

where r is the reference point and $[a_{ij}, b_{ij}]$ are the ranges of differences. The position of the reference point r is not fixed in the matrix and can be arbitrary.

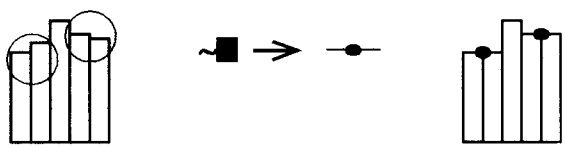

Figure 3: Production matches two times, because differences from the reference point are tested. Sign $\sim$ indicates that it is independent to the z-coordinate

The z-coordinate of the investigated profile can be arbitrary in this method, but the differences must fit the ranges in order to match (see Figure 3).

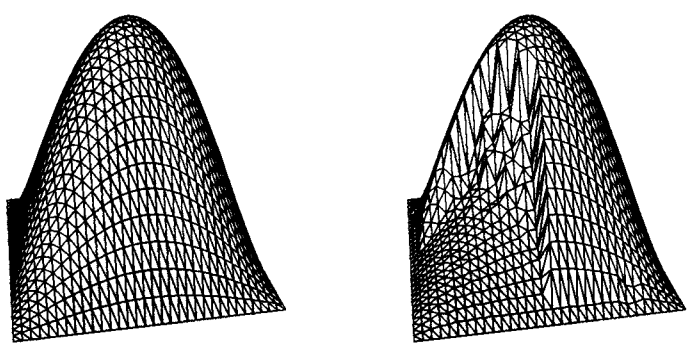

Figure 4: This erosion has been generated by matching reference point with floating array in ten iterations.

Using this kind of matching we can model erosion caused by the rain, where the differences are related to gradient (see Figure 4).

The rule used for the Figure 4 has the form (maximum of altitude is 1.0)

$$\begin{bmatrix} r & [0, 0.1] \\ [0, 0.1] & [0.05, 0.1] \end{bmatrix} \rightarrow \mathbf{B} \tag{5}$$

where **B** according to formula (4) and the matrix of multipliers has the form

$$\begin{bmatrix} -0.1 & -0.1 \\ -0.1 & -0.1 \end{bmatrix} \tag{6}$$

This production is gradient dependent (as can be seen on Figure 4); it erodes only some part of the terrain depending on the slope (it does not erode the top of the mountain).

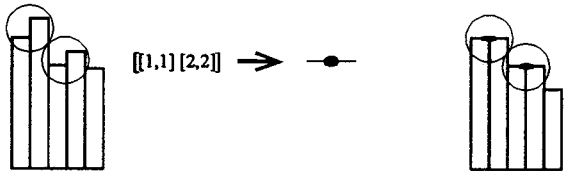

Figure 5: The order matches two times regardless of the differences and of the absolute z coordinates. It means the second element $[2,2]$ has to be greater then the first one $[1,1]$.

Matching with local ordering works as an order statistic. It sorts the height field (or its tested subsquare) out of z coordinate and assigns the position number to every element (see Figure 5). Left side of the production in this method has *e.g* form

$$\begin{bmatrix} [a_{11}, b_{11}] & [a_{12}, b_{12}] \\ [a_{21}, b_{21}] & [a_{22}, b_{22}] \end{bmatrix}$$

where a_{ij} and b_{ij} are ranges of the *positions* in the sorted rank.

With this kind of matching we can erode some shapes regardless of their gradient (see Figures 6 and 4). This production is independent on the relative differences and depends on the positions in the rank.

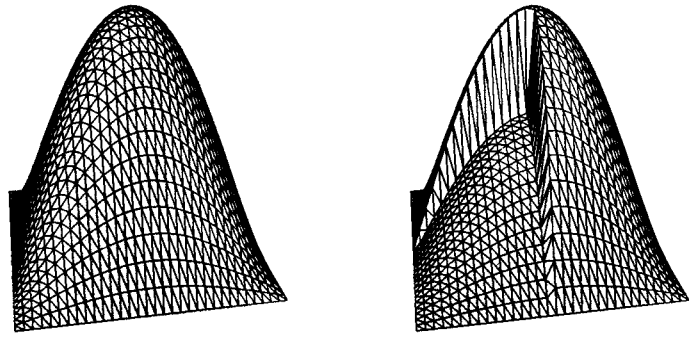

Figure 6: This erosion has been generated with local ordering with floating array in 10 iterations.

Pattern rule from the erosion in Figure 6 has form

$$\begin{bmatrix} [1,3] & [1,3] \\ [1,3] & [4,4] \end{bmatrix} \rightarrow \mathbf{B} \tag{7}$$

where **B** according to (4) and the matrix of multipliers has the same form as in previous example (see (6)).

$$\begin{bmatrix} -0.1 & -0.1 \\ -0.1 & -0.1 \end{bmatrix} \tag{8}$$

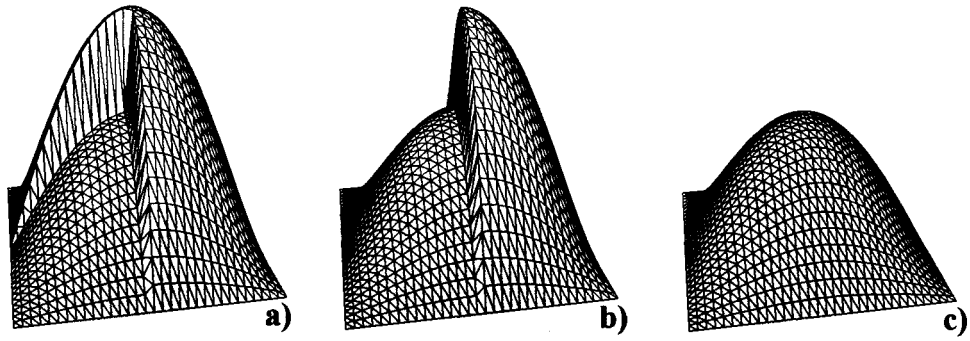

Figure 7: Erosion with rotation of pattern matrix. It was applied once a), twice b) and four times c).

A matrix which is not symmetrical to its center causes erosion in certain direction (see Figure 7). If we want to erode in a different direction using the same production, we can rotate the pattern matrix and the matrix on the right side of the rule. Figure 7 demonstrates application of one production rule without its rotation - in the second case it was rotated by 90°, and in the last one it was rotated three times.

4 Implementation and results

The program is very easy to implement, we use ANSI C and our program works under MS-DOS and UNIX (we have tested Solaris, IRIX and Linux).

We have used data generated by the random midpoint displacement method as well as the data generated by spectral synthesis.

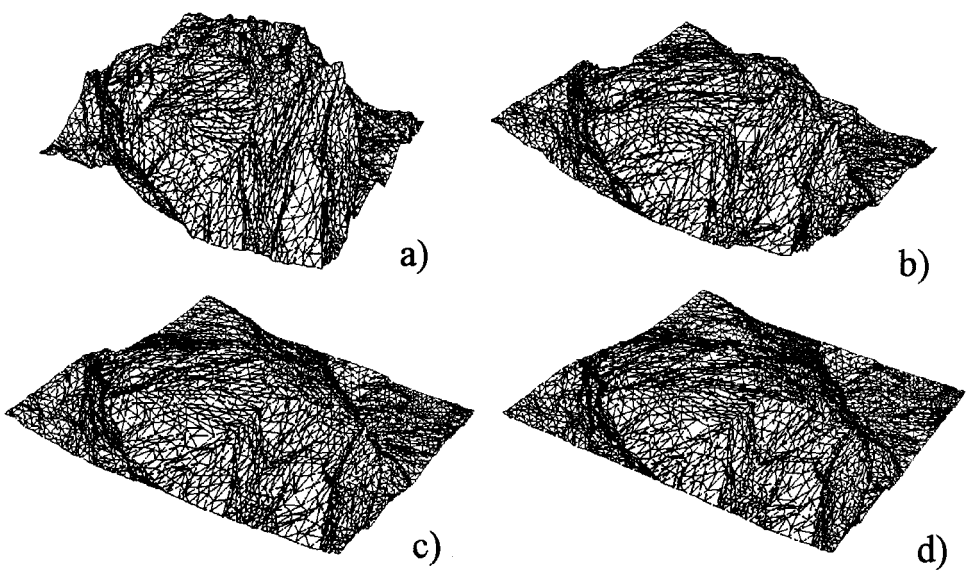

Figure 8: This erosion has been generated with local ordering with floating array and with rotation.Height field resolution is 64 × 64. Figure a) represents uneroded terrain. On Figure b) c) d) there are some terrains after 200, 400 and 600 iterations.

Figure 8 consists of several frames from erosion of fractal terrain using local ordering with floating array with three rotations of the pattern matrix. The total time of compu-

tation on SGI Indigo^2 with R4400/200MHz in resolution 64×64 with number of iteration 600, was 10 minutes.

5 Conclusions

A new method for synthetical terrain erosion is presented here. This method is based on rewriting of matrices. We use three kind of rewriting. The absolute rewriting allows erosion of objects in predefined altitude, rewriting with the reference point erodes arbitrary object in any altitude and the last method – matching with local ordering – erodes some shapes in any scale.

This paper presents a new approach for erosion of synthetical terrains. The other methods describe only one kind of erosion. Our approach is general - we can simulate wide classes of erosion using one algorithm.

Future work includes parallel implementation of the algorithm as well as including stochastic rules.

We would like to thank our anonymous referees for helpful comments and suggestions.

References

[1] M. Dacey. The Syntax of a Triangle and some other Figures. *Pattern Recognition*, 2:11–30, 1970.

[2] A. Dixon and G. Kirby. A Data Structure for Artificial Terrain Generation. *Computer Graphics Forum*, 13:37–48, 1994.

[3] A. Fournier, D. Fussel, and L. Carpenter. Computer Rendering of Stochastic Models. *Communications of the ACM*, 25:371–384, 1982.

[4] A. Kelley, M. Malin, and G. Nielson. Terrain Simulation Using a Model of Stream Errosion. *Computer Graphics*, 22(4):263–268, 1988.

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

[6] F. Musgrave, C. Kolb, and R. Mace. The Synthesis and Rendering of Eroded Fractal Terrains. *Computer Graphics*, 23(3):11–1–11–9, 1989.

[7] R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of SIGGRAPH '96, in Computer Graphics, Annual Conference Series 1996*, volume 30(4), pages 397–410, 1996.

[8] H. Peintgen and D. Saupe. *The Science of Fractal Images.* Springer–Verlag, New York, 1988.

[9] C. Pickover. Generating Extraterrestial Terrain. *IEEE Computer Graphics and Applications*, 17:18–21, 1995.

[10] A. Prusinkiewicz, P.and Lindenmayer. *The Algorithmic Beauty of Plants.* Springer-Verlag, New York, 1990.

[11] P. Prusinkiewicz and M. Hammel. A Fractal Model of Mountains with Rives. In *Proceedings of Graphics Interface'93*, pages 174–180, 1993.

[12] P. Prusinkiewicz, M. James, R. Měch, and J. Hannan. The Artificial Life of Plants. In *Course Notes of SIGGRAPH '95, in Computer Graphics, Annual Conference Series 1995*, volume I, pages 1–38, 1995.

[13] P. Room and J. Hanan. Virtual plants: new perspectives for ecologists, pathologists and agricultural sciencists. *Trends in Plant Science*, 1:33–38, 1996.