ETL-0504

AD-A201 171

Computer Generation of Fractal Terrains

Eugene A. Margerum
Anne Werkheiser

September 1988

U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060-5546

E T L

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA201171

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ETL-0504	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION U.S. Army Engineer Topographic Laboratories 6b. OFFICE SYMBOL (If applicable) CEETL-RI-B	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Fort Belvoir, Virginia 22060-5546	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 6.11.01.A; PROJECT NO. 4A161101A91; TASK NO. D; WORK UNIT ACCESSION NO. 01

11. TITLE (Include Security Classification)
Computer Generation of Fractal Terrains

12. PERSONAL AUTHOR(S)
Eugene A. Margerum and Anne Werkheiser

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Research Note	FROM _____ TO _____	1988, September	30

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Fractals, LISP, Terrain, Artificial Landscapes, Simulation, Computer Graphics.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The use of fractals for the generation of artificial terrains is presented. An introduction to the introduction to the relevant basic properties of fractals is given and a method for the generation of artificial fractal landscapes is described. The algorithm has been used to develop a LISP computer program for synthesizing topographic surfaces. Examples of the resulting structures are given in the form of a series of profiled surfaces representing landscapes of varying fractal dimension and varying vertical dilation. The LISP computer programs are also given and described.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL E. James Books	22b. TELEPHONE (Include Area Code) (202) 355-2774 22c. OFFICE SYMBOL CEETL-IM-T

DD Form 1473, JUN 86 *Previous editions are obsolete*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

PREFACE

The work described in this report was part of an ILIR project (work unit 4A161101A91D01) conducted in the Center for Artificial Intelligence of the Research Institute, USAETL. Its purpose was to study the general properties of fractals and to determine some of their potential applications to mapping and the topographic sciences. Particular goals included the development of rudimentary computer programs for generating synthetic terrain and the elucidation of "fractal dimension" in terms of surface roughness and natural textures.

During the period of the investigation, COL Alan L. Laubscher, EN, was the Commander and Director, Mr. Walter E. Boge was the Technical Director and Dr. Robert D. Leighty was the Director of the Research Institute.

The work was performed in the Center for Artificial Intelligence and the report was prepared under the supervision of Lawrence A. Gambino, Director, Research Institute.

CONTENTS

ILLUSTRATIONS

COMPUTER GENERATION OF FRACTAL TERRAINS

INTRODUCTION

The concept and the word "FRACTAL" were introduced by Benoit B. Mandelbrot around 1975. He made the important observation that many objects in the natural world are not continuous, smooth or measurable by the traditional standards of the applied mathematician but require the use of more general methods and constructs of pure mathematics. Furthermore, they are characterized by a quantity which, for any particular object, remains invariant regardless of the scale of observation. In the case of natural objects, this quantity, the 'fractal dimension', may retain its invariance over many orders of magnitude. The importance of the concept can be surmised by the fact that most physical theories, aside from fundamental laws, break down or lose their validity over a much smaller range due to inapplicability of the underlying assumptions.

Since their introduction, there has been a nearly explosive interest in fractals in physics, in the natural sciences, in computer graphics, and many other areas; as well as for their existence as objects that are beautiful and interesting in their own right. Several of the most well-known 'fractals' have a long existence in classical mathematics and have been historically important in the development of the theory of point sets, dimension theory, measure theory and topology. Continued developments in applications of fractals will probably result in the relocation of the superficial borderline between applied and pure mathematics deep into the territory of the latter.

It was recognized by Mandelbrot that such terrestrial features as coastlines and terrain were fractal. As such, one of the parameters used to describe a given instance is a 'dimension' that need not be (usually is not) a whole number. In the work described here, a computer algorithm (due to Fournier, Fussell and Carpenter) was used to generate fictitious computer representations of terrains. The display of the resulting data was in the form of profiles of the synthetic terrain surfaces. The terrain figures to be presented were made by directly photographing the video display of the computer terminal. No consideration was given to the generation of perspective views or to the problems of representing ("colored") surface reflectances under conditions simulating natural lighting. A Gaussian distribution was used to generate a series of surfaces for which the fractal dimension varied from 1.05 to 1.80. It was of interest to determine the parameters that would lead to realistic simulations of real terrains.

Another important application of fractals to the topographic sciences is the determination of the fractal dimension for given real natural objects and the use of the dimension as a descriptor or identifier. This problem is called the inverse problem and is not treated in the present report.

BASIC FRACTAL CONCEPTS

Fractals have been defined formally in the following way.

Definition. A fractal is a set for which the Hausdorff Besicovitch dimension strictly exceeds the topological dimension.

It is apparent that this definition contains two types of dimension.

The topological dimension D_T for a set will always be a whole number and corresponds with our intuitive concept of dimension. It is defined by assigning the dimension -1 to the empty set and using the definition below.

Definition. A space X has dimension $\leq D_T$ if every point p can be separated by a closed set of dimension $\leq D_T - 1$ from any closed set not containing P.

In this way, a set of isolated or disconnected points has $D_T = 0$ since the points are already disconnected by the null set. Similarly, ordinary (non-fractal) curves are found to have $D_T = 1$ since any point can be separated from the rest of the curve by removing points. And, for surfaces, $D_T = 2$, the segmentation is accomplished by removal of the points on a curve $D_T = 1$. When dealing with fractal sets, the same principle holds but much greater care must be taken because of the complexity of fractal sets and sometimes ingenious constructions may be required.

On the other hand, the Hausdorff Besicovitch dimension D is a measure theoretic concept and, depending upon the set, its determination can be extremely difficult. The set of points, P, to be measured is covered with (closed) balls, B_i , of maximum radius ϵ such that

$$P \subset \bigcup_{i=0}^{n(\epsilon)} B_i(\epsilon)$$

The measure of a k-dimensional ball of radius R is given by

$$M_k(R) = \alpha(k)\, R^k$$

and where α is given in terms of the Γ function.

$$\alpha(k) = \frac{[\Gamma(\frac{1}{2})]^k}{\Gamma(1+\frac{k}{2})}$$

The Hausdorff measure, of dimension k, of the set P is then given by

$$H^k(P) = \lim_{\epsilon \to 0^+} \inf \left\{ \alpha(k) \sum_{i=0}^{n(\epsilon)} [r_i(\epsilon)]^k \right\}$$

and where the lim inf is to be taken over all possible coverings. For ordinary (non-fractal) sets, k is taken as the dimension D_T and the Hausdorff measure for any reasonable case turns out to be the corresponding count, length, area, or volume. For fractal sets, k need not be an integer and there will be one particular value, $k=D$, for which the Hausdorff measure is positive and finite.

$$0 < H^D < \infty$$

This dimension, D, is called the Hausdorff Besicovitch dimension and is the most fundamental parameter defining the fractal. The measure itself is usually of little importance since it is difficult to give it a physical meaning for non-integral D. For $k > D$ we will have $H^k(P) = 0$ and for $k < D$ the corresponding measure is infinite $H^k(P) = \infty$. It is well known [Hurewicz & Wallman] that the Hausdorff Besicovitch dimension is never less than the topological dimension. However, there are fractals with dimension given by an integer.

Because of the complexity of the above concepts, a more appropriate working definition has been adopted for the present study.

> **Definition.** Fractals are mathematical constructs that give realistic representations of natural patterns and textures and that exhibit invariant properties under scale change (magnification).

For geometrically regular fractals, enlargement by a certain factor will bring a subset of the fractal set back into coincidence and this accounts for the striking appearance of many fractals. For the surfaces to be generated here, the fractal should exhibit the same statistical properties under enlargement. This is insured by the generating algorithm.

Equally important in this study is the question of whether the fractal has an appearance similar to that of a natural terrain. Specifically, what dimensions and other generating parameters lead to realistic results.

THE FRACTAL GENERATION ALGORITHM

The method to be described for generation of fractal surfaces was originated by Fournier, Fussell and Carpenter[1] and is usually known as the FFC algorithm, although this designation may confuse it with a different algorithm also given by them. This algorithm was chosen because of the simplicity of the computation, its flexibility and adaptability to potential applications in the topographic community, its realism, and its general compatibility with "standard" methods of computer graphics. These criteria are discussed in the next paragraph.

The FFC algorithm to be described allows for local computation of terrain elevations by densification of points from a set of elevations given over a square grid. The computation requires only the averaging of elevations at the corners of a square and the addition of a scaled random number (according to a fractal generating rule). This permits filling of fictitious terrain textures in cases where actual topographic data may be given on a sparse (coarse) grid. It also permits the filling of more detail in regions of higher magnification, such as in the foreground, or in cases of "zooming-in." The method also allows for variation of fractal dimension, vertical dilation, or the distribution function from one local area to another. Over square regions where the grid size does not vary, only elevations need be stored since horizontal coordinates are easily generated from grid indices and from the grid spacing and origin.

Consider the situation shown in figure 1a. The grid lies in a horizontal plane and elevations are known for the points given by the intersections of the grid lines (including those on the bounding sides). Elevations are to be generated over the center point of each square as depicted here by the dots.

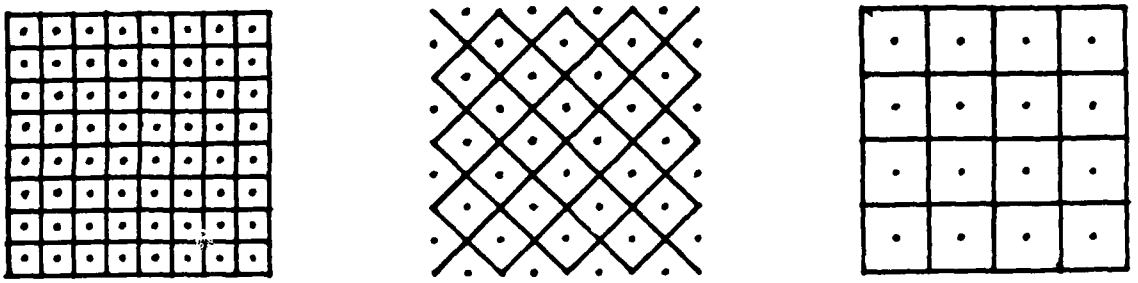

FIGURE 1. FRACTAL FILLING OF THE TERRAIN GRID.

[1] Alain Fournier, Don Fussell, and Loren Carpenter, "Computer Rendering of Stochastic Models," Communication of the ACM, vol 25, no. 6, June 1982, pp. 371-384

For each small square cell, the average elevation E_{av} of the corner (vertex) points is taken. A pseudo-random number chosen from a Gaussian distribution R is used to generate the new elevation, E_{new}, the center point by

$$E_{new} = E_{av} + R\ \delta^{3-D}$$

where δ is the distance from the center point to the vertex. Here D is the fractal dimension and 3 is the dimensionality of the embedding space.

After this has been repeated for each cell, the elevations will be known over a finer diagonal grid containing both the original elevations and the newly generated ones as shown in figure 1b. The procedure is repeated again for elevations of the center points of this grid with the exception that for outer bounding edges, the average cell elevation is determined from only three points.

At this stage a new (horizontal-vertical) grid has been established as shown in figure 1c. It will contain four times as many points as the initial grid in figure 1a. The complete procedure can then be repeated again and again until a sufficiently dense set of elevations has been computed.

After the elevations have been computed, a vertical dilation is applied to adjust the amount of vertical exaggeration. It was found that surfaces of larger fractal dimension required much less shrinkage in the vertical direction than those with fractal dimension near 2.

EXAMPLES OF FRACTAL TERRAINS

Examples of computer synthesized terrain are given in appendix A. They include results for various dimension and vertical exaggeration and should serve to demonstrate the effects of varying these parameters. The distribution that was used in all cases was an approximated cumulative distribution function for a Gaussian. The views of the surface displayed on the screen are not in perspective, but are drawn in the same orthographic way that three-dimensional mathematical objects are drawn (e.g. on a blackboard). The three-dimensional appearance is partly an illusion since only profiles of the surface are plotted. Nevertheless, since most people are accustomed to such plots, they can give an excellent feel for the roughness, texture, and other qualities of the surfaces involved.

The effect of varying the amount of vertical exaggeration is shown in terrains 1,2, and 3. These three examples all have fractal dimension D= 2.05 and were produced by the same set of random numbers. They were produced using vertical dilation factors of 0.05, 0.1, and 0.2, respectively.

The computer renderings in terrains 4, 5, 6, 7 and 8 show the effect of varying the fractal dimension. The dilation factor was also different for each example. For these surfaces, the fractal dimension D took the values 2.05, 2.1, 2.2, 2.4 and 2.8, respectively, while the amount of vertical exaggeration for each case, in turn, was 0.05, 0.1, 0.2, 0.4 and 0.8. The difference in these two parameters is clearly seen by comparing terrains 4 and 8. The former is much smoother than the latter, which shows roughness much greater than any imaginable real world terrain, while the total vertical exaggeration is not drastically different. Terrains 5 and 6 are more rugged than 4 and show greater vertical differences than either 4 or 8, while remaining much smoother than terrain 8.

An example of an undesirable result is shown in terrain 9. The unnatural spikes on the back edge are believed to result from the chance occurrence of choosing numbers far out on the tails of the generating Gaussian. This can be eliminated by using a different distribution that does not possess such long tails; for example, a chopped off Gaussian. The rarity of such occurrences may be surmised by their absence from the other renderings since each one represents 263,169 generated elevations on a 513 x 513 grid with each 8th profile plotted.

THE LISP COMPUTER PROGRAMS

A listing of the computer programs is given in Appendix B. They were written in the LISP language for compatibility with other work in the Center for Artificial Intelligence, ETL, and because of the available hardware. A brief description of the principal functions will be given here.

Before running a program to generate a fractal surface, the function **foooz** must be executed to initialize the program and to define some of the arrays needed. The array **nordis** contains a numerical table representing the cumulative distribution function for a Gaussian. The function **u-to-g** converts a random number from a uniform distribution to a random number chosen from a Gaussian distribution.

The fractal generated is three dimensional, but the indices of a two- dimensional array are used for horizontal coordinates while the stored numbers give the vertical coordinate. The functions **ssu** and **ssv** generate screen coordinates from given three-dimensional coordinates. They do not give perspective views but only a convenient readily visualized representation. They are used by **dsl** to draw a straight line given its three- dimensional endpoints.

The function **elev** generates a fractal elevation in the center of a cell in the fractal grid. Its arguments are: the average elevation of the corner points, the fractal dimension, and the distance from the center of the cell to a corner.

The function **surfract** generates the fractal surface by using the previously described filling algorithm. Its sole argument is the fractal dimension. Some intermediate views are displayed on the screen to monitor the filling operation.

The function **dilate** is used to adjust the amount of vertical exaggeration by applying a dilatation factor. The arguments of this function are the scaling factor and another number specifying how many profiles should be skipped before plotting the next one. It is interesting to note that a negative scaling factor has the effect of inverting the surface, i.e. exchanging mountain peaks and valleys.

The function **stratum** is used to put a base on the surface and to give it the appearance of a solid slab. Its first two arguments specify the thickness of the slab and the distance between profiles, respectively.

Typical running times for a 513 x 513 surface generation and display have been a little less than 10 minutes.

CONCLUSION

Computer programs were written in LISP that make use of a Fournier, Fussell, Carpenter algorithm to generate synthetic fractal terrains having various dimensions. It was found that the degree of realism was best for fractal surfaces with dimension closer to 2 and that those with dimension closer to 3 were much too rough for any imaginable real world terrains.

It is believed that this algorithm can be used for filling scenes where actual elevations are given on a sparse grid in order to achieve realistic rendering of the terrain. To accomplish this will require the generation of perspective views where color or gray levels are used to depict surface reflectances under simulated real world illumination conditions. Such renderings would involve considerably more research and programming.

BIBLIOGRAPHY

Falconer, K.J. *The Geometry of Fractal Sets.* Cambridge University Press, 1985.

Federer, Herbert. *Geometric Measure Theory.* Springer Verlag, 1969.

Fournier, Alain, Don Fussell, and Loren Carpenter. *Computer Rendering of Stochastic Models.* Communications of the ACM, Vol. 25, No. 6, June 1982, pp. 371-384.

Hurewicz, Witold and Henry Wallman. *Dimension Theory.* Princeton University Press, 1941.

Mandelbrot, Benoit B. *The Fractal Geometry of Nature.* W. H. Freeman and Company, 1983.

Pentland, Alex P. *Fractal-Based Description of Natural Scenes.* SRI Technical Note No. 280, March 20, 1984.

APPENDIX A

FRACTAL TERRAINS

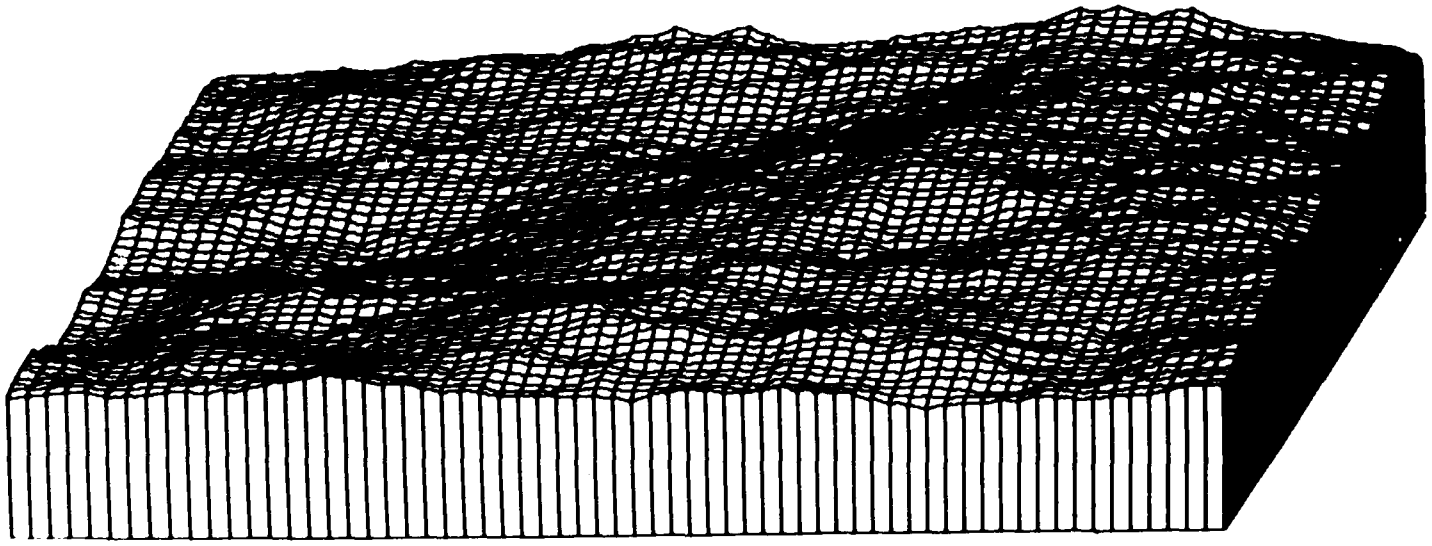

Figure A1. Terrain 1, Fractal Dimension 2.05.

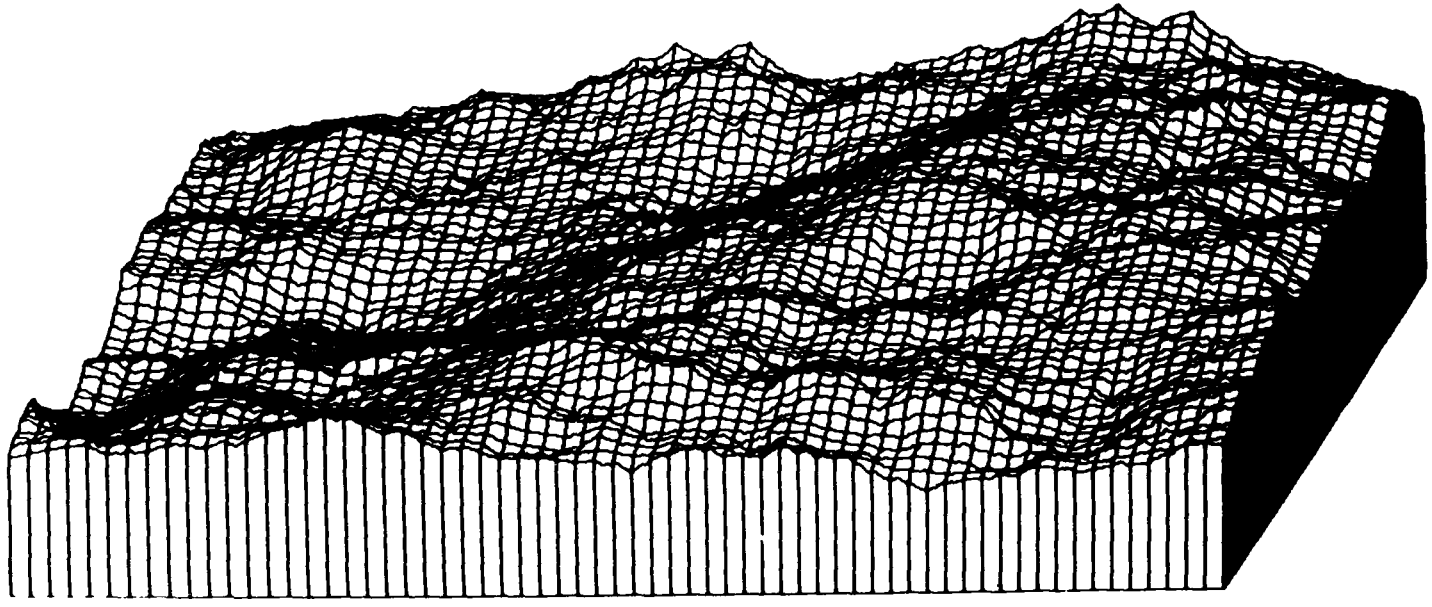

Figure A2. Terrain 2, Fractal Dimension 2.05.

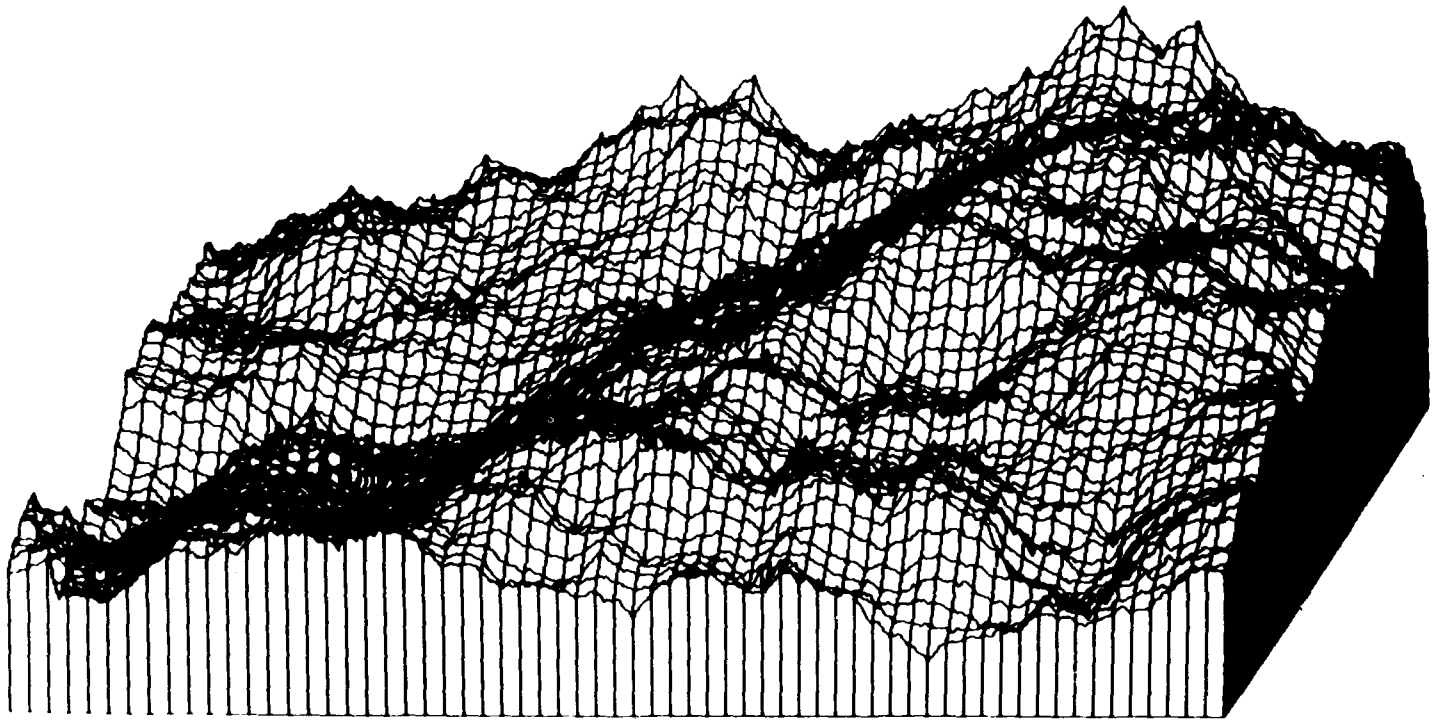

Figure A3. Terrain 3, Fractal Dimension 2.05.

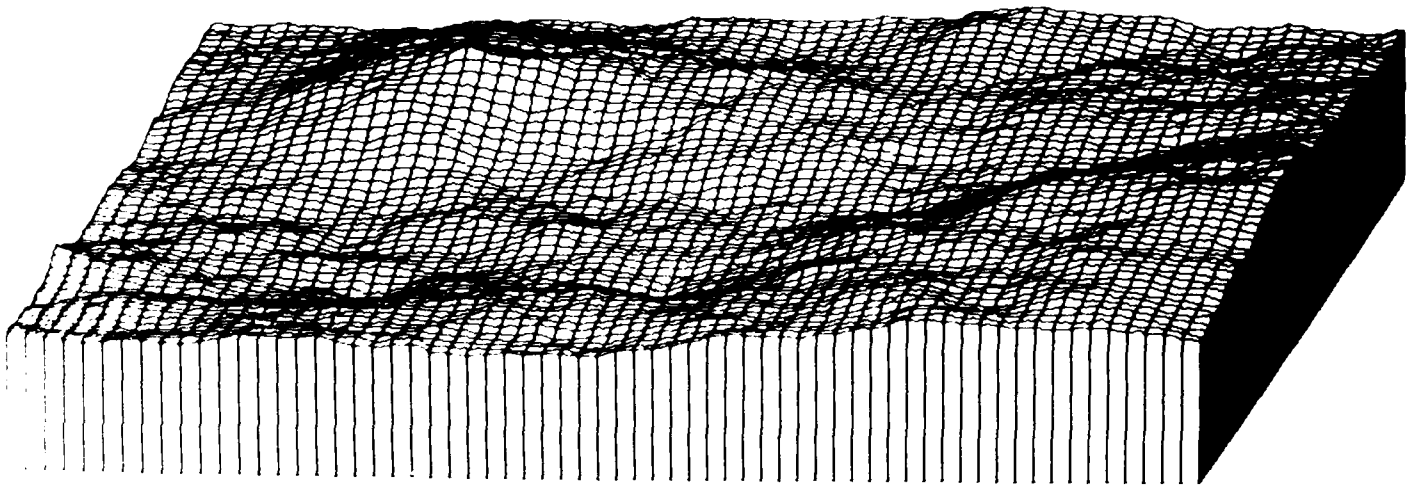

Figure A4. Terrain 4, Fractal Dimension 2.05.

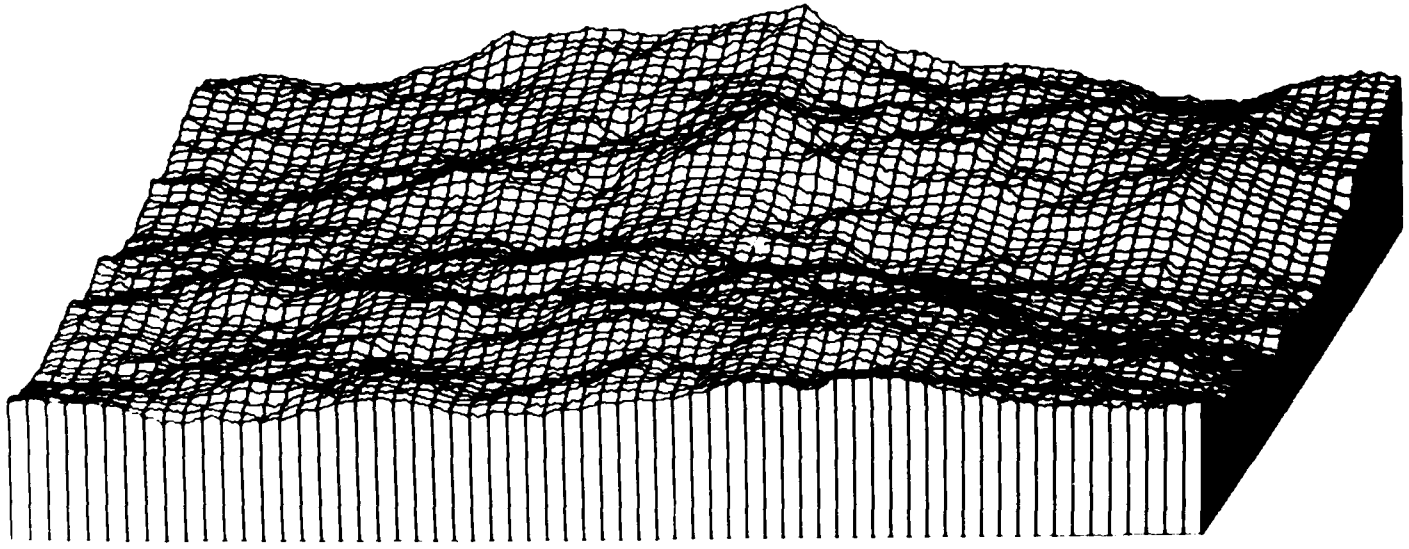

Figure A5. Terrain 5, Fractal Dimension 2.1.

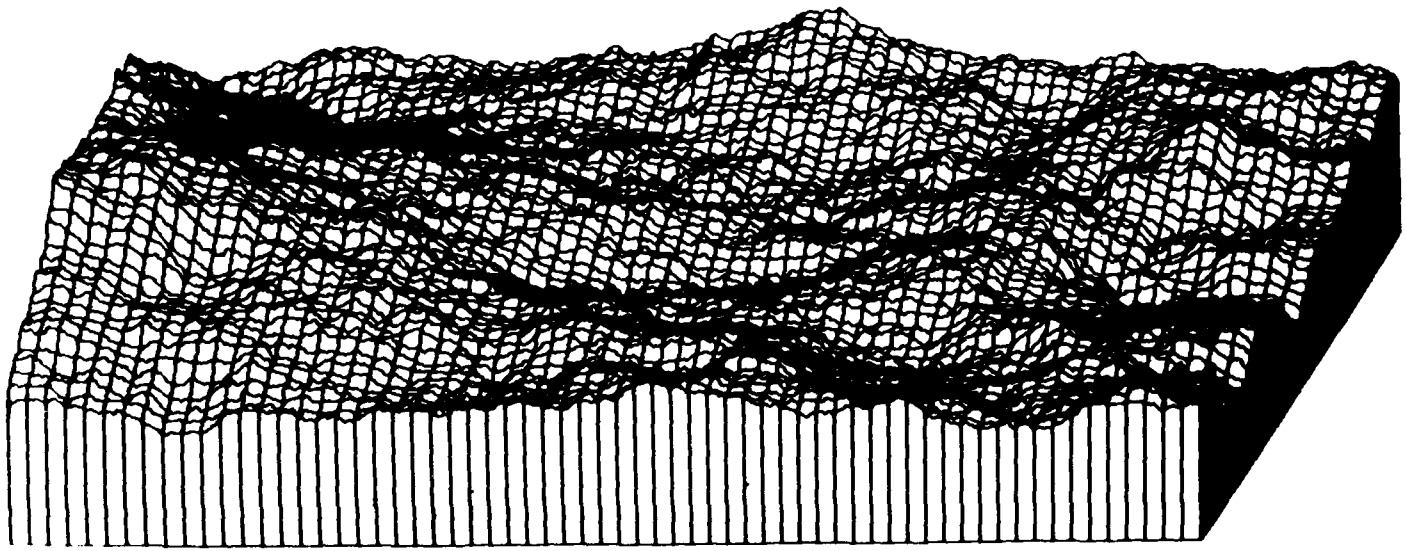

Figure A6. Terain 6, Fractal Dimension 2.2.

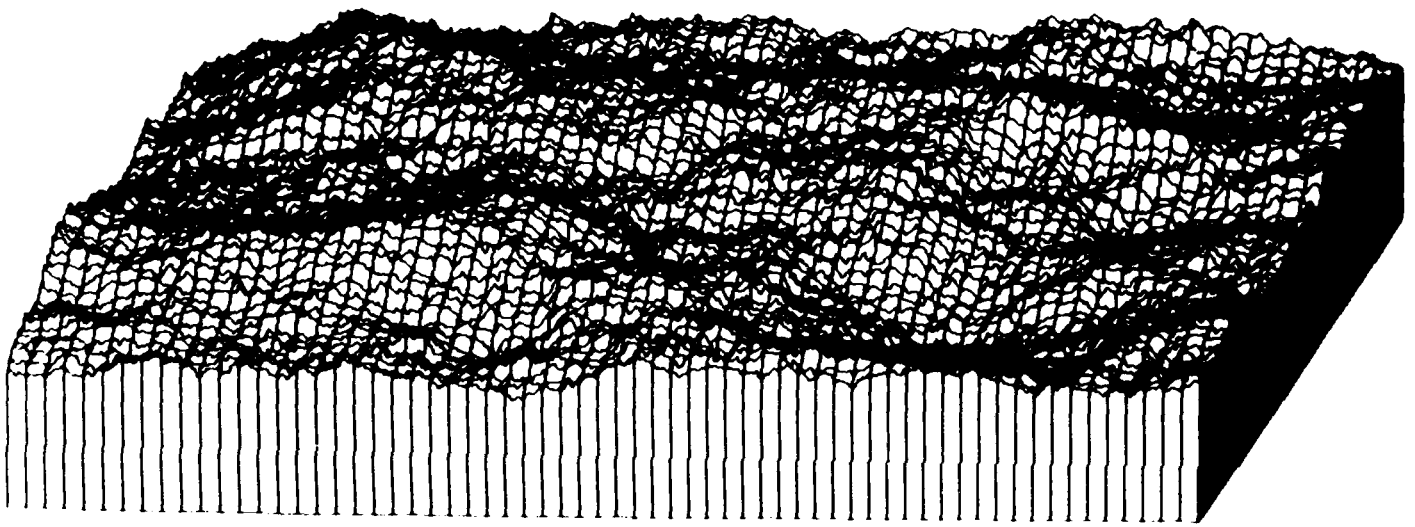

Figure A7. Terrain 7, Fractal Dimension 2.4.

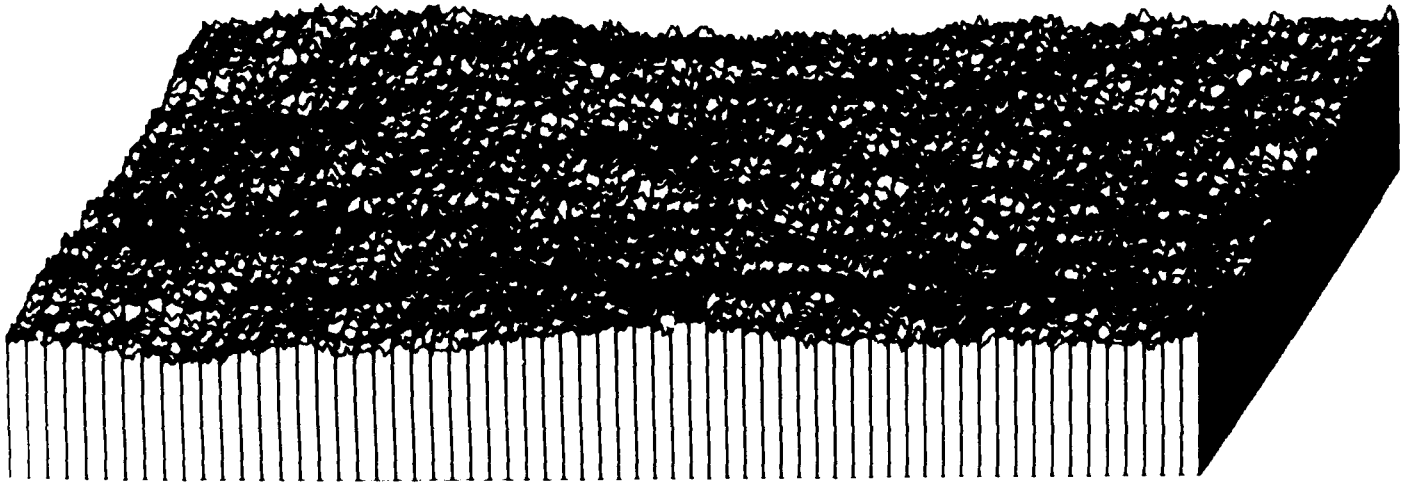

Figure A8. Terrain 8, Fractal Dimension 2.8.

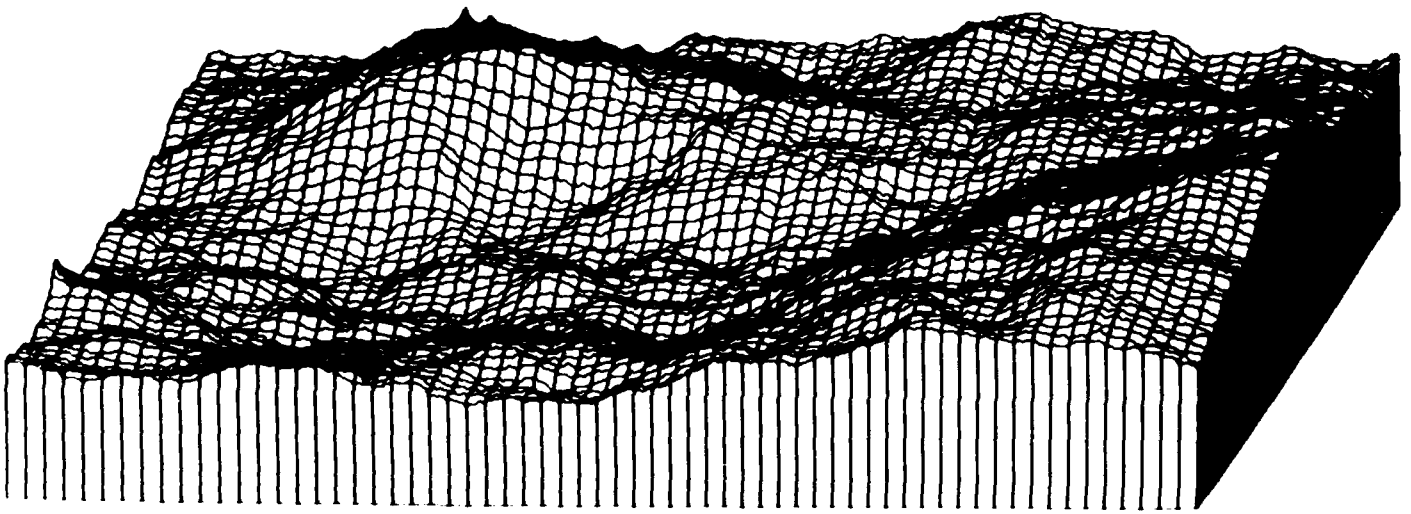

Figure A9. Terrain 9, Fractal Dimension 2.05.

APPENDIX B

THE LISP COMPUTER PROGRAMS

```
(defun foooz ()
  (set 'base 10.)
  (set 'ibase 10.)
  (set 'surf (make-array '(513 513) ':type 'art-4 ':initial-value 0.0))
  (set 'smurf (make-array '(513 513) ':type 'art-q ':initial-value 0.0))
  (set 'grating (make-array '(40 40) ':type 'art-q ':initial-value 0.0)))

(set 'nordis (make-array '(61 2) ':type 'art-q))

(fillarray nordis '(0.00 0.500 0.05 0.523 0.10 0.540 0.15 0.560 0.20 0.579
     0.25 0.597 0.30 0.618 0.35 0.637 0.40 0.655 0.45 0.674 0.50 0.691 0.55
     0.709 0.60 0.726 0.65 0.742 0.70 0.759 0.75 0.773 0.80 0.788 0.85 0.802
     0.90 0.816 0.95 0.829 1.00 0.841 1.05 0.853 1.10 0.864 1.15 0.875 1.20
     0.885 1.25 0.894 1.282 0.900 1.30 0.903 1.35 0.911 1.40 0.919 1.45
     0.926 1.50 0.933 1.55 0.939 1.60 0.945 1.645 0.950 1.65 0.951 1.70
     0.955 1.75 0.960 1.80 0.964 1.85 0.968 1.90 0.971 1.95 0.974 1.960
     0.975 2.00 0.977 2.05 0.980 2.10 0.982 2.15 0.984 2.20 0.986 2.25 0.988
     2.30 0.989 2.326 0.990 2.35 0.991 2.40 0.992 2.45 0.993 2.50 0.994
     2.576 0.995 2.65 0.996 2.75 0.997 2.90 0.998 3.00 0.999 3.80 1.100))


(defun u-to-g(x)
     (cond ((lessp x 0.5)
            (set 's -1.0)
            (set 'x (- 1.0 x)))
           (t (set 's +1.0)))
     (loop for i from 0 until
          () (aref nordis i 1) x)
          do
          (set 'j i))
     (cond ((< j 59)
            (cond ((>= (- (aref nordis (+ j 1) 1) x)
                       (- x (aref nordis j 1)))
                   (* s (aref nordis j 0)))
                  (t (* s (aref nordis (+ j 1) 0)))))
           (t (* s (aref nordis j 0)))))


(defun ssu (x y z)
  (+ 50. (+ (* 1.5 x) (* .25 y))))


(defun ssv (x y z tilt)
  (- 600. (+ tilt (+ (* .25 y) (* 1.5 z)))))


(defun dsl (xa ya za xb yb zb)
  (send terminal-io ':draw-line (fix (ssu xa ya za)) (fix (ssv xa ya za 1.5))
        (fix (ssu xb yb zb)) (fix (ssv xb yb zb 1.5))))


(defun elev (ave dim rr)
  (+ ave (* (u-to-g (// (float (random 10001)) 10000.))
            (^ rr (- 3.0 dim)))))
```

```
(defun surfract (dim)
  (set 'surf (make-array '(513 513) ':type 'art-4 ':initial-value 0.0))
  (set 'edge 512)
  (set 'dist (// edge 2))
  (loop for n from 1 until (< dist 1)
        do
        (set 'rr (// (float edge) (sqrt 2.0)))
        (loop for i from dist by edge until (> i 512)
              do
              (loop for j from dist by edge until (> j 512)
                    do
                    (aset (elev (// (+ (aref surf (- i dist) (- j dist))
                                       (aref surf (+ i dist) (- j dist))
                                       (aref surf (- i dist) (+ j dist))
                                       (aref surf (+ i dist) (+ j dist))) 4.0)
                                dim rr)
                          surf i j)))
        (loop for i from 0 by dist until (> i 512)
              do
              (loop for j from (if (oddp (// i dist)) 0 dist) by edge
                    until (> j 512)
                    do
                    (cond ((equal i 0)
                           (aset (elev (// (+ (aref surf (+ i dist) j)
                                              (aref surf i (+ j dist))
                                              (aref surf i (- j dist))) 3.0)
                                       dim (float dist)) surf i j))
                          ((equal i 512)
                           (aset (elev (// (+ (aref surf (- i dist) j)
                                              (aref surf i (+ j dist))
                                              (aref surf i (- j dist))) 3.0)
                                       dim (float dist)) surf i j))
                          ((equal j 0)
                           (aset (elev (// (+ (aref surf (+ i dist) j)
                                              (aref surf (- i dist) j)
                                              (aref surf i (+ j dist))) 3.0)
                                       dim (float dist)) surf i j))
                          ((equal j 512)
                           (aset (elev (// (+ (aref surf (+ i dist) j)
                                              (aref surf (- i dist) j)
                                              (aref surf i (- j dist))) 3.0)
                                       dim (float dist)) surf i j))
                          (t
                            (aset (elev (// (+ (aref surf (+ i dist) j)
                                               (aref surf (- i dist) j)
                                               (aref surf i (+ j dist))
                                               (aref surf i (- j dist))) 4.0)
                                        dim (float dist)) surf i j)))))
        (set 'edge dist)
        (set 'dist (// dist 2))
        (turf 64 surf))
  (send terminal-io ':clear-window)
  (turf 64 surf))

(defun turf (grate smurf)
  (loop for i 'from 0 to 512 by grate
```

```
        do
        (loop for j from 1 to 512
              do
              (dsl (float i) (- (float j) 1.0) (aref smurf i (- j 1))
                   (float i) (float j) (aref smurf i j))))
  (loop for j from 0 to 512 by grate
        do
        (loop for i from 1 to 512
              do
              (dsl (- (float i) 1.0) (float j) (aref smurf (- i 1) j)
                   (float i) (float j) (aref smurf i j)))))


(defun dilate (scale del)
  (loop for i from 0 until (> i 512)
        do
        (loop for j from 0 until (> j 512)
              do
              (aset (* scale (aref surf i j)) smurf i j)))
  (turf del smurf))


(defun sift (scale del)
  (dilate scale del)
  (loop for i from 0 until (> i 39)
        do
        (loop for j from 0 until (> j 39)
              do
              (aset (aref smurf (+ 6 (* 12 i)) (+ 5 (* 12 j))) grating i j))))


(defun stratum (thickness grate (&optional (smurf smurf)))
  (dsl 0.0 0.0 (- 0.0 thickness) 512.0 0.0 (- 0.0 thickness))
  (dsl 512.0 0.0 (- 0.0 thickness) 512.0 512.0 (- 0.0 thickness))
  (loop for i from 0 to 512 by grate
        do
        (dsl (float i) 0.0 (- 0.0 thickness) (float i) 0.0 (aref smurf i 0))
        (dsl 512.0 (float i) (- 0.0 thickness) 512.0 (float i) (aref smurf 512 i))))
```