Pergamon

Comput. & Graphics, Vol. 22, No. 2–3, pp. 301–305, 1998

PII: S0097-8493(98)00014-4
Printed in Great Britain
0097-8493/98 $19.00 + 0.00

Chaos & Graphics

A TUTORIAL AND RECIPE FOR MOVING FRACTAL TREES

JOSE ANGEL GONZALEZ RODRIGUEZ

Capuchinos, 9-6°C, 31500 Tudela (Navarra), Spain *e-mail:* jgonzalezr@nexo.es

Abstract—This article provides computational recipes for building moving fractal trees. The resultant complex structures will interest both educators and students. The pseudocodes are included to encourage reader experimentation. © 1998 Elsevier Science Ltd. All rights reserved.

Key words: moving fractal tree, animated fractal tree.

1. MOVING FRACTALS

In this tutorial and artistic statement, the author describe tree-like objects based in a figure discovered when he was a child. In particular, he calls these 'moving fractal trees' with which students can spend numerous hours experimenting...The two-dimensional version (shown on Fig. 1) is a fractal whose infinite branches can move in three-dimensional space in a way such that no branch can touch any other [1].

This fractal is very easy to program. First, one draws a central branch of length L (the longest one), then one draws two smaller perpendicular branches of length K_1L, then four of length K_1K_2L, then eight of length $K_1K_2K_1L$, 16 of length $K_1K_2K_1K_2L$...and so on [2].

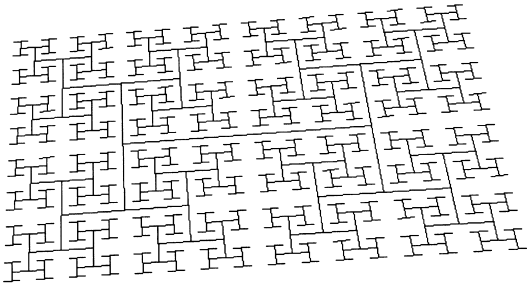

Fig. 1. Two-dimensional "static" tree.

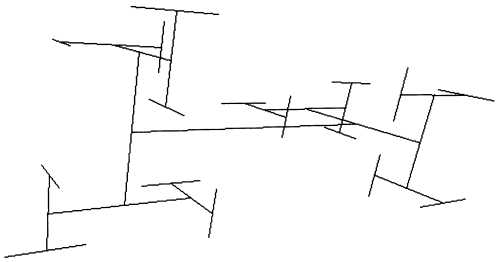

Fig. 2. Tree with opposite rotations.

If we make the product of $K_1K_2 = 1/2$, the Hausdorff Besikovitch dimension [3] of the fractal will be 2, and the tree will fill a rectangle of dimen-

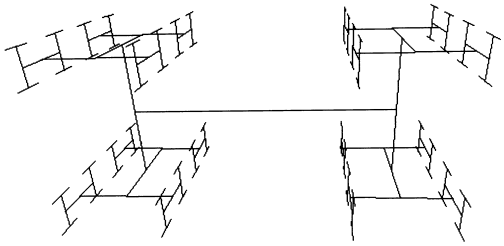

Fig. 3. Another two-dimensional rotated tree.

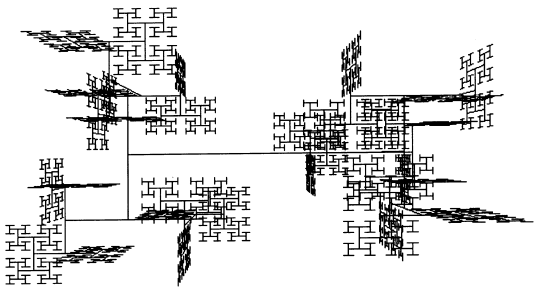

Fig. 4. From the fifth iteration, the branches do not rotate.

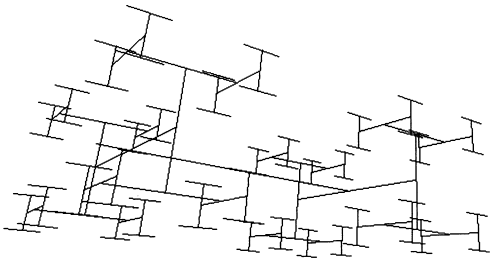

Fig. 5. Two-dimensional tree rotated in a similar way to the three-dimensional static tree shown on Fig. 7. This one would not fill space because its Hausdorff Besikovitch dimension is two instead of three.

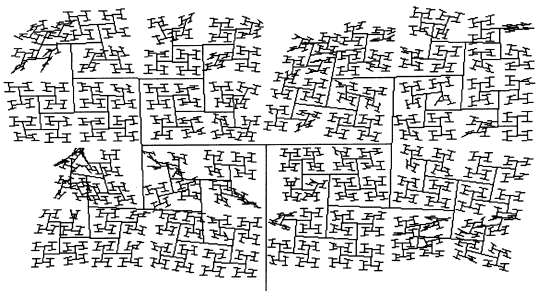

Fig. 6. Tree under random rotations.

sions $2L$ by $2K_1L$ without two branches ever touching. Moreover, its branches can rotate as shown on Figures 2, 3, 4, 5 and 6, without touching.

If we make $K_1 = 1$ and $K_2 = 1/2$ we will get a filled square of $2L$ by $2L$ after an infinite number of iterations.

2. PSEUDOCODE TO CREATE THE TREE

Pseudocode 1 in Scheme 1 outlines the steps necessary to create the (for the present) 'static' tree structure. The algorithm to draw it is in Pseudocode 2 (Scheme 2).

The tree goes from $(-S, -S)$ to (S, S). The appropriate value for 'S' should be given in the first line. The center of the screen should be at (0, 0).

Niter is the number of iterations. The variable $T_{\text{Points}} = 2^{Niter}$ holds the total number of points in the tree. The variable: $a(T_{\text{Points}}, 2)$ is dimensioned to hold the three-dimensional coordinates of every point in the tree.

2.1. *Center of gravity*

Figure 6 is shaped through random rotations. No matter how deeply disordered is the tree, its Center of Gravity (C. G.) remains always in the middle of the longest branch. It is easily demonstrable:

If n is the number of iterations (10 in Fig. 6), then the 2^n smallest branches can be removed without changing the C. G. of the whole tree, because

Pseudocode 1

```
S = 100
Niter = 9
TPoints = 2^Niter
Length = S / 2
Stp = TPoints / 4
K = 1 / SQR(2)
DIM a(TPoints, 2)

FOR i = 1 TO Niter - 1
  FOR x = Stp TO TPoints STEP Stp * 4
    IF i MOD 2 = 0 THEN
      a(x, 0) = a(x + Stp, 0)
      a(x, 1) = a(x + Stp, 1) - Length
      a(x + 2 * Stp, 0) = a(x + Stp, 0)
      a(x + 2 * Stp, 1) = a(x + Stp, 1) + Length
    ELSE
      a(x, 0) = a(x + Stp, 0) - Length
      a(x, 1) = a(x + Stp, 1)
      a(x + 2 * Stp, 0) = a(x + Stp, 0) + Length
      a(x + 2 * Stp, 1) = a(x + Stp, 1)
    END IF
    a(x, 2) = 0
    a(x + 2 * Stp, 2) = 0
  NEXT x
  Stp = Stp / 2
  Length = Length * K
NEXT i
```

Scheme 1. Pseudocode 1: code to create the two-dimensional tree. Programmed in MS QBasic, it is possible to add the statements: SCREEN 12 and WINDOW (–S, –S) – (S, S) after the first line of code to run it under this programming language.

Pseudocode 2

```
Stp = TPoints / 4
FOR i = 1 TO NIter - 1
  FOR x = Stp TO Tpoints STEP Stp * 4
    x0 = a(x, 0)
    y0 = a(x, 1)
    P2 = 2 * Stp
    x1 = a(x + P2, 0)
    y1 = a(x + P2, 1)
    LINE (x0, y0)-(x1, y1)
  NEXT x
  Stp = Stp / 2
NEXT i
```

Scheme 2. Pseudocode 2: code to draw the two-dimensional tree.

the C. G. of each small branch is at its own center (where it 'hangs' from its father branch). The 2^{n-1} branches can be recursively removed, then the 2^{n-2} and so on...until only the longest branch is left, whose C. G. is, of course, in the middle.

This property makes more feasible to build a big mobile tree whose branches could be spun by motors (or water if a fountain was made).

3. A POINT ROTATING AROUND AN AXIS

To program a tree with rotations it is necessary to know how to rotate an arbitrary point around an arbitrary axis in three-dimensional space. Let be (Ax, Ay, Az) and (Bx, By, Bz) two points that define an axis and (Px, Py, Pz) the point. $\mathbf{P}'$ is the point $\mathbf{P}$ after rotating it 'a' radians around the axis. Let n be the unit vector pointing from A to B:

$$(nx,ny,nz) = \frac{(Bx-Ax,By-Ay,Bz-Az)}{\sqrt{(Bx-Ax)^2+(By-Ay)^2+(Bz-Az)^2}}$$

Then (using three-dimensional vectors):

$$\begin{aligned}\boldsymbol{P}' = \boldsymbol{A} &+ \boldsymbol{n}[\boldsymbol{n}(\boldsymbol{P}-\boldsymbol{A})] \\ &+ \cos(a)[(\boldsymbol{P}-\boldsymbol{A}) - \boldsymbol{n}(\boldsymbol{P}-\boldsymbol{A})]] \\ &\pm \sin(a)\boldsymbol{n} \times (\boldsymbol{P}-\boldsymbol{A})\end{aligned}$$

Pseudocode 3 in Scheme 3 shows how this can be implemented. This pseudocode will generate random rotations of the whole tree as shown on Fig. 6. To make other types of rotation it can easily be modified, making angle 'Alfa' (see the code) fixed instead of random, for example.

The command 'RND' in the pseudocode returns a random value between zero and 1.

4. THREE-DIMENSIONAL TREE

The tree on Fig. 7 is a three-dimensional version of the tree on Fig. 1. To program it we follow a similar procedure.

First, one draws a central branch of length L, then one draws two smaller perpendicular branches of length K_1L, then four of length K_1K_2L (perpendicular now to the former three), then eight of length $K_1K_2K_3L$ (note we introduce K_3 here), 16 of length $K_1K_2K_3K_1L\ldots$ and so on.

If we make the product of $K_1K_2K_3 = 1/2$, the Hausdorff Besikovitch dimension of the fractal will be 3, and the tree will fill the space without two branches ever touching. Now, however, if the branches rotated they would collide.

If we make $K_1 = 1$, $K_2 = 1$ and $K_3 = 1/2$, we will get a filled cube of $2L$ by $2L$ by $2L$.

5. THE PROGRAM

The author has made an MSDOS program to display this group of figures; the current version is arbol209.zip, the full source code is included and can be downloaded from http://www.geocities.com/CapeCanaveral/3325. There are lots of options to deal with in the program, including a perspective toggle and two different screensavers.

6. CORRELATION OF BINARY NUMBERS WITH PATHS ON THE TREE

There is a correlation between pairs of numbers written in binary system and the paths to follow in the tree to reach the position indicated by the numbers. In Fig. 8, the longest line in the middle has a length = 1. Its middle point is (0, 0).

To reach any point on the plane, a continuous path on the tree can be followed.

Pseudocode 3

```
Stp = TPoints / 4
Pi = 3.14159
FOR i = 1 TO Niter - 1
  Counter = 0
  FOR x = Stp TO TPoints - 1 STEP Stp * 2
    P2 = 2 * Stp
    Counter = Counter + 1
    Ax = a(x, 0)
    Ay = a(x, 1)
    Az = a(x, 2)
    IF Counter MOD 2 = 1 THEN
      Bx = a(x + P2, 0)
      By = a(x + P2, 1)
      Bz = a(x + P2, 2)
    ELSE
      Bx = a(x - P2, 0)
      By = a(x - P2, 1)
      Bz = a(x - P2, 2)
    END IF
    Mn = SQR((Bx - Ax) ^ 2 + (By - Ay) ^ 2 + (Bz - Az) ^ 2)
    nx = (Bx - Ax) / Mn
    ny = (By - Ay) / Mn
    nz = (Bz - Az) / Mn

    Alfa = INT(RND * 8) * Pi / 256
    IF RND > .5 THEN Alfa = -Alfa
    ca = COS(Alfa)
    sa = SIN(Alfa)

    FOR j = x - Stp + 1 TO x + Stp - 1
      Px = a(j, 0) - Ax
      Py = a(j, 1) - Ay
      Pz = a(j, 2) - Az
      nPEscalar = nx * Px + ny * Py + nz * Pz
      nnPx = nPEscalar * nx
      nnPy = nPEscalar * ny
      nnPz = nPEscalar * nz
      cx = ca * (Px - nnPx)
      cy = ca * (Py - nnPy)
      cz = ca * (Pz - nnPz)

      snxPx = sa * (ny * Pz - nz * Py)
      snxPy = sa * (nz * Px - nx * Pz)
      snxPz = sa * (nx * Py - ny * Px)

      a(j, 0) = nnPx + cx + snxPx + Ax
      a(j, 1) = nnPy + cy + snxPy + Ay
      a(j, 2) = nnPz + cz + snxPz + Az
    NEXT j
  NEXT x
  Stp = Stp / 2
NEXT i
```

Scheme 3. Pseudocode 3: code to make random rotations.

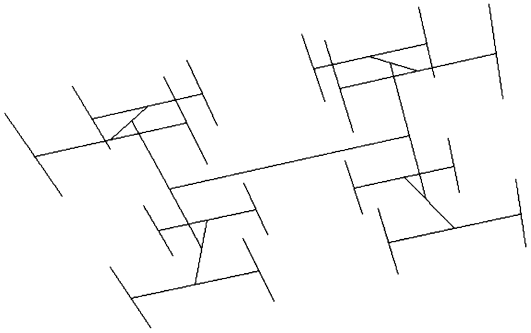

Fig. 7. This tree would fill the space

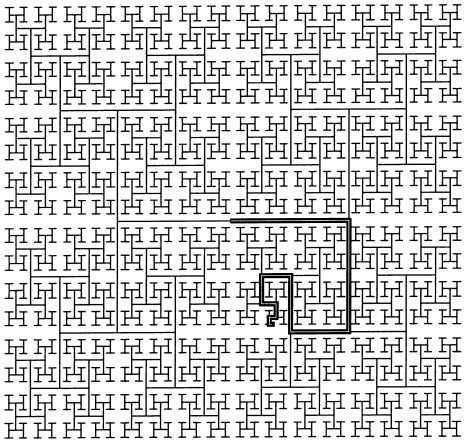

Fig. 8. Follow the binary path.

To reach the point $(0.001011, -0.01111)$ for example, the black path must be followed. The horizontal displacements are separated from the vertical ones as seen in Tables 1 and 2).

Table 1. Horizontal

.	0	0	1	0	1	1
r	l	l	r	l	r	

r, right; l, left.

Table 2. Vertical

–.	0	1	1	1	1
d	u	d	d	d	

d, down; u, up.

As can be seen in the horizontal table the 'R' under the dot means that the number 0.001011 is positive, and the 0s must be substituted by Ls (left) and the 1s by Rs (right). The last numeral (of a finite binary number) is always a 1 and does not have a match. If the number were negative, the letter under the dot would be, of course: L, and then, 0s would be transformed in Rs and 1s in Ls. The rule is the same for the vertical table.

This simple rule shows how it fills the *n*-space.

REFERENCES

1. González, J. Á., Árboles móviles. *Carrollia*, 1997, **53,** 10–12.
2. González, J. Á., Los árboles de González. *Cacumen Ingenio, Juegos y Humor*, 1984, **15,** 38–39.
3. Mandelbrot, B. B., *The Fractal Geometry of Nature*. Freeman, New York, 1977.