FRACTAL LANDSCAPES

PHIL KOOPMAN, JR.- NORTH KINGSTOWN, RHODE ISLAND

In the last issue of *Forth Dimensions*, I presented a program that used the Bresenham algorithm to draw lines on a computer graphics display. Now I will build upon those definitions to create a program that draws pseudo-randomly generated fractal landscapes with height-based coloring (including a sea-level) and hidden-surface elimination.

What's a Fractal?

The word fractal is short for "fractional dimension," and is used to describe various geometrical shapes that are not strictly one, two, or three-dimensional objects. Interestingly enough, most naturally occurring objects (such as coastlines) and natural phenomena (such as Brownian motion) are best described by fractal geometry[1].

To get a better feel for what a fractal is, consider the western coastline of the United States. From far away in space, the coastline looks more-or-less straight; that is, it looks like a one-dimensional line having a fractional dimension near 1. As an observer gets closer, the perceived length of the coastline and its fractional dimension increases as more details such as inlets and peninsulas become visible. In fact, with a very close look at an area like San Francisco Bay, the coastline becomes a two-dimensional object with a fractional dimension near 2.

How To Draw Random Fractals

To use the fractal concept to draw a landscape, let's first look at how a fractal can be used to draw a shape, starting from a line. We will look at a random fractal line, but of course there are other ways to draw fractal lines (see the recommended reading list).

```
SCREEN #15
\ Special 32-bit unsigned multiply
DECIMAL
\ This is a special-purpose unsigned multiply that returns only
\    the low-order 32 bits.  For a more generalized multiply,
\    see: P. Koopman, MVP-FORTH INTEGER & FLOATING POINT MATH
\         MVP-FORTH series Vol. 3 (revised), Mountain View Press

: XD*   ( UD1 UD2 -> UD3 )
   OVER  5 PICK  U*  >R >R
   4 ROLL   U*  DROP  >R
   U* DROP
   0 SWAP    R> 0 SWAP D+
   R> R> D+ ;



SCREEN #16
\ 32-BIT BASED LINEAR CONGRUENT RANDOM NUMBER GENERATOR
DECIMAL   \ From Knuth's Art of Computer Prog. vol. 2, page 170
DVARIABLE SEED        \ High 16-bits of SEED are actual random #
   \ Store all 32-bits initially to re-seed generator
3141592653. SEED D!

: RNDF   ( -> N )
  SEED D@  3141592621.  XD*   1. D+
  DUP  ROT ROT  SEED D! ;

\ The below random number generator is very poor, but
\    produces interesting smoothed/rounded rolling hills
\ : RNDF  ( -> N )
\    [ SEED 1+ ] LITERAL @ 31417 U*  1. D+ DUP ROT ROT SEED D! ;



SCREEN #17
\ FRACTAL LAYOUT
DECIMAL
\ SQUARE              P1+----x-----+P2
\ LAYOUT:               !  I !  II !   Roman #'s are square #'s
\                       x----x-----x   P1..P4 are corner point #'s
\                       ! IV ! III !
\                     P4!----x-----!P3
\ In the recursive routines, P1..P4 are the address of the
\   points within the layout array
\ Each recursion calculates the new "x" point height values
: WALL ;     \ Useful point for FORGETting
: RECURSE
      LATEST   PFA CFA ,   ;  IMMEDIATE

: CELL*  2*  ;      \ Change to fit bytes/cell of your machine
: CELL+  2+  ;      \ Change to fit bytes/cell of your machine
```

First, consider the straight line segment shown in Figure One-a. We will take the midpoint of that line segment and pull it to one side, as shown in Figure One-b. Next, we will recursively take each midpoint of the resulting line segments and pull them randomly to one side or the other as shown in Figures One-c and One-d. As you can see, this process quickly results in a wandering line. For the most pleasing shape, the amount of "pull" applied is cut in half at each level of recursion, forming a smooth result.

In order to extend this concept to an area instead of a line, the "Landscape" program on screens 15-28 forms a two-dimensional array. Each cell in the array holds the height of a point above or below sea level. The word **CALCULATE-SERVICE** recursively breaks this array into smaller and smaller squares, using the addresses of the four corners of the array instead of four pairs of (X,Y) coordinates. **SET-HEIGHTS** sets the heights for the array cells at the midpoints of the sides of the current square and for the center of the current square, then breaks the square up into four sub-squares (see the diagram on screen 17 for a description of the nomenclature used by the subdividing algorithm). After the data array has a height associated with each point, the program uses the **SEA-LEVEL** word to reassign all negative heights to sea level.

After the heights are computed, the landscape is drawn on the screen. As each point of the array is drawn, it is assigned a color based on height and the number of colors available.

Screen-Drawing Tricks

I have used several tricks in "Landscape" to speed up the screen-drawing time. This drawing time would be prohibitively long if conventional, three-dimensional graphics techniques were used.

The most time-consuming part of many graphics drawing programs involves 3-D transformations, especially rotations. On the other hand, a top or side view of a fractal landscape would not be terribly exciting. I solved this speed-versus-prettiness dilemma using two techniques: a "sleazy" rotation to elevate the rear of the picture, and an

```
SCREEN #18
 0 \ FRACTAL DATABASE
 1 DECIMAL
 2 5  CONSTANT #LEVELS   \ Number of recursion levels
 3 65 CONSTANT SIZE        \ array size = 1 +  2**(#LEVELS+1)
 4     \ NOTE: Change SIZE to 129 and #LEVELS to 6 for EGA
 5 \ SQUARE-P1 is a 2D array that holds heights of all grid points
 6 CREATE SQUARE-P1    SIZE SIZE * CELL* ALLOT
 7 SIZE 1-              CELL*  SQUARE-P1 + CONSTANT SQUARE-P2
 8 SIZE SIZE *     1-   CELL*  SQUARE-P1 + CONSTANT SQUARE-P3
 9 SIZE SIZE 1- *       CELL*  SQUARE-P1 + CONSTANT SQUARE-P4
10
11 : SCALE 2* 2* ;    \ Scale value of pixels per data array point
12 1 SCALE  CONSTANT DELTA
13 : AVE    ( U1 U2 -> UAVE )   \ unsigned average of 2 addresses
14     \ NOTE: This is a prime candidate for machine code speed-up!
15     0 SWAP  0  D+  2  U/MOD  SWAP DROP ;


SCREEN #19
 0 \ SPECIAL LINE DRAWS FOR FRACTAL LANDSCAPES
 1 DECIMAL
 2 : Y-CONVERT   ( HEIGHT Y1 -> Y2 )
 3     \ For now, assume tilted up 30 degrees in back, no X change
 4     \ Inputs are x/y data points & height, outputs screen coords
 5     + 2/  NEGATE  YMAX +    ;
 6 : F-MOVE   ( X HEIGHT Y-INDEX -> )
 7 \ Use the code on the next line for tracing if desired
 8 \   ." MOVE:"   SWAP U. U. CR    ?TERMINAL  ABORT" F-MOVE" ;
 9     SCALE Y-CONVERT  MOVE-CURSOR ;
10 : F-LINE   ( X HEIGHT Y-INDEX -> )
11 \ Use the code on the next line for tracing if desired
12 \   ." LINE:"   SWAP U. U. CR    ?TERMINAL  ABORT" F-LINE" ;
13     SCALE Y-CONVERT    DUP 0<
14     IF   ( Clip line that is off screen ) DDROP
15     ELSE   LINE  THEN ;


SCREEN #20
 0 \ INITIALIZE THE HEIGHT ARRAY  & CALCULATE COLOR FOR A HEIGHT
 1 HEX
 2 : INITIALIZE-SQUARE   ( -> )
 3     \ Fill all initial heights with 8181 for a recognizable tag
 4     SQUARE-P1   SIZE 0
 5     DO   DUP  SIZE CELL*  81 FILL   SIZE CELL* + LOOP  DROP
 6     20 SQUARE-P3 !          \ Initial values to slant landscape
 7     18 SQUARE-P4 !          \   "forward" for a better view
 8     -15 SQUARE-P1 !        -10 SQUARE-P2 ! ;
 9 : SET-COLOR    ( HEIGHT -> )    \ Figure color for given height
10 \ Adjust the "40" on the line below to individual taste.
11 \  In particular, change to a "18" for EGA
12     DUP 8 < IF    ( near sea level ) DROP  1
13              ELSE  40 /    #COLORS 2-  MOD  2+ THEN   COLOR ! ;
14 \ Redefine as : SET-COLOR  DROP 1 COLOR ! ;  for CGA/HIRES
15 DECIMAL


SCREEN #21
 0 \ DRAW THE HEIGHT ARRAY ON THE CRT DISPLAY
 1 DECIMAL
 2 : DRAW-SURFACE     ( -> )  \ Draw from bottom to top on screen
 3   SIZE 2-  0  DO  ( column )  I SIZE +  CELL*  SQUARE-P1 +
 4   10000  ( initial min Y value )   SIZE 1-  1  DO ( row )
 5   \ Test for hidden surface removal
 6   OVER    @  I SCALE    Y-CONVERT DDUP  >
 7    IF   ( new min y value means visible point )  SWAP DROP
 8      \  Add a 2* where indicated when using CGA/HIRES mode
 9      OVER @ SET-COLOR  J SCALE    ( 2* )
10      DUP DELTA ( 2*) + 4 PICK SIZE CELL* - CELL+ @ I 1- F-MOVE
11      DUP              4 PICK                     @ I    F-LINE
12           DELTA ( 2*) + 3 PICK SIZE CELL* + CELL+ @ I 1+ F-LINE
13   ELSE   ( hidden )  DROP    THEN
14   SWAP SIZE CELL* + SWAP    1 /LOOP
15   ?TERMINAL ABORT" BREAK"   DDROP  1 /LOOP ;
```

unconventional point-connection scheme to eliminate the need for spinning the picture.

A standard rotation of a landscape to elevate the rear of the picture involves using the equation:

NEWY = yvalue * sin(angle) + height * cos(angle)

for each height data point in the landscape array. In order to eliminate the scaled integer or floating-point arithmetic involved, I chose my rotation angle to be 30 degrees and changed the "* sin(angle)" term to a division by two. Then, to get rid of the cosine term, I decided to approximate cos(30)=.866 by 0.5 (division by two) and increased the original height value on line 7 of screen 28 to compensate. The elevation using this strategy is accomplished by **Y-CONVERT** on screen 19.

Even with the rear of the picture elevated, the result is pretty unexciting if points are connected by columns and rows. You would only see regularly spaced vertical lines with landscape profile lines wiggling horizontally across the screen. In order to fix this, lines 10 through 12 of screen 21 connect points in sideways "V" patterns to form a picture composed of diagonal lines instead of mostly horizontal and vertical lines. The lines are drawn and colored by columns of points, front to back.

It turns out that hidden-surface elimination, a major computational drain on many graphics programs, comes at almost no charge when using the drawing technique described above. Since points are drawn from front to back, lines 5 - 7 of screen 21 simply ensure that each new Y value for a point to be displayed is further up on the screen than any previous Y values for that column.

Running The Program

Simply type **LANDSCAPE** from the Forth "OK" prompt. The program will draw a landscape and wait for a keystroke on a PC-compatible machine with a Color Graphics Adapter (CGA) display. If you change the constants on lines 2 - 3 of screen 18, redefine the coloring word on lines 9 - 14 of screen 20, and substitute

(Continued on page 11.)

```
SCREEN #22
 0 \ SET-HEIGHT -- Set height of a point for recursive processing
 1 HEX
 2 : SET-HEIGHT   ( DH LEVEL PX VALUE PY VALUE -> DH LEVEL )
 3     ROT + 2/    ROT ROT  AVE
 4     DUP @  8181 =
 5        IF ( store )   SWAP  4 PICK  RNDF +-  +   SWAP !
 6        ELSE  DDROP THEN ;
 7
 8 DECIMAL
 9
10
11
12
13
14
15
```

```
SCREEN #23
 0 \ SET HEIGHTS FOR ALL THE "x" POINTS TO MAKE SUB-SQUARES
 1 DECIMAL
 2 : SET-HEIGHTS   ( P1 P2 P3 P4  DELTA-H  LEVEL# -> <unchanged> )
 3 \  Following 2 lines are debug/trace code to watch recursion
 4 \   CR 6 PICK U.  5 PICK U. 4 PICK U. 3 PICK U. OVER U. DUP U.
 5 \   ?TERMINAL ABORT" SET-HEIGHTS"
 6     ( ave P1/P2 )  6 PICK  DUP @   7 PICK  DUP @  SET-HEIGHT
 7     ( ave P2/P3 )  5 PICK  DUP @   6 PICK  DUP @  SET-HEIGHT
 8     ( ave P3/P4 )  4 PICK  DUP @   5 PICK  DUP @  SET-HEIGHT
 9     ( ave P1/P4 )  6 PICK  DUP @   5 PICK  DUP @  SET-HEIGHT
10     ( ave P1/P3 )  6 PICK  DUP @   6 PICK  DUP @  SET-HEIGHT ;
11
12
13
14
15
```

```
SCREEN #24
 0 \ WORD TO SET UP PARAMETERS FOR SUB-SQUARES # 1-2
 1 DECIMAL
 2 : SQUARE1   ( P1 P2 P3 P4  DELTA-H  LEVEL# -> <2.copies> )
 3     6 PICK                      DUP 7 PICK   AVE
 4     OVER  7 PICK AVE          9 PICK  7 PICK    AVE
 5     6 PICK  2/    6 PICK  1- ;
 6
 7 : SQUARE2   ( P1 P2 P3 P4  DELTA-H  LEVEL# -> <2.copies> )
 8     6 PICK  6 PICK   AVE      6 PICK
 9     DUP    7 PICK   AVE       OVER     7 PICK    AVE
10     6 PICK  2/    6 PICK  1- ;
11
12
13
14
15
```

```
SCREEN #25
 0 \ WORD TO SET UP PARAMETERS FOR SUB-SQUARES # 3-4
 1 DECIMAL
 2 : SQUARE3   ( P1 P2 P3 P4  DELTA-H  LEVEL# -> <2.copies> )
 3     6 PICK   5 PICK   AVE    6 PICK  6 PICK   AVE
 4     6 PICK                   DUP     7 PICK    AVE
 5     6 PICK  2/    6 PICK  1- ;
 6
 7 : SQUARE4   ( P1 P2 P3 P4  DELTA-H  LEVEL# -> <2.copies> )
 8     6 PICK  4 PICK   AVE     6 PICK  5 PICK    AVE
 9     6 PICK  6 PICK   AVE             6 PICK
10     6 PICK  2/    6 PICK  1- ;
11
12
13
14
15
```

(Screens continued of next page.)

(Continued from page 15.)

the appropriate graphics-mode initialization word on line 9 of screen 28 you should be able to run this program with any graphics display. Playing with the scale factor on screen 18 and/or the horizontal scale factors commented out in screen 21 may make the pictures more pleasing on some displays.

References

1. *Fractals: Form, Chance, and Dimension*, B.B. Mandelbrot. W.H. Freeman and Company, San Francisco, 1977.

Recommended Reading

"3-D Fractals," M. van de Panne. *Creative Computing*, July 1985, pp. 78-82.

Art of Computer Programming, D.E. Knuth. Addison-Wessley Publishing, Vol. 2, 1981 (includes a discussion of random number generators).

"Fractals," P.R. Sorensen. *BYTE*, September 1984, pp. 157-172.

The Fractal Geometry of Nature, B.B. Mandelbrot. W.H. Freeman and Company, San Francisco, 1983.

"IBM Fractal Graphics," P.W. Carlson. *Compute!*, March 1986, pp. 78-80.

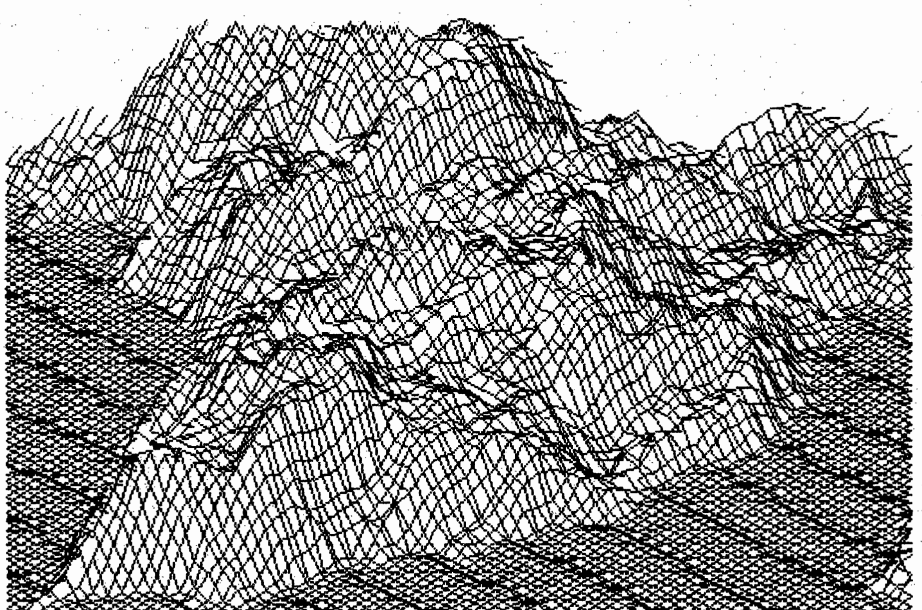
Phil Koopman, Jr. is the vice-president and chief engineer for WISC Technologies. The program discussed in this article was originally developed as a demonstration for the WISC CPU/16.

```
SCREEN #26
\ RECURSIVE PROCEDURE TO SET HEIGHTS FOR RANDOM 3-D TERRAIN
DECIMAL  \ BASED ON SUB-DIVIDED SQUARE FRACTALS
: CALCULATE-SURFACE   ( P1 P2 P3 P4  DELTA-H  LEVEL# -> )
   SET-HEIGHTS
   DUP        ?TERMINAL ABORT" BREAK IN CALCULATE-SURFACE"
   IF   ( non-zero level )
       SQUARE1      RECURSE
       SQUARE2      RECURSE
       SQUARE3      RECURSE
       SQUARE4      RECURSE
   THEN
   DDROP DDROP DDROP ;


SCREEN #27
\ SEA-LEVEL -- SET SEA LEVEL FOR NEGATIVE HEIGHT POINTS
DECIMAL
: SEA-LEVEL     ( -> )
   SQUARE-P1    SIZE 0 DO
      SIZE 0 DO
         DUP @   DUP 0<
         IF   ( below sea level -- add fudge factor for waves )
            1 AND  I J + +   7 AND    OVER !
         ELSE DROP THEN
      CELL+    LOOP
   LOOP     DROP   ;


SCREEN #28
\ MASTER PROCEDURE TO DRAW A RANDOM 3-D FRACTAL
DECIMAL   \ BASED ON SUB-DIVIDED SQUARES
DVARIABLE SEED-SAVE   \ Saves random seed.  Placing the saved
   \ value back into SEED will re-create the same landscape
: LANDSCAPE  ( -> )
   SEED D@   SEED-SAVE D!  INITIALIZE-SQUARE
   CR ." Computing new heights"
   SQUARE-P1  SQUARE-P2  SQUARE-P3  SQUARE-P4
      YMAX 2/   #LEVELS  CALCULATE-SURFACE
   CR ." Computing sea level"    SEA-LEVEL
   SET-CGA-MODE   \ Change to SET-EGA-MODE or SET-CGA-HIRES-MODE
\     SEED-SAVE D@ CR ." SEED=" D.    ( Optional SEED display )
   DRAW-SURFACE
   CR ." Press any key to continue"   KEY DROP
   SET-TEXT-MODE    ;

```



FORTH
DIMENSIONS