Pergamon

Comput. & Graphics, Vol. 19, No. 2, pp. 309–313, 1995

Printed in Great Britain
0097-8493/95 $9.50 + .00

0097-8493(94)00157-X

AUTOMATIC GENERATION OF GENERAL QUADRATIC MAP BASINS

JULIEN C. SPROTT
Department of Physics, University of Wisconsin, Madison, WI 53706, USA,
e-mail: sprott@juno.physics.wisc.edu

and

CLIFFORD A. PICKOVER
IBM Watson Research Center, Yorktown Heights, New York 10598, USA,
e-mail: cliff@watson.ibm.com

Abstract—In this informal article, we describe simple approaches whereby a computer can automatically select parameters and generate a large collection of diverse, aesthetically appealing fractal patterns based on general quadratic map basins. Computational recipes are included to encourage reader involvement. In essence, we describe methods for teaching a computer to be both an artist and a critic of its own art.

1. INTRODUCTION

An infinite number of monkeys with an infinite number of typewriters (word processors, nowadays) will eventually reproduce every work of Shakespeare. The problem is that someone has to sort through all the gibberish to find the occasional gem. On the other hand, the criteria for visual art are much less constrained than for literary composition. A monkey with a paintbrush has a reasonable chance of producing a pattern that someone would consider artistic.

We have been experimenting with replacing the monkey by a computer and having it generate a collection of visually interesting patterns. Unlike the monkey, the computer can be trained to select those images that are likely to appeal to humans. The procedure is to take some simple equations with adjustable coefficients chosen at random, solve the equations with the computer, and display those which meet criteria that we have found to be strong indicators of artistic quality.

One approach is to use equations with chaotic solutions. Such solutions are unpredictable over the long term yet exhibit interesting structures as they move about on a strange attractor, a fractal object with non-integer dimension. Our books show many examples of computer art produced by these and related methods[1–3]. Here we propose another simple method for producing fractal art.

2. GENERAL TWO-DIMENSIONAL QUADRATIC ITERATED MAPS

Traditionally when physicists or mathematicians saw complicated results, they often looked for complicated causes. In contrast, many of the shapes which follow describe the fantastically complicated behavior of the simplest of formulas. The results should be of interest to artists and nonmathematicians, and anyone with imagination and a little computer programming skill.

One lesson of chaos theory is that simple, nonlinear equations can have complicated solutions. The solutions are most interesting if they involve at least two variables, x and y, which can be used to represent horizontal and vertical positions, and if the variables are advanced step-by-step in an iterative process. The simplest nonlinearities are quadratic (x^2 or xy). The most general two dimensional quadratic iterated map is:

$$x_{new} = a + bx + cx^2 + dxy + ey + fy^2$$

$$y_{new} = g + hx + ix^2 + jxy + ky + ly^2$$

where a through l are constants chosen at random but held the same throughout the calculation. Think of this as a mathematical feedback loop where new values for x and y are used again in the next round of iteration by setting $x = x_{new}$ and $y = y_{new}$. The constants can be considered as settings of a combination lock, each revealing a different pattern for you to admire. They open doorways to an infinite reservoir of magnificent shapes and forms.

There are many ways to display the solution to our general maps. One way is to plot successive values of x and y as dots on the screen. Many of the solutions will move toward a point and remain there. Others will settle into a periodic orbit, or will move off toward infinity. The interesting cases are the chaotic ones which remain confined to a limited region but whose orbits produce a strange attractor with intricate fractal structure. You can choose the starting values of x and y arbitrarily within the basin of attraction, but you should discard the first few iterates since they probably lie off the attractor.

Another way you can display the solution is to solve the equations with many different starting values and count the number of iterations required for the solution to wander outside some region in the xy plane. You

can use the final number of iterations to determine the color for that point in the plane of initial xy values. Plots produced in this way are called escape-time fractals because the color contours indicate the time required for the orbit to escape from the region. Some initial values may have orbits that never escape, and so you need to have a "bailout condition" beyond which your program stops iterating and moves on to test the next initial condition.

Some of the most artistic examples of escape-time fractals explored in the past have been Julia sets of the map $z_{new} = z^2 + c$, where z and c are complex numbers [4,5]. In terms of x and y this map is

$$x_{new} = x^2 - y^2 + a$$

$$y_{new} = 2xy + b$$

and the test escape region is normally taken as a circle of radius of 2 centered on the origin. These traditional Julia sets are a special case of the more general maps we propose in this article.

3. COMPUTER ART CRITIC

Visually interesting escape-time fractals are those for which the orbits escape slowly. We have found that escape times between about a hundred and a thousand produce the most fascinating maps. We start by choosing the twelve coefficients randomly over the range -1.2 to 1.2 in increments of 0.1, and then we iterate the equations for our generalized quadratic map with initial conditions of $x = y = 0$. If we find a group of coefficients that result in the initial point escaping beyond a circle of radius 1000 centered at the origin within 100 to 1000 iterations, then we save the param-

```
DIM a(12)                              'Array of coefficients
RANDOMIZE TIMER                        'Reseed random numbers
SCREEN 12                              'Assume VGA graphics
n% = 0
WHILE INKEY$ = ""                      'Loop until a key is pressed
    IF n% = 0 THEN CALL setparams(x, y)
    CALL advancexy(x, y, n%)
    CALL testsoln(x, y, n%)
    IF n% = 1000 THEN CALL display: n% = 0
WEND
END

SUB advancexy (x, y, n%)               'Advance (x, y) at step n%
SHARED a()
xnew = a(1) + x * (a(2) + a(3) * x + a(4) * y) + y * (a(5) + a(6) * y)
y = a(7) + x * (a(8) + a(9) * x + a(10) * y) + y * (a(11) + a(12) * y)
x = xnew
n% = n% + 1
END SUB

SUB display ()                         'Plot escape-time contours
FOR i% = 0 TO 639
    FOR j% = 0 TO 479
        x = -5 + i% / 64
        y = 5 - j% / 48
        n% = 0
        WHILE n% < 128 AND x * x + y * y < 1000000
            CALL advancexy(x, y, n%)
        WEND
        PSET (i%, j%), n% MOD 16
    NEXT j%
NEXT i%
END SUB

SUB setparams (x, y)                   'Set a() and initialize (x,y)
SHARED a()
x = 0: y = 0
FOR i% = 1 TO 12: a(i%) = (INT(25 * RND) - 12) / 10: NEXT i%
END SUB

SUB testsoln (x, y, n%)                'Test the solution
IF n% = 1000 THEN n% = 0               'Solution is bounded
IF x * x + y * y > 1000000 THEN        'Solution escaped
    IF n% > 100 THEN n% = 1000 ELSE n% = 0
END IF
END SUB
```

Fig. 1. ESCAPE.BAS: A BASIC program for automatically producing an unlimited number of escape-time fractals similar to those shown.

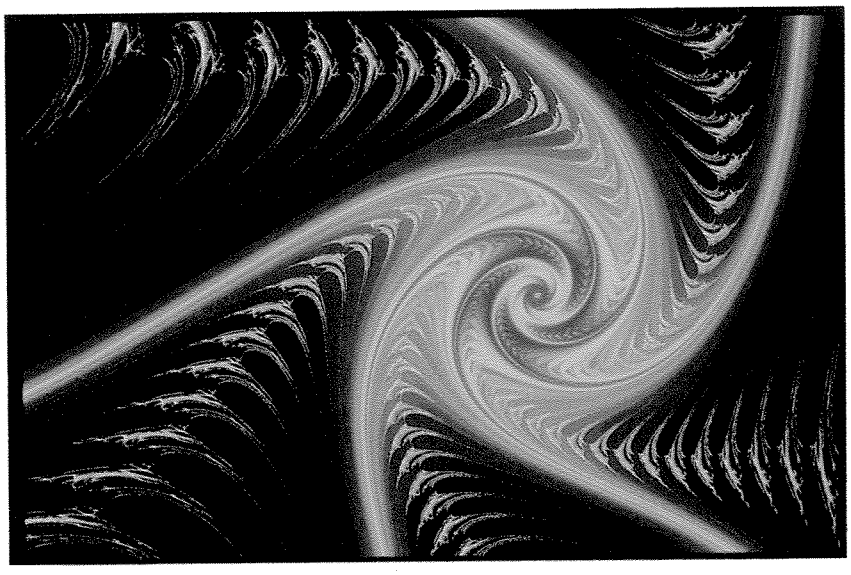

Fig. 2. Escape-time fractal for parameters EHPLWTDJRCAP.

eter set and compute an escape-time fractal for a particular region of the x-y plane. The choice of 1.2 for the coefficient range is about optimal for speeding the search, and it allows the coefficients to be compactly represented as letters of the alphabet ($A = -1.2$, $B = -1.1$, through $Y = 1.2$) for easy reference and replication.

The listing ESCAPE.BAS (Fig. 1) shows a BASIC program for producing an endless succession of escape-time fractals by this method. It should run without modification under QBASIC, QuickBASIC, VisualBASIC for MS-DOS, or PowerBASIC on the IBM PC. It assumes VGA graphics (640 × 480 pixels × 16 colors). When run on a 486DX33 under PowerBASIC 3.0, the program takes on average about ten seconds to find each interesting case and about a minute to plot it. In the process of searching for what we call "beautiful parameters," it discards about 300 sets for every one it saves for plotting. This still leaves about 200 trillion cases, nearly all of which are different. If you view them at the rate of one per second, it would take over six million years to see them all, and thus it is very unlikely that any of the patterns you produce will ever have been seen before. We often run the program overnight and capture the screens to graphics files, which can be rapidly examined the next morning.

4. SAMPLE ARTWORK

Although the programs and methods we have described work well with personal computers and produce good-quality patterns within VGA screen limits, we became curious as to how our approach could be

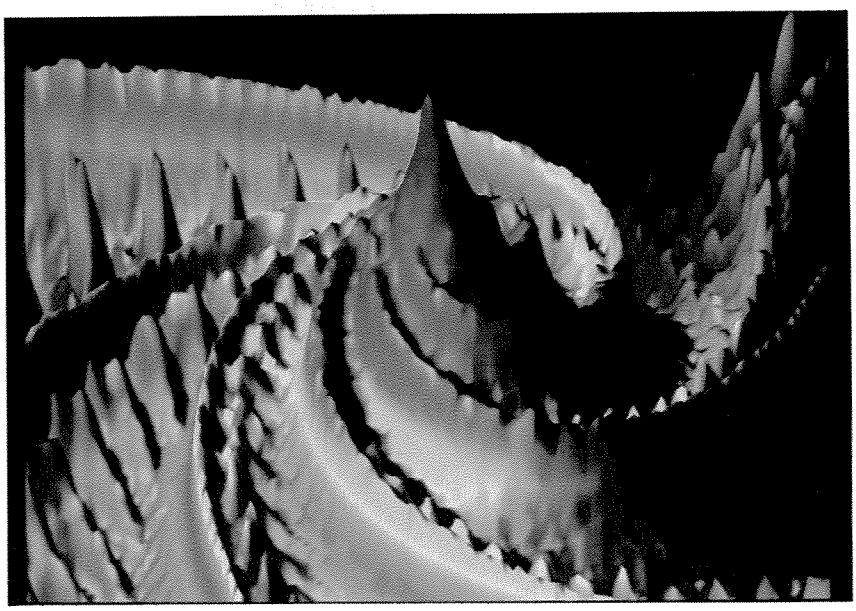

Fig. 3. Mount Fractalia, a three-dimensional representation of Fig. 1.

extended to higher resolutions (Figs. 2–8). Naturally, high-resolution, high-iteration computations would take longer to perform. To overcome these problems and push our method to the limit, we made use of IBM's Power Visualization System, a graphics supercomputer which can compute our general quadratic iterated map in parallel using up to 32 Intel i860XP processors. This means that the computer can simultaneously work on different portions of the artwork and therefore greatly reduce the time needed to explore a large number of images to find interesting structures. With our software and special hardware we can compute and view images at VGA resolution in under a second. The images in this article were computed at a resolution of 1600 × 2400 pixels using 1000 iterations for each pixel. Computation and display required less than a minute for each high-resolution image. The final step in aesthetic presentation involves the mapping of escape times to colors. Although beautiful images result from simple color mappings, such as the one used in the BASIC code, we used custom software which permits the algorithmic or mouse selection of colors from a palate of 2^{24} colors in a convenient fashion. It is also possible for us to render the maps in three dimensions, representing escape time as height. The Power Visualization System enables us to fly-by the 3D maps in real-time, permitting us to make video-tape animations characterizing the intricate behavior of these functions.

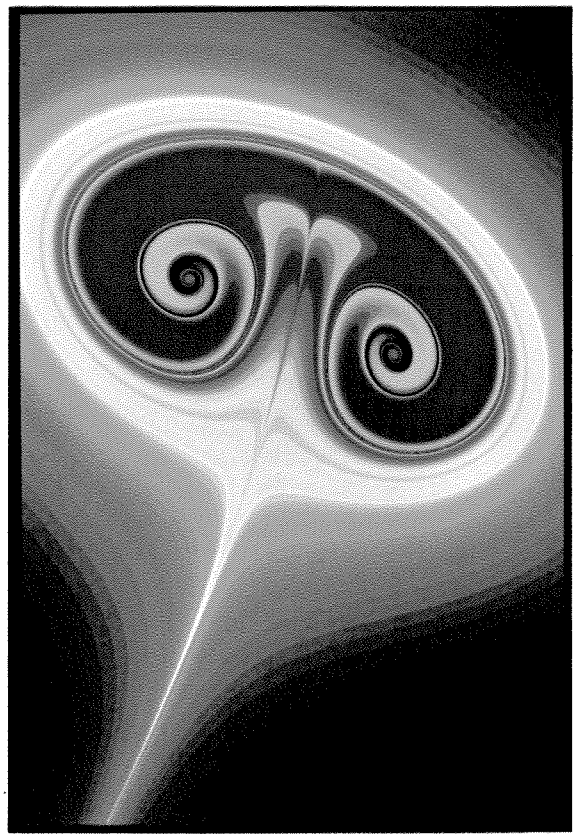

Fig. 5. Escape-time fractal for parameters MWRQFVMKUVFK.

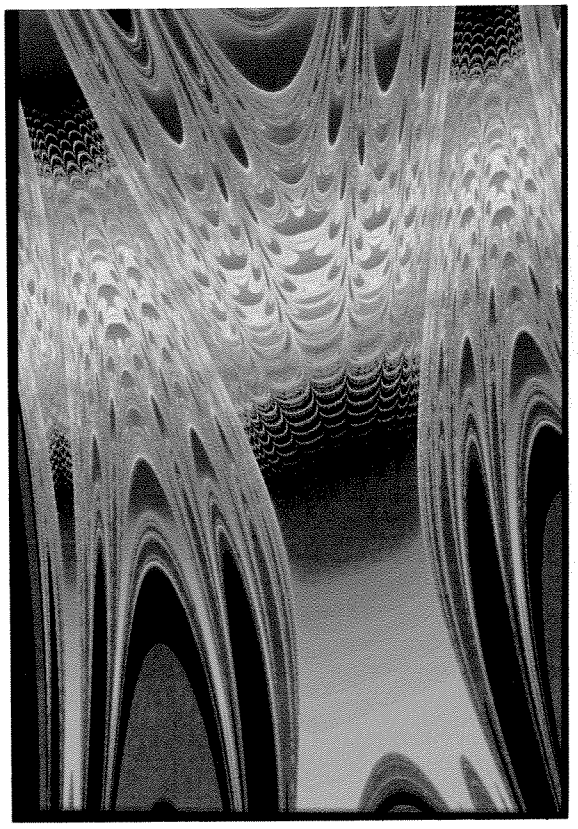

Fig. 4. The Fractal Curtain, an escape-time fractal for parameters WBMLNRQMNRAA.

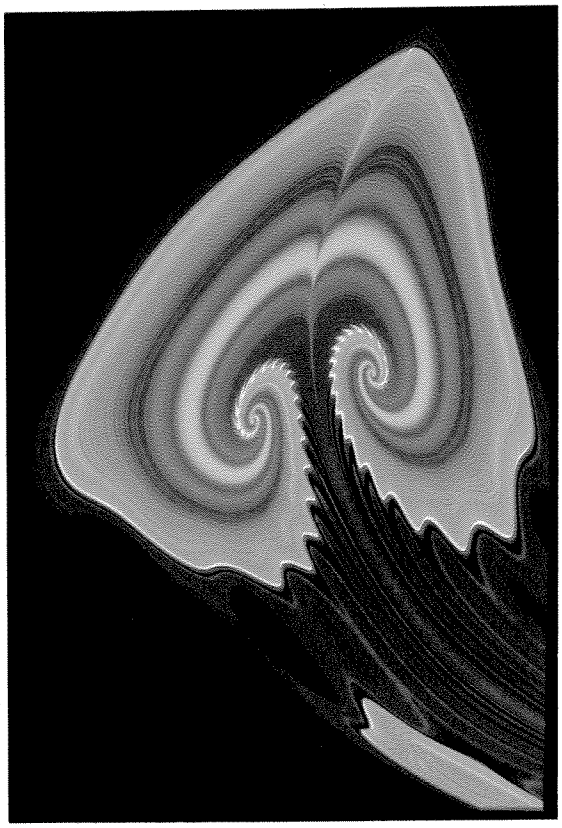

Fig. 6. Escape-time fractal for parameters KUONOVSVFLAR.

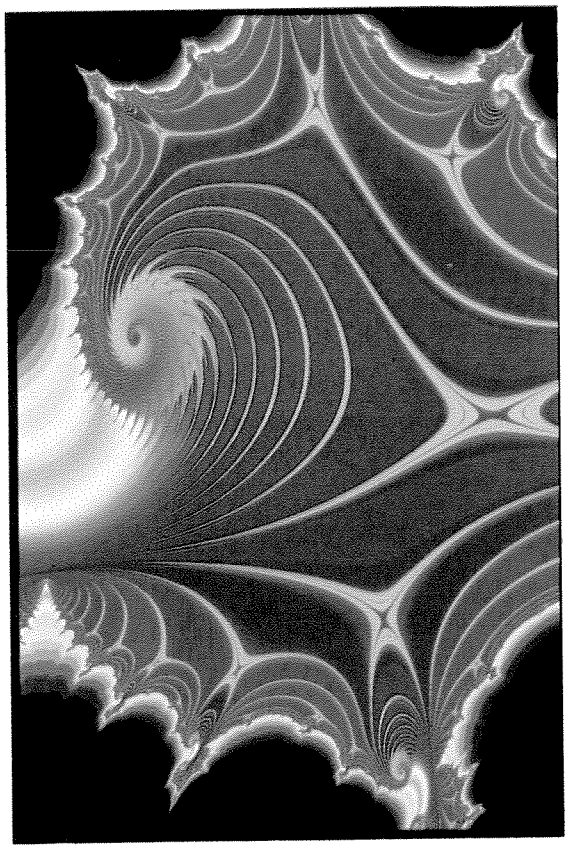

Fig. 7. Escape-time fractal for parameters EMXI-XPQMMTWRJ.

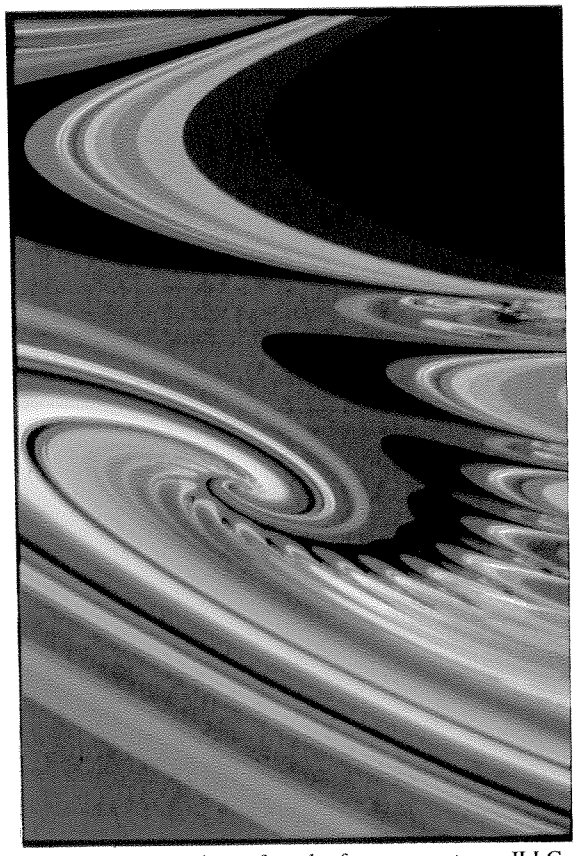

Fig. 8. Escape-time fractal for parameters JLLG-JTKDCRWY.

In the future, high-end methods such as these will be accessible to everyone as PC power increases, cost decreases, and Internet collaborations such as ours become commonplace[2]. In our own collaboration, input parameters were generated on a personal computer in Wisconsin, electronically sent over the Internet to New York, at which point they were read by the Power Visualization System and rendered.

To artists and computer graphics programmers, the automated generation of our generalized quadratic-map representations has great appeal because the speed allows the human analyst the freedom to experiment with many parameters. Students and teachers will enjoy such an approach as they explore and demonstrate the complexity and chaos associated with simple formulas. Mathematicians may find the approach useful because the sheer number of different structures they can generate has the potential for making complex behaviors apparent which might have been overlooked using traditional approaches.

5. FURTHER SUGGESTIONS

The method we have described is simple and effective, but we do not claim it is the final word on automated computer art. You can explore other criteria for selecting the patterns and other ways to display them. We chose a generalized two dimensional quadratic map to emphasize the complexity and variety that arises from simple equations. You can easily extend the technique to other mathematical functions and to higher dimensions. The same ideas can be used for automatic generation and evaluation of computer music. We would like to hear of any interesting results obtained by readers using our approach.

REFERENCES

1. J. C. Sprott, *Strange Attractors: Creating Patterns in Chaos,* M&T Books, New York (1993).
2. C. A. Pickover, *Mazes for the Mind: Computers and the Unexpected,* St. Martin's Press, New York (1992).
3. C. A. Pickover, *Chaos in Wonderland: Visual Adventures in a Fractal World,* St. Martin's Press, New York (1994).
4. H. O. Peitgen and P. H. Richter, *The Beauty of Fractals: Images of Complex Dynamical Systems,* Springer-Verlag, New York (1986).
5. B. Mandelbrot, *The Fractal Geometry of Nature,* Freeman, New York (1993).