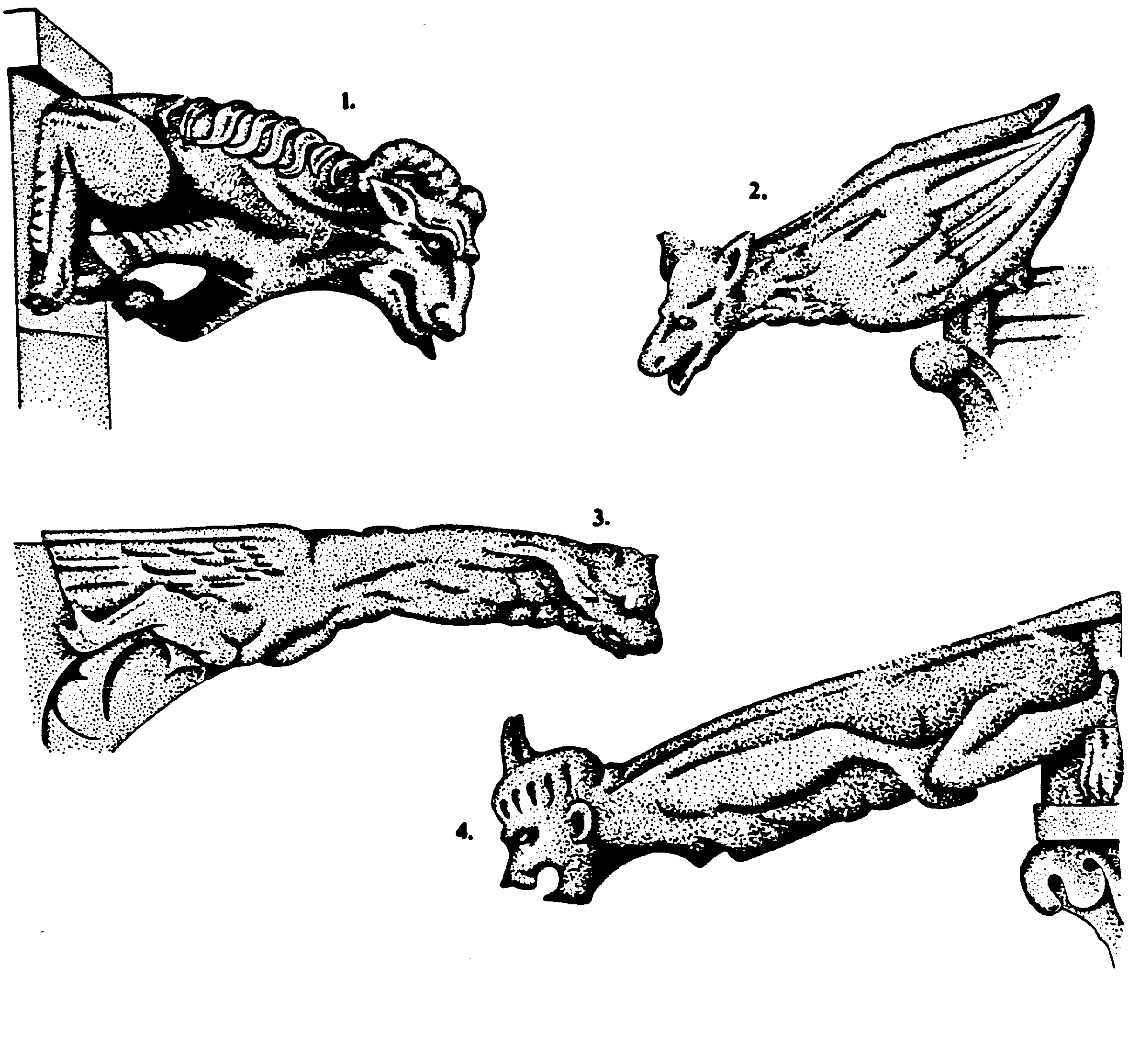
Volume 3, August, 1988

The Journal of Chaos and Graphics



Edited by Clifford A. Pickover

CONTENTS

Scope of Journal -

The Journal of Chaos and Graphics publishes articles, notes, reviews, notices, computational recipes, and correspondence on the subject of the graphics of chaos. ***JCG*** involves the study of how complicated behavior and structures can arise in systems which are based on simple rules. Topics for the journal include: iteration, cellular automata, bifurcation maps, Julia sets, Life, creation of the patterns of nature from simple rules, phase plane portraits ..., primarily ***aesthetic and unusual graphics derived from mathematics***. In addition, a "***Potpourri***" section will contain short articles on mathematical curiosities, and interesting equations and their graphs. Contributions can include good-quality monochrome graphics with a few-paragraph description on how the graphics were created.

Information for Contributors

Submissions should be sent to the editor (CP), along with any graphics which are of sufficient quality to be incorporated into the newsletter. Aesthetically intriguing graphics are definitely desirable. Include a short biographical sketch and full mailing address. Since the newsletter will have a varied audience, please explain all technical terms used. Along with any graphics, include a description of how figures are generated, including when possible: the device used, generating equation and technique, picture resolution, any novel features of the algorithm or plot, and other relevant parameters. Authors should send a ***floppy*** containing the text the article when possible.

The editor (CP) reserves the right to edit any materials submitted to conform with ***JCG*** style, clarity, space considerations, and editorial policy. The author is urged to include short pieces of ***pseudocode***, but to write in such a way that the layperson could grasp the essential ideas being discussed. Short articles ranging from 1 page to 3 pages are encouraged.

Sometimes references are made to "IBM Research Reports", and these can be obtained from: IBM Thomas J. Watson Research Center, Distribution Services F-11 Stormytown, PO Box 218, Yorktown Heights, NY 10598. ***JCG***, Volume 1 (RA number 186) (April 1987) and Volume 2 (RA number 192) (July 1987) can be obtained from this address.

About the Cover

To ancient man, Chaos represented the unknown, the spirit world -- menacing, nightmarish visions that reflected man's fear of the irrational and the need to give shape and form to his apprehensions. ***This volume's cover displays gargoyles from the Cathedral at Freiburg Germany.***

Copying

Copying of articles is permitted provided that the copies are not made or distributed for direct commercial advantage and credit to the source is given. Readers are free to use algorithms appearing in this journal in their publications, however whenever possible please include a reference to this work.

Subscriptions

Those interested in subscribing or in sending submissions should send their address to the address below. ***JCG*** will be published once or twice a year.

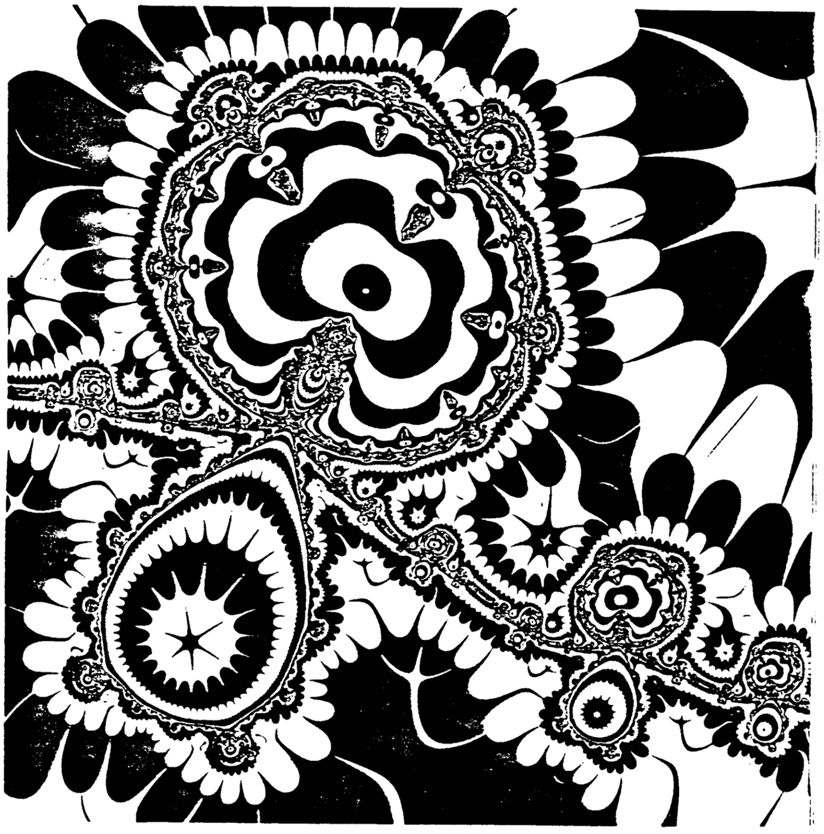
Clifford A. Pickover
Editor, Journal of Chaos and Graphics
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598.

Journal of Chaos and Graphics - Briefing

Readers may be interested to know that ***JCG*** has been very favorably reviewed in *Computer Pictures* (1988, January/February, 88-90) and the *Whole Earth Review* (1987, Winter, 57: 36-37). The requests are numerous, and the readership is growing!

In this issue, we are especially pleased to have the works of Michael Keith of the David Sarnoff Research Center at Princeton, Rudy Rucker, author of numerous books including *Infinity and The Mind* (Bantam), and William Benzon, writer, artist, and musician.

As with the last issue, I present readers with another Halley Map. For more information on how to generate this pattern and some of its mathematical properties see: Pickover, C. (1987) Chaos and Halley's Method (IBM RC 12747). Also be on the look-out for my new book, *Computers, Pattern, Chaos, and Beauty*, to be published by Springer-Verlag near the end of 1988. Presented below is a graphical result on Halley's method for a one-parameter family of rational functions in order to gain insight as to where the method can be relied upon and where it behaves strangely. The resulting plots reveal a visually striking and intricate class of patterns indicating behavior ranging from stable points to chaos. Iterative approximation methods such as Halley's method occur frequently in science and engineering.



What is a Sierpinski Gasket? - by Clifford A. Pickover

Number theory -- the study of properties of integers -- is an ancient discipline. Much mysticism accompanied early treatises (1); however, today integer arithmetic is important in a wide spectrum of human activities and has repeatedly played a crucial role in the evolution of the natural sciences (for a description of the use of number theory in communications, computer science, cryptography, physics, biology and art, see (2)). One of the most famous integer patterns in the history of mathematics is Pascal's triangle (PT) (Biaise Pascal was the first to write a treatise about this progression in 1653 -- although the pattern had been known by Omar Khayyam as far back as 1100 A.D.). The first 7 rows of Pascal's triangle can be represented as:

```
        1
       1 1
      1 2 1
     1 3 3 1
    1 4 6 4 1
  1 5 10 10 5 1
1 6 15 20 15 6 1
```

The figure on the next page represents Pascal's triangle computed with modular arithmetic. It is Pascal's triangle, mod(3); i.e., points are plotted for all numbers divisible by 3.

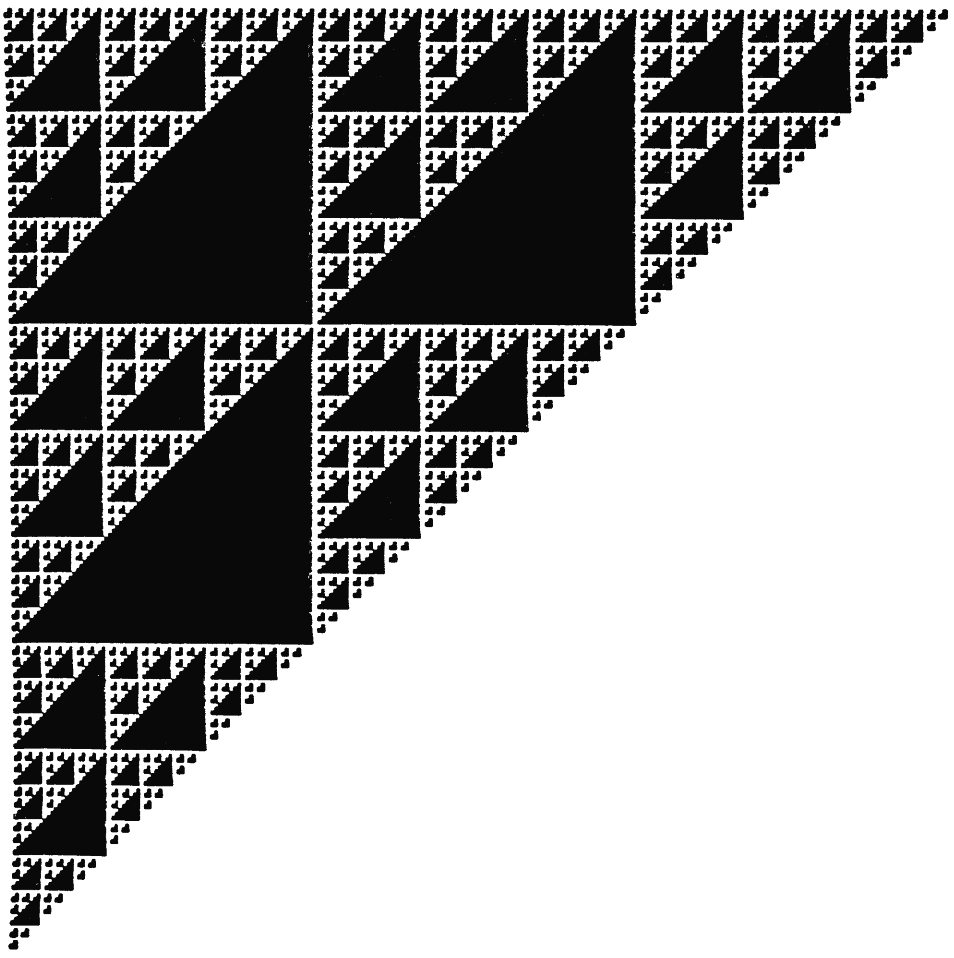
The resulting shapes are of interest mathematically, and they reveal a visually striking and intricate class of patterns which make up a family of regular, fractal networks. Fractals are patterns which have increasing "detail" with increasing magnification (e.g. the edge of a coastline). The fractals often of most interest are ones that are self-similar, for example, if we look at any one of the triangular motifs within Pascal's triangle we notice that the same pattern is found at another place in another size. The two-dimensional networks are known as Sierpiński gaskets which share important geometrical features with percolation problems and cellular automata. The Sierpiński gasket consists of triangles nested in one another "like Chinese boxes or Russian dolls".

```
/* Compute Pascal's Triangle using Modular Arithmetic */
Variables: c has the modular value of the Pascal triangle for a
given row, n, and column, r.

  p(*)=0;c(*)=0;                       /* initialize c and p arrays    */
  do n = 1 to 255;                     /* 255 rows                     */
    do r = 2 to n;                     /* generate the entries in a row */
       c(r)=mod(p(r)+p(r-1),imod);     /* imod = modulus index chosen  */
       if c(r)= 0 then PLOTDOT(n,r);   /* place dot at position (n,r)  */
    end;
    p(*)=c(*);                         /* update p array for next row  */
  end;
```

For Further Reading

1. Spencer, D. (1982) Computers in Number Theory. Computer Science Press: Maryland.
2. Schroeder, M. (1984) Number Theory in Science and Communication. Springer-Verlag: New York.
3. Pickover, C. (1986) Exotic symmetries in Sierpiński gaskets formed from large Pascal's triangles, (IBM Watson Lab, RC 12106).
4. Mandelbrot, B. B., *The Fractal Geometry of Nature*, Freeman, San Francisco (1983).

A Gasket



Logical Chaos - by Michael Keith

Michael Keith is a member of technical staff at the David Sarnoff Research Center. He is currently working on computer graphics software development, and is also interest in computer art, fractals, realistic image synthesis, and computer music. He can be reached at the David Sarnoff Research Center, Princeton, NJ.

This article describes beautiful and intricate geometric shapes produced by mathematical feedback. In the world of mathematics, feedback is often referred to as "iteration" or "recursion".

In particular, the accompanying figures show the results of iterating simple functions over the complex numbers, but with an additional twist: the functions contain, in addition to arithmetic operations, Boolean logic operations.

Since it may not be obvious what it means to do a Boolean logic operation on a complex number, a bit of explanation is in order. Suppose we have a complex number z = a + bi. We convert both a and b to binary with a fixed number of binary digits after the decimal point (I have used 16 in all these examples). We then operate on both a and b using the equations

```
a = a OP val1
and
b = b OP val2
```

where OP is a boolean logic operation (AND, OR, XOR, etc.), and val1 and val2 are two binary values. The operations are understood to be bit-by-bit operations (i.e., 01 OR 10 is 11, not 1). Finally, after operating on a and b using this procedure, we convert them back into real numbers (with, of course, some loss of precision because of the intermediate conversion to fixed-length binary).

We will represent this operation, on a given complex number z, by

```
z OP (val1,val2)
```

As illustrated by the three figures, these functions (in combination with other functions such as polynomials or trig functions) add another dimension of complexity and beauty to the already large space of possibilities in iterated functions over C. Aside from producing interesting pictures, logic functions have the advantage of being readily accessible and having fast execution speed on even the smallest of computers.

The figures were all obtained from slight variations on the function associated with the Mandelbrot set, namely

$z = z^2 + c.$

Specifically, they were produced from the following data (values are listed in hexadecimal):

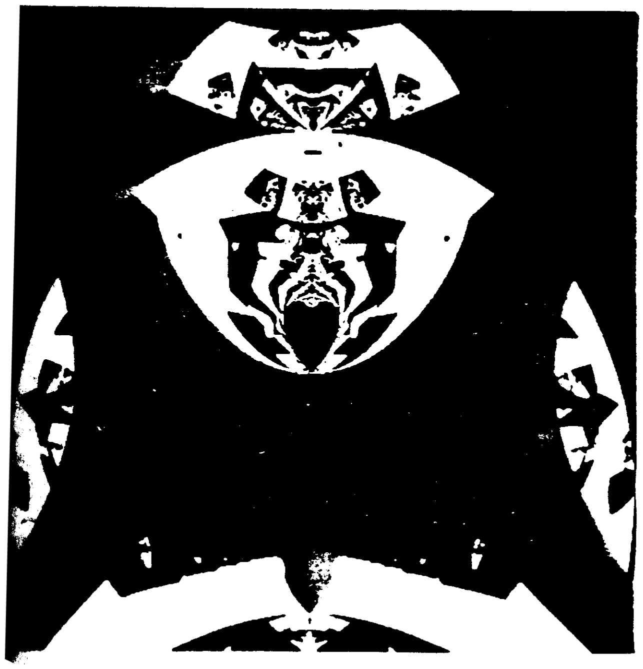
```
Fig. Function

1    z**2 XOR (1.0000,0) + c
2    z**2 XOR (.F000,0)  + c
3    z**2 AND (F.0FFF,0) + c
```

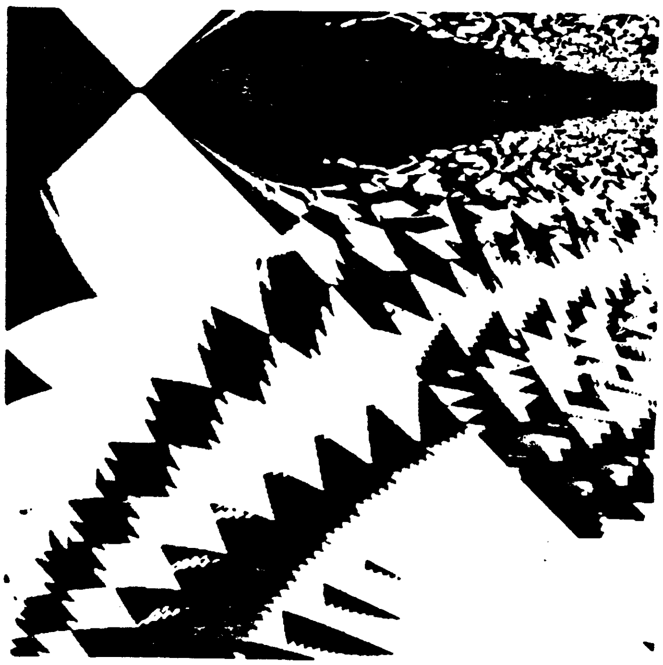
The plot centers for Figures 1, 2, and 3 are: (-.82,0), (-.7,.3), and (-.9, 0), respectively. These pictures, as with most chaotic system pictures, are quite striking. Figure 1 is quite reminiscent of C. Pickover's "biomorph"; particularly since in the middle of the largest biomorph form there is an almost perfectly shaped "heart"! The other two figures illustrate the other delicate effects that can appear.

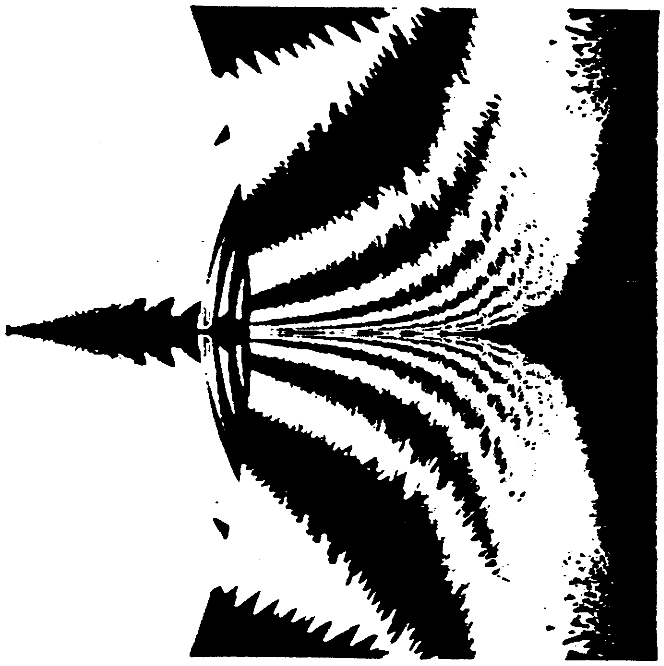
There is clearly much room left for further study. I have not tried any functions where val2 <> 0, nor any other logic functions besides XOR and AND. These possibilities are left for your experimentation.

For Further Reading

1. Pickover, C. (1987) Biomorphs: computer displays of biological forms generated from mathematical feed back loops. *Computer Graphics Forum* 5(4): 313-316.
2. Sorenson, P. (1984) Fractals. Byte. Sept. 9: 157-172 (a fascinating introduction to the subject).


1


2


3

Efficient Computation of Dynamical Maps - by Michael Keith

Michael Keith is a member of technical staff at the David Sarnoff Research Center, Princeton, NJ (see previous article for biographical information).

This article suggests methods for improving the speed of computation for iterated systems involving complex numbers. For example, suppose we wish to construct images of an iterated dynamical system over the complex numbers C. Such images are constructed by the following well-known algorithm: For each point of the image, initialize the complex number z to some specific initial value, and then iterate the function z' = f(z) until either (a) the value of z goes outside the radius-2 circle centered at the origin (which means that the point is being attracted to infinity) or (b) the number of iterations exceeds some arbitrary limit N (in which case we assume that the point is being attracted to 0). The point is then colored based on which attractor it belongs to, and also, usually, its velocity.

As one quickly discovers the first time one writes a program to produce such pictures, these computations can take a long time - minutes to hours, depending upon the computer and software used. This is because a nice image requires at least several-hundred pixel resolution in each of X and Y, and the value of N must be at least 50 or so. Thus we are looking at several million computations of f(z) just to create one image. This is exacerbated by the fact that the closer we want to zoom into the complex plane (and/or the closer we are to the boundary between the two regions of attraction), the larger the value of N must be to show sufficient detail.

This article presents a few tips and tricks for improving the speed (decreasing the run-time) of such programs.

The Floating Point Problem

Since a complex number is really just an ordered pair of real numbers, the obvious way to implement calculation of f(z) is with real (i.e., floating point) arithmetic. This is particularly true if one uses the built-in math functions of the computer language the program is written in. Such built-in functions almost invariably use floating-point arithmetic. However, there is an alternative way of doing real arithmetic that is considerably faster on most machines (although it has its limitations - see below). This method is fixed-point arithmetic. It essentially reduces a floating-point computation to one or more integer-arithmetic computations.

For example, suppose one wishes to multiply two real numbers. Further suppose the word size of our machine is 32 bits (i.e., an integer is this size, and we have available a machine instruction to multiply two integers). We can represent a real number within these 32 bits by using the upper 16 bits as the "integer part" and the lower 16 bits as the "fractional part" of the number. Then, to multiply two such numbers, we just multiply the two integers and then shift the result down (toward the LSB) by 16 bits (being careful to do a sign-extend shift). If we only had a 16x16 multiply available (instead of a 32x32), we could still accomplish the same result by using 4 16x16 multiplies and adding up the cross-products appropriately. Such arithmetic is called fixed point because numbers are represented with the "binary point" in a fixed location within the binary word.

Using fixed-point arithmetic can easily speed up image generation by a factor of 3 to 10, depending on the machine and software being used. The drawback is that you cannot zoom into the complex plane as far as you can with floating-point arithmetic, since the precision of fixed-point arithmetic is limited by the relatively small binary word size used. In the above example, for instance, we can only go down to a resolution of 1/65536 (≈.000015) in the complex plane. This is not really much of a limitation for many pictures, however. At the very least, one could use it in a program adaptively: If the zoom factor is moderate, use fixed-point arithmetic to generate the image quickly; otherwise, use full floating-point arithmetic.

Look-up Tables

As well as avoiding floating-point arithmetic, it is always good to avoid computation of transcendental functions. One simple trick that can sometimes be used (depending on the functions involved) is to simply store a pre-calculated lookup table of all possible values of the function (to a certain precision). This works particularly well for trigonometric functions or other functions that are periodic. Only the values within one period of the function need to be pre-calculated and stored.

Simplifying Complex Powers

Many dynamical systems that we wish to draw pictures of involve an f(z) that is (or includes) a polynomial function. For example, the classic Mandelbrot set is f(z) = z**2 + u. On many machines (practically all microprocessors, and some bigger machines), a multiply takes much longer than an add; therefore, the time to calculate the polynomial is approximately equal to the number of multiplies.

Suppose we have two complex numbers z = a + bi, and w = c + di. Then z*w = (ac - bd) + (bc + ad)i. Thus, one complex multiply is equivalent to 4 real multiplies. However, in computing a power, we are not multiplying arbitrary numbers; we are multiplying the same number by itself! Thus, since z**2 = (a**2 - b**2) + (2ab)i, we can reduce this from 4 multiplies to 3. But, factoring, we can get

z**2 = (a-b)*(a+b) + (ab + ab)i

which only requires 2 multiplies! This gives us a factor-of-two speed-up over the simple method of using a general complex-multiply routine.

This also works for higher powers. Straightforward calculation of z3 requires 8 multiplies, but by algebraic manipulation we can obtain the expression

z**3 = a((a**2 - b**2) - b**2 - b**2) + b((a**2 - b**2) + a**2 + a**2)i

which only requires 4 multiplies: compute a**2 and b**2 once (2 multiplies), then add up the terms in the brackets, then multiply by a and b. Thus we once again have a factor of two improvement. As an interesting exercise, you might want to see how good you can do with 4th or higher powers. Does the factor of two continue?

Simplifying the Divergence Test

Each time we iterate the function f(z), we have to test for divergence; i.e., is the complex number outside the circle of radius 2? This is equivalent to asking if magnitude(z) > 2, which is in turn equivalent to (a**2 + b**2) > 4. Thus, we have to do two more multiplies. In the case of the Mandelbrot set (f(z) = z**2 + u), where the optimization of the last section reduces the f(z) computation to just 2 multiplies, this effectively doubles the time per iteration.

If we only care about the image of the zero attractor (the Mandelbrot Set itself, for instance), then we can eliminate these two multiplies by instead testing for (|a| + |b|) > 4. If a point is outside a square of radius 2, then it is certainly outside the inscribed circle of radius 2. Thus, points will still be correctly classified as to divergence or convergence.

Even when creating traditional images where points outside the Mandelbrot set are colored according to velocity, this approximation is not unacceptable. The affect that it has is to add circular arcs (sort of "epicycles") to the contour lines separating the regions. The visual effect is interesting. Try it; you may like it!

Order and Disorder in Computer Art - by William Benzon

Dr. William Benzon is a writer, artist, and musician. He is currently working on a popular book (**Visualization**) *on computer graphics and scientific imaging for Harry Abrams, Inc. and an article on "Visual Thinking" for the Encyclopedia of Computer Science and Technology (Marcel Dekker). He has published articles on cognitive networks and literary semantics. An essay on "The Principles and Development of Natural Intelligence", which he wrote with David Hays, is scheduled for publication in the Journal of Social and Biological Structures. His address is: William L. Benzon, 161 Second St, Troy NY 12180.*

It is becoming increasing clear that artists should use and study the myriad images from both the scientific and mathematical world — not just dynamic systems images, or cellular automata, but the whole range of images such as electron micrographs, molecular structures, galaxies, etc. In my opinion, the art world has been somewhat stagnant since the middle 60s. I am uncertain on how to address the problem, but certainly new graphic reservoirs will help. Science is an obvious source of such material. And computer graphics is an obvious route to such images, especially since computer graphics is gradually becoming recognized as a legitimate art medium.

In particular, it is culturally important for artists to begin to use scientific images. This is a way of beginning to close the gap between science and technology, and society. Thus I think it very important that people in computer graphics attempt to make their images available to artists. Much of the computer graphics produced purely as science or engineering has more aesthetic potential than the computer graphics produced as art.

The figures which accompany this article were created with MacPaint running on an ancient 128K Macintosh with Imagewriter output at 72 dpi. I have tried to use the computer medium without trying to imitate the work I have done in traditional media (primarily ink drawings and paintings done with oils or acrylics). On the negative side, the image is limited to a relatively coarse 8 inch by 10 inch grid of black (actually, dark gray) dots or white dots, no gray scale. When one considers the range of line, shade and texture possible with pen, brush, and black ink, this limitation is very severe. On the positive side, there are two points: 1) the drawing and image manipulation tools which MacPaint provides, and 2) memory, which allows one to save the image at any point in its development. *I find these two advantages so stimulating that I no longer worry much about the limitations imposed by the Imagewriter output.*

All art involves tension between order and disorder. If there is too much order, then the result is decorative pattern. If there is too much disorder, then the result is, well, . . . there just isn't anything you can grab or see. Possibilities for both order and disorder are inherent in MacPaint. The relatively coarse raster pattern which is the basis of Macintosh graphics imposes a relentless rectilinearity on the image space and on many of the MacPaint drawing tools. That is one source of order. Another source of order comes from the editing tools, which make it very easily precisely to copy and repeat image fragments.

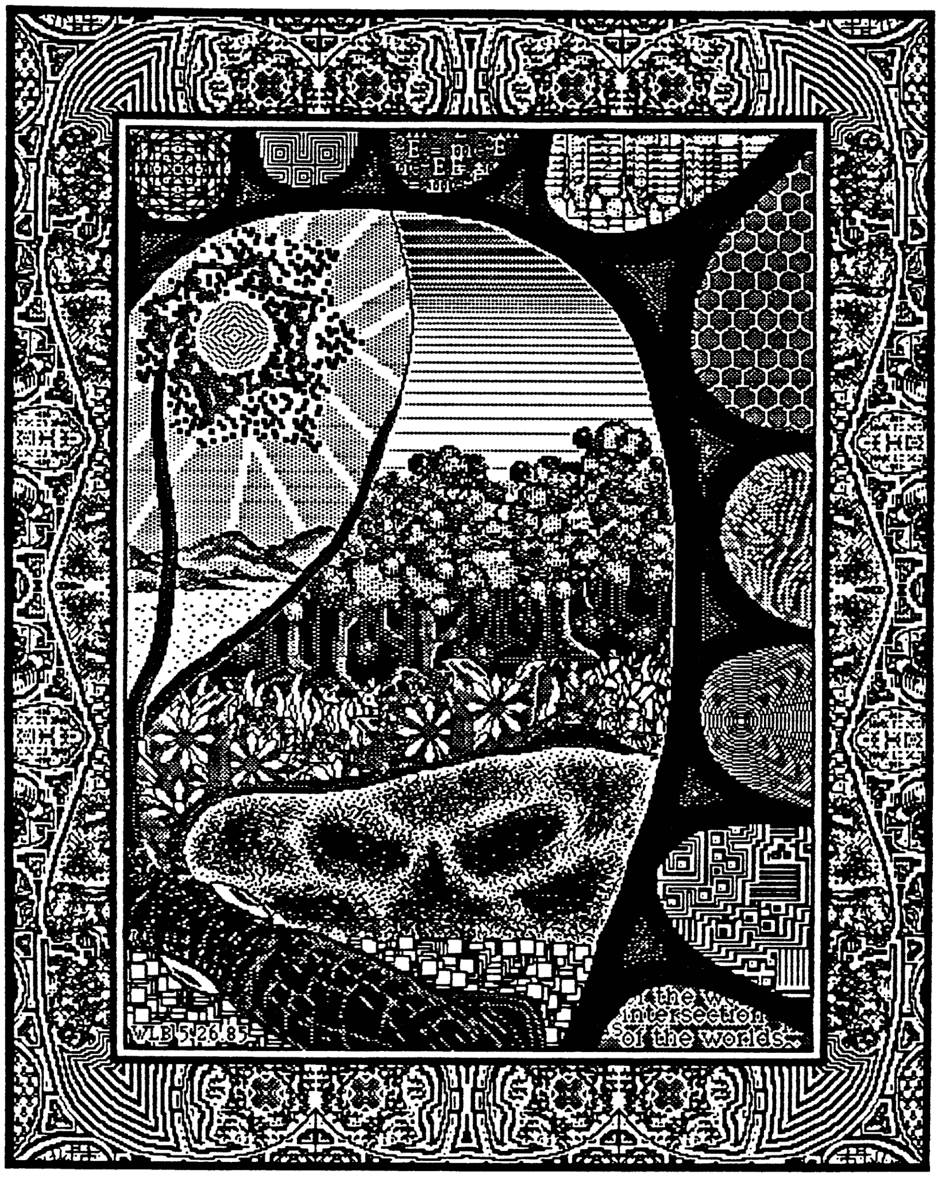
The major source of disorder comes from the fact that the various MacPaint tools can be used rapidly. For drawing tools, this means that you can create lines and forms so rapidly that you can't control precise placement. For manipulation and editing tools, this means that you don't really know how a specific operation will modify an image until you execute the operation. Thus I have found it easy to adopt an operating style which allows the image to get away from me, to surprise me. I use such a style to create textures and abstract patterns.

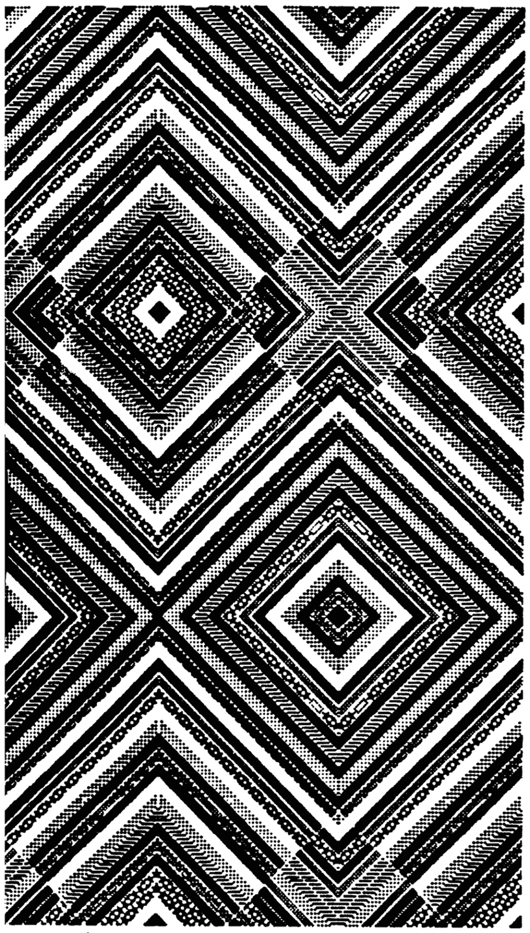
Finally, computer painting tools have changed my notion of what art is, or can be, about. With traditional media there is a clear distinction between the creative process and the final result. You start with a blank piece of paper, or canvas, and work on the image until it is done. Determining just when it is done is often tricky, but, once that decision is made, there is a clear distinction between all of the intermediate images and the final image. The intermediate images have all been subsumed in the final image, which is all that remains. This is not true for all of work I have been doing with MacPaint. While I began saving intermediate versions as a precautionary measure, I soon realized that many of the intermediates were as interesting as the final versions. The distinction between intermediate and final became blurred and I began thinking in terms of exploring visual spaces, each characterized by certain kinds of images created by certain techniques of using MacPaint tools. Each image is simply one example of what exists in a particular visual world, with no particular distinction between intermediate and final images. Thus the images tend to become the means of indicating a certain creative process rather than the process being a way of creating a individual images.

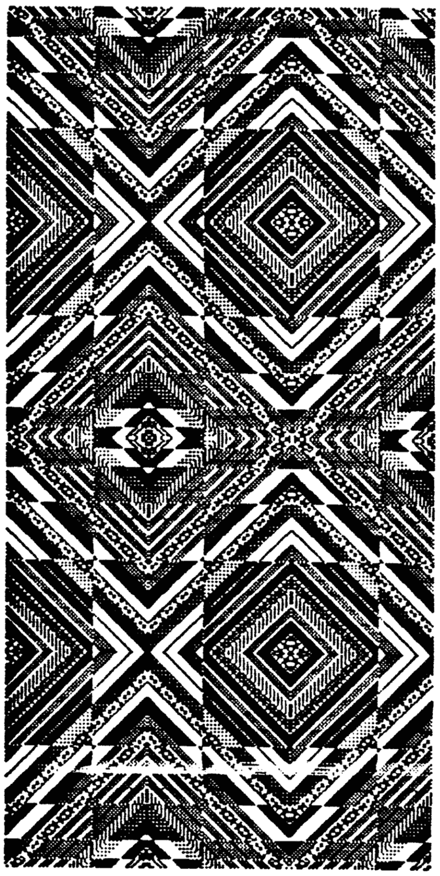
The Intersection of the Worlds: This picture is a rendering of an image I originally created in oils. That painting is brightly colored, relatively untextured, and measures 2 feet by 3 feet, considerably larger than the MacPaint image. For the MacPaint version I used just about every technique I knew. The frame is developed from the a half-size version of the picture itself (this frame is not a part of the original oil painting). This is most easily seen by looking at the upper righthand corner. The large curve stretching around the corner is simply the arch which dominates the main picture space. By starting from this arch you should be able to match motifs in the frame with their sources in the picture.

What's are Whose Best Friend? For both images I started by first filling the screen with widely-spaced diagonal lines ("order"). A few lines perpendicular to the diagonals were then drawn across the space. Then I rapidly, and in no particular order, filled these narrow spaces with various patterns ("disorder"). Next, I used an editing tool to cut a rectangular piece from the pattern. I produced the each final image by copying the seed and rotating the copies and joining them to create an overall pattern with both horizontal and vertical symmetries ("order").

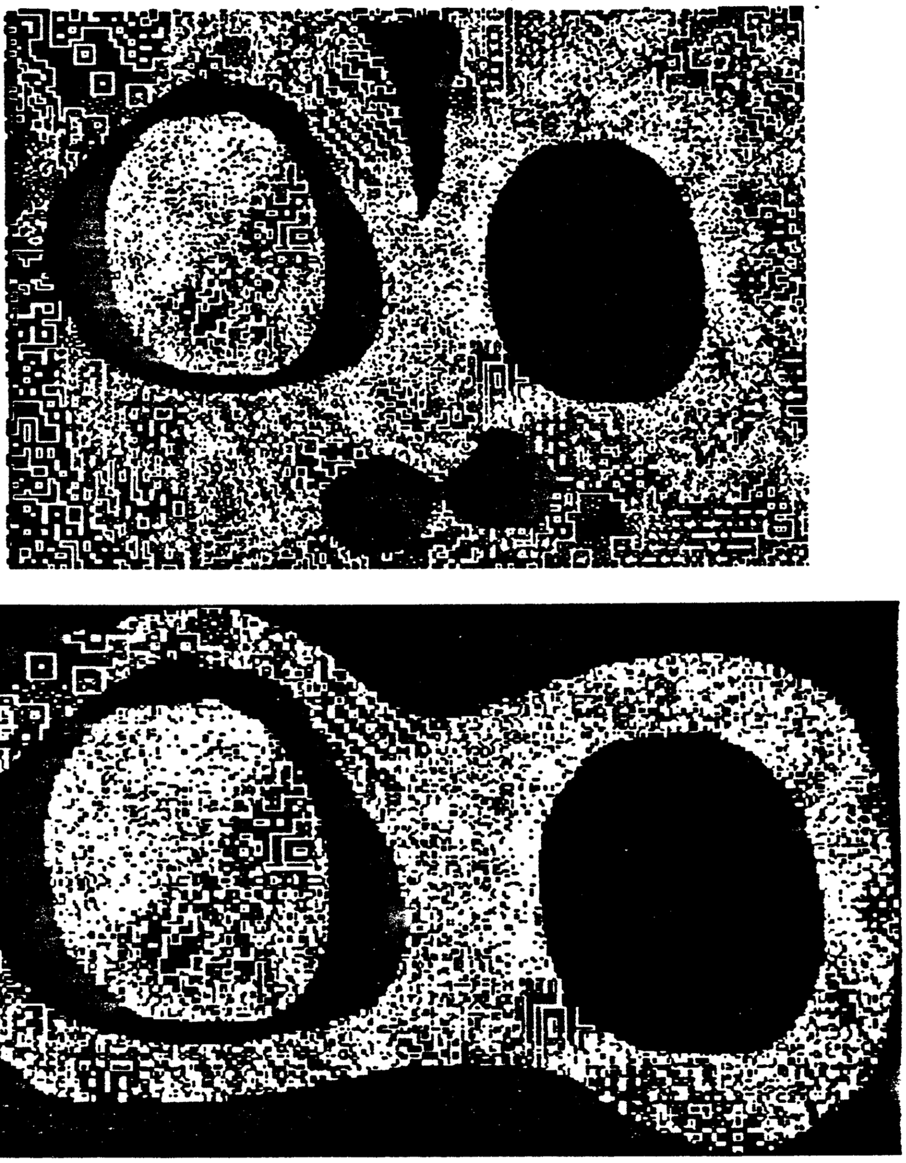
Deathlock the Demolisher: I created the background by first drawing a highly rectilinear pattern of small figures. This is easy to do, but hard to describe. It involves the rapid use of 3 or 4 MacPaint tools. Then I went over this pattern with some spray paint. Finally I painted the totally black areas. The lower image is a slightly enlarged version of the material used in the upper image. Deathlock was the title character of a short-lived comic book which Marvel Comics published in the middle 70s.


WLB 5.26.85
the w
intersection
of the worlds





What's are Whose Best Friend?
WLB • Spring 1985



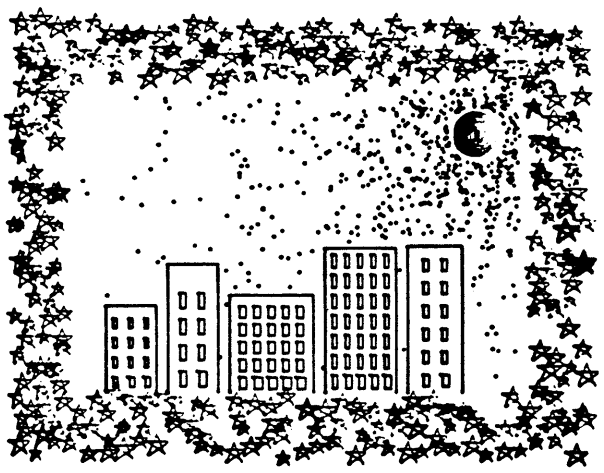
Deathlock, the Demolisher • VLB • Spring 1985

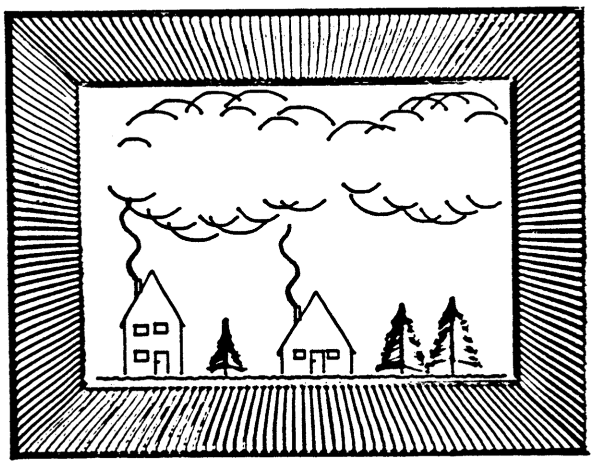
Simple Structures - by Russell Hoffman

Russell Hoffman is interested in computer graphics and algorithms. He can be reached at: P11 Enterprises, PO Box 5185, Bridgeport, CT 06610.

I have concentrated on algorithms which utilize certain basic primitive patterns. They also allow the computer to randomly pick colors, positions, and sizes of objects so that each resultant drawing is different. Below are several samples. I have let the computer produce 15,000 of these to date. The pictures are drawn directly by a Panasonic VP6801-P30 digital plotter driven by an NEC 8201A Laptop portable using its ROM BASIC. Ranges for the random numbers controlling various aspects of the pictures can be entered. There are no complex mathematical formulas involved, but the basic idea is useful in industry. For example, housing developments could contain different and unique houses, if designed with computers using programs that incorporate this controlled randomness.

Readers may write to me about software to create these forms, as well as others such as a galloping horse logo. Free samples of these kinds of drawings, in color, are available. The program, P11, is used in hospitals, factories, laboratories, and by consultants and hobbyists to write animated educational tutorials.

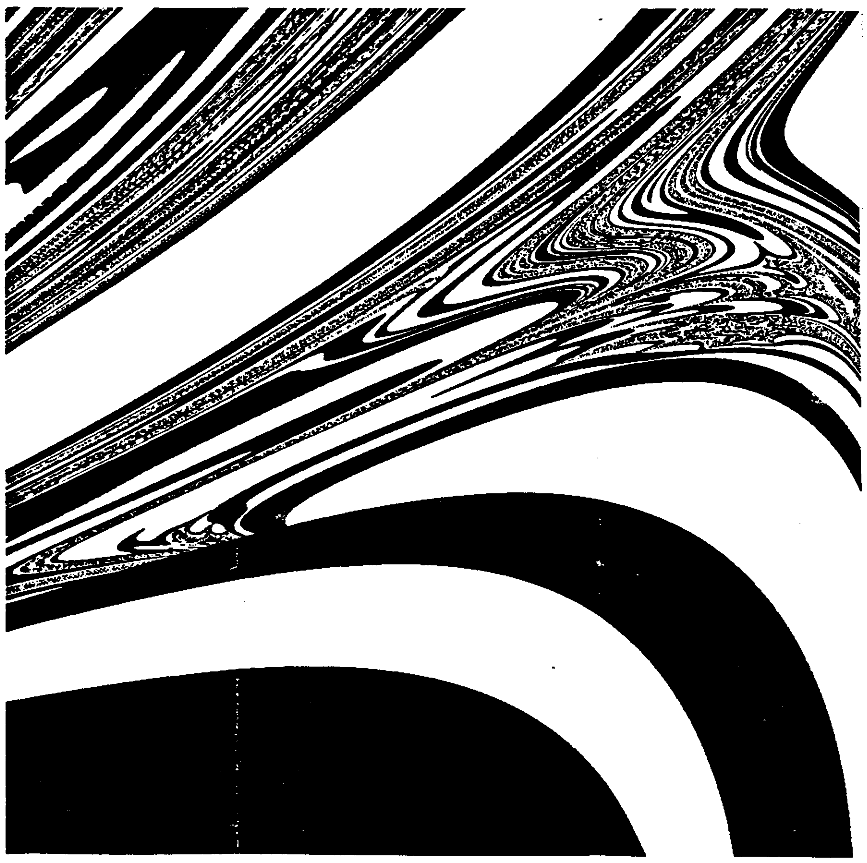




Iteration Maps - by David Scruton

David Scruton can be reached at 7818 Manchester Ave 6, Venice CA 90291.

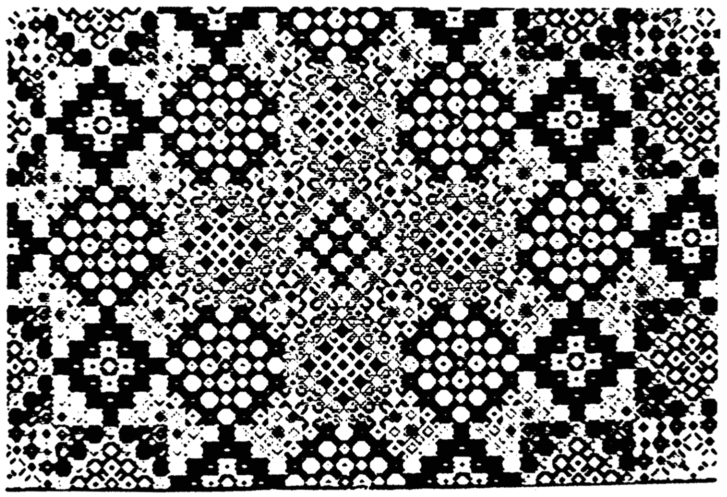
This short note presents an image derived from complex dynamics (see also the article in this issue, "Logical Chaos"). In particular, the picture below of a complex variable recursion was produced by iterating $z = z^3 + \mu$ forty times. Diverging points whose magnitude is between 2 and 3 are plotted. The picture was computed on a μ VAX and plotted on a Talaris laser printer.

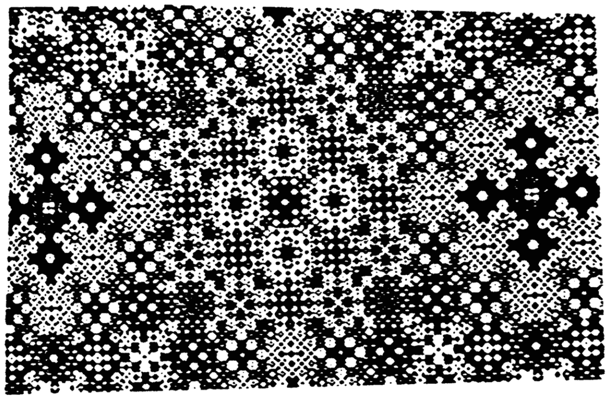


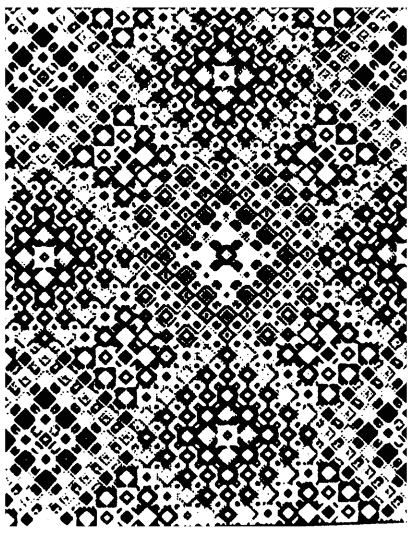
Cellular Patterns - by Rob Cave

Rob Cave is interested in computer graphics and algorithms. He can be reached at: PO Box 928, Princeton, TX 75077.

The following pictures were created on a Tandy 1000SX and CM-11 color monitor (original pictures were color). The algorithms use eight nested FOR loops and were coded in GW-BASIC. Note that every time the program runs, different patterns are generated as a result of a random number generator. For more information on the algorithms used, contact the author.







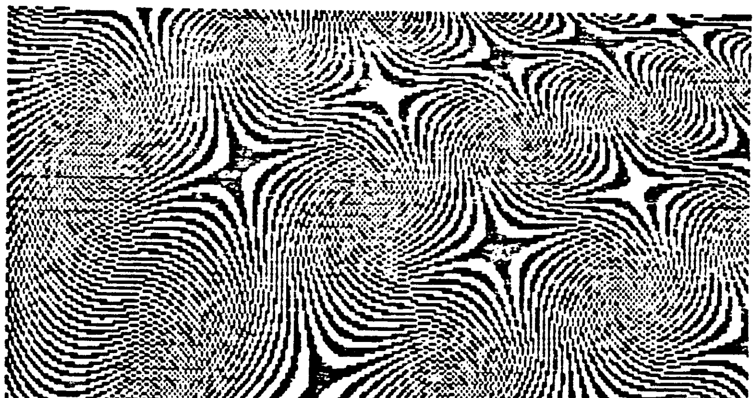
Complex Curves from Simple Math - by Rastislav Telgarsky

Rastislav Telgarsky is a Visiting Professor in the Department of Mathematical Sciences at the University of Texas, El Paso, TX 79968. He has worked in the area of topology and game theory. His interests include recursive curves, fractals, mosaics, and other graphics patterns arising in mathematics and art.

Below is pseudocode which creates complicated structures from quite simple formula. I hope the readers of JCG will find them of interest.

```
Program Graph;
VAR i,j integer;
Function c(a,b,:real):integer;
Begin
 c:= round(a*sqrt(a*b));
End;
Begin
  GraphMode; Palette(2);
  For i := 1 to 150 Do
   For j := 1 to 150 Do
    Plot(70+i,30+j,1+ (c(5+i/7,5+j/7) mod 3));
  END;
 END;
END;
```

The 70+i and 30+j terms in the plot function can be adjusted to suit the users screen size. The third term of the Plot function controls the color. I had only four colors including black (hence mod 3), however the reader may use other settings, for example mod 2 for black and white or mod 16 for 16 colors.



Analytic Computer Art - by Joe Jacobson

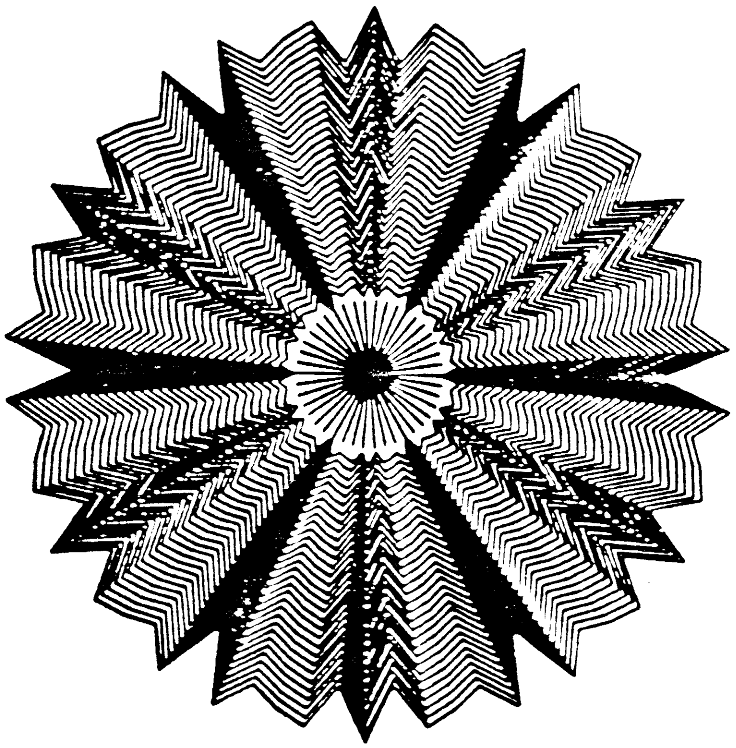
Joe Jacobson has written numerous articles for "Creative Computing". His interests include applied physics and computer art. He has a BSEE degree from Drexel University. He can be reached at: Apt. 1009, 675 E. Street Rd., Warminster, PA 18974.

This article describes a facet of "analytic computer art" consisting of geometrical designs based on explicit mathematical functions. A common motif in analytical computer art is the polar coordinate curve. This has the form R=f(A) where R is radius, f is a mathematical function, and A is the angle. The angle parameter, A, is swept through some range of values, the radius R is calculated, and the computer polar coordinate points (R,A) are converted to rectangular coordinates and plotted. The resulting curves frequently (but not always) exhibit angular symmetry; that is, they look the same after being rotated through a suitable angles. What's special here is that simple polar coordinate curves are swept through the interval 0 to 360 degrees and incremented by a fixed amount between sweeps. The pictures were generated on a Tektronix 4052 intelligent terminal.

For Further Reading

J. Jacobson, "Analytic computer art". Proc. 2nd Symp. on Small Computers in the Arts, pp 47-60 (1982).

```
09 REM FLUTED SCALLOPS
100 PAGE
109 SET DEGREES
110 WINDOW -501,501,-501,501
111 VIEWPORT 15,115,0,100
112 PAGE
113 PRINT "ENTER L"
114 INPUT L
115 PAGE
120 FOR B=100 TO 400 STEP 10
130 FOR A=0 TO 360 STEP 5
140 GOSUB 180
145 IF A>0 THEN 150
146 MOVE X,Y
147 GO TO 160
150 DRAW X,Y
160 NEXT A
170 NEXT B
175 GO TO 211
180 R=B*(1+0.25*ABS(SIN(L*A)))
190 X=R*COS(A)
200 Y=R*SIN(A)
210 RETURN
211 FOR N=0 TO L-1 STEP 1
212 R=100
213 T=N*(180/L)
214 X=R*COS(T)
215 Y=R*SIN(T)
216 MOVE X,Y
217 X=R*COS(T+180)
218 Y=R*SIN(T+180)
219 DRAW X,Y
220 NEXT N
221 END
```



A Noah's Ark Program - by Rudy Rucker

Dr. Rudy Rucker is author of **Infinity and the Mind** *(Bantam: New York, 1982). He is also the author of The 57th Franz Kafka, Software, White Light, and Spacetime Donuts (published by Ace Books). He can be reached at the Dept. of Mathematics and Computer Science, San Jose Sate University 95192.*

Today, there are several scientific fields devoted to the study of how complicated behavior can arise in systems from simple rules and how minute changes in the input of a nonlinear system can lead to large differences in the output; such fields include chaos and cellular automata theory. "Cellular automata" are a class of simple mathematical systems which are becoming important as models for a variety of physical processes. Usually cellular automata consist of a grid of cells -- and the cells' life or death is determined by a mathematical analysis of the occupancy of neighbor cells. One popular set or rules is set forth in what has become known as the game of "LIFE". Though the rules governing the creation of cellular automata are simple, the patterns they produce are very complicated and sometimes seem almost random, like a turbulent fluid flow or the output of a cryptographic system.

The figure on the next page is is a screen dump of some output from a simple assembly language program which runs one-dimensional cellular automata.

The rule depicted is what is called Rule 46 according to the notation in the appendices of Steven Wolfram (*Theory and Applications of Cellular Automata*). Instead of using graphics capability, my program produces images "typographically", using blanks for zeros and solid squares (ASCII code DBh) for ones. The patter starts with a line of zeros with a single one.

In general, a r-2, n-2,1-D CA pattern like this is updated according to a rule where a cell C looks at its let neighbor L and right neighbor R to get a three -digit binary number LCR. LCR can range through the eight values v from 000 to 111. The rule depicted is based on the lookup table 00101110, where the update for value v is the vth lookup value from the right. In decimal, the lookup table is the number 46.

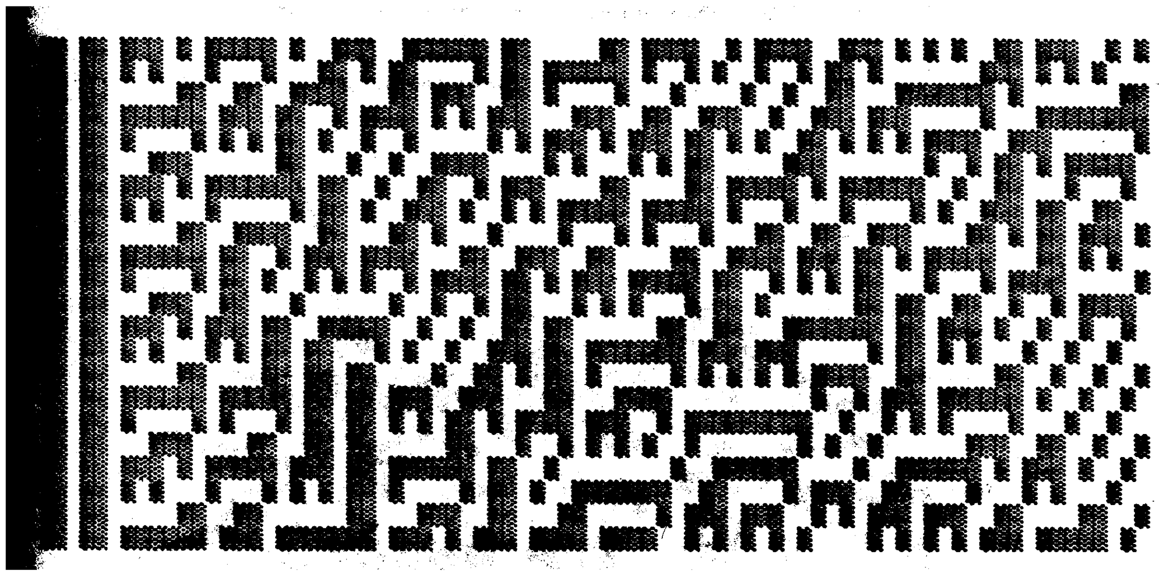
What makes this picture interesting is the handling of the boundary conditions. As is customary, we use "cyclic boundary conditions", meaning that the rightmost cell is regarded as the cell left of the leftmost cell. Bun in this run, I set the leftmost cell always to 0. In effect, the space is like a tin-can that has a seam running down it.

The seam acts as a generator that pulses out alternating leopards and elephants. The neat thing is that these animals then shuffle and mutate to produce giraffes, dinosaurs, etc.

For Further Reading

1. Peterson, I. (1987) Forest fires, barnacles, and trickling oil. Science News. 132: 220-221.
2. Poundstone, W. (1985) The Recursive Universe. William Morrow and Company, New York.
3. Wolfram, S. (1983) Statistical mechanics of cellular automata. Rev. Modern Physics. 55, 601-644.

Output of the Noah's Ark Program



```
C:\ASSEMBLY >
C:\ASSEMBLY >                    Noah's Ark                    Rudy Rucker 87
```

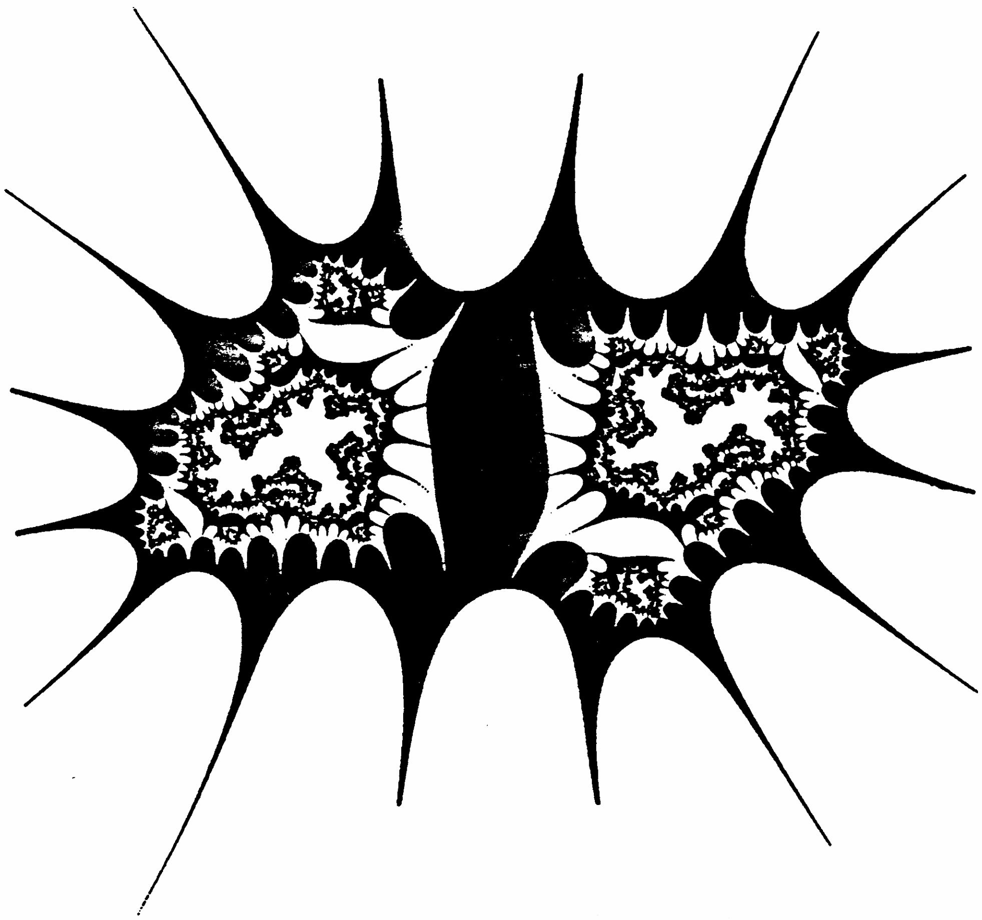
Iteration of Algebraic Transformations - by Ian Entwistle

Ian Entwistle is a master of high-resolution fractal images. He can be reached at 44 Woodside Gardens, Stittingbourne, Kent, England.

To create the figure in this article, I start with an array of complex values (z) and have the computer follow the outcome of the process defined by $z = (z^2 + \mu)^2 + \mu$ and then checking to see if $z(real) < 0$. $\mu = (0.9, -0.1)$. The picture boundaries are $(-2,2,-2,2)$. The resolution is 1150 x 1484 pixels. Once the initial points are selected, each iteration represents a step along a path that hops from one complex number z to the next. The collection of all such points along a path constitutes an orbit. The basic goal is to understand the ultimate fate of all orbits for a given system.

For Further Reading

Peitgen, H. and Richter, P. (1986) *The Beauty of Fractals*. Springer: Berlin.; Mandelbrot, B. B., *The Fractal Geometry of Nature*, Freeman, San Francisco (1983).



Is Music Fractal? - by Stephanie Schneider

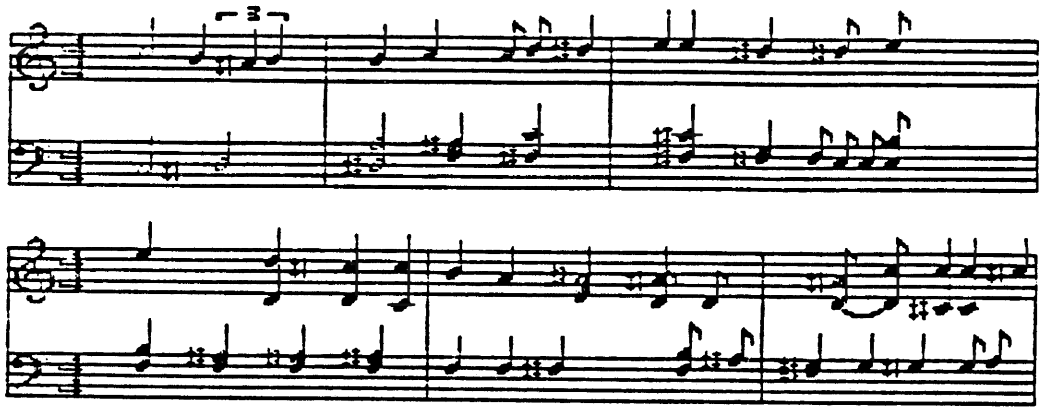
Stephanie Schneider is a high school student and can be reached at 80-17 264th St. Floral Park, N.Y. 11004.

Fractals are irregular shapes with interesting visual and mathematical properties. In my research, music was converted to fractal curves and fractal curves were converted to music.

Rather than going into detail here, I simply summarize the results of my work and welcome questions from interested readers. Graphs were constructed by plotting the duration of the note on the x axis and the pitch on the y axis. Fractal curves, especially those resembling simple landscapes, were found in the treble portions of nocturnes and in the bass portions of ragtimes (no deep mathematical characterizations were done -- I used only visual inspection). Also, landscape fractal curves can be mapped into pleasing music, with the melodic lines of a nocturn and the syncopation of a ragtime.

For further reading

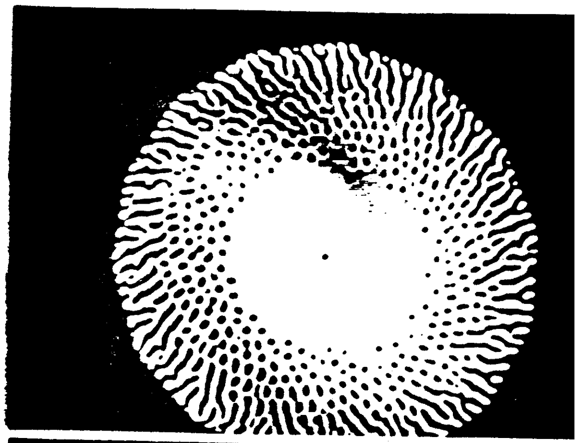
1. Peterson, I. Making music fractally, Science News, March, 117(12):187-190.
2. Pierce, J. (1983) *The Science of Musical Sound.* Scien. Amer. Library: New York.

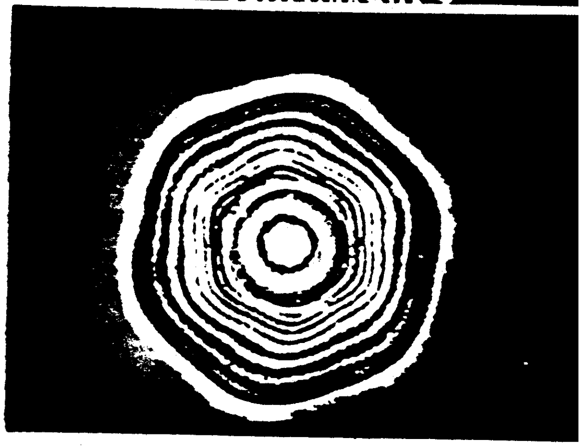


Video and Chaos - by Gustavo Chaux

Gustavo Chaux's main activity is in community development work and alternate knowledge tools. Gustavo Chaux can be reached at Gustavo Wilches - Chaux, Apartado Aero 1280, Popayan - Cauca, Columbia, South America.

The figures below are excerpted from a video entitled "Territorios" (Territories) using chaos-based procedures. The original figures are in vivid color. Those interested in learning more about the formation of the structures in these examples may contact the author.

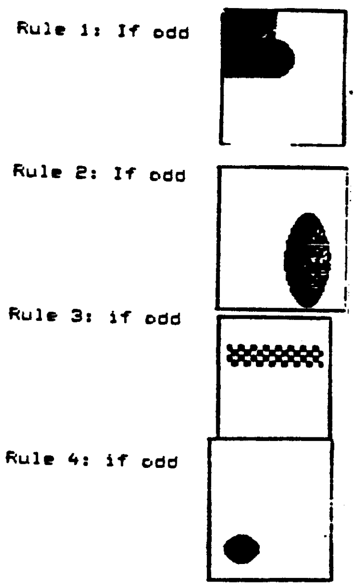


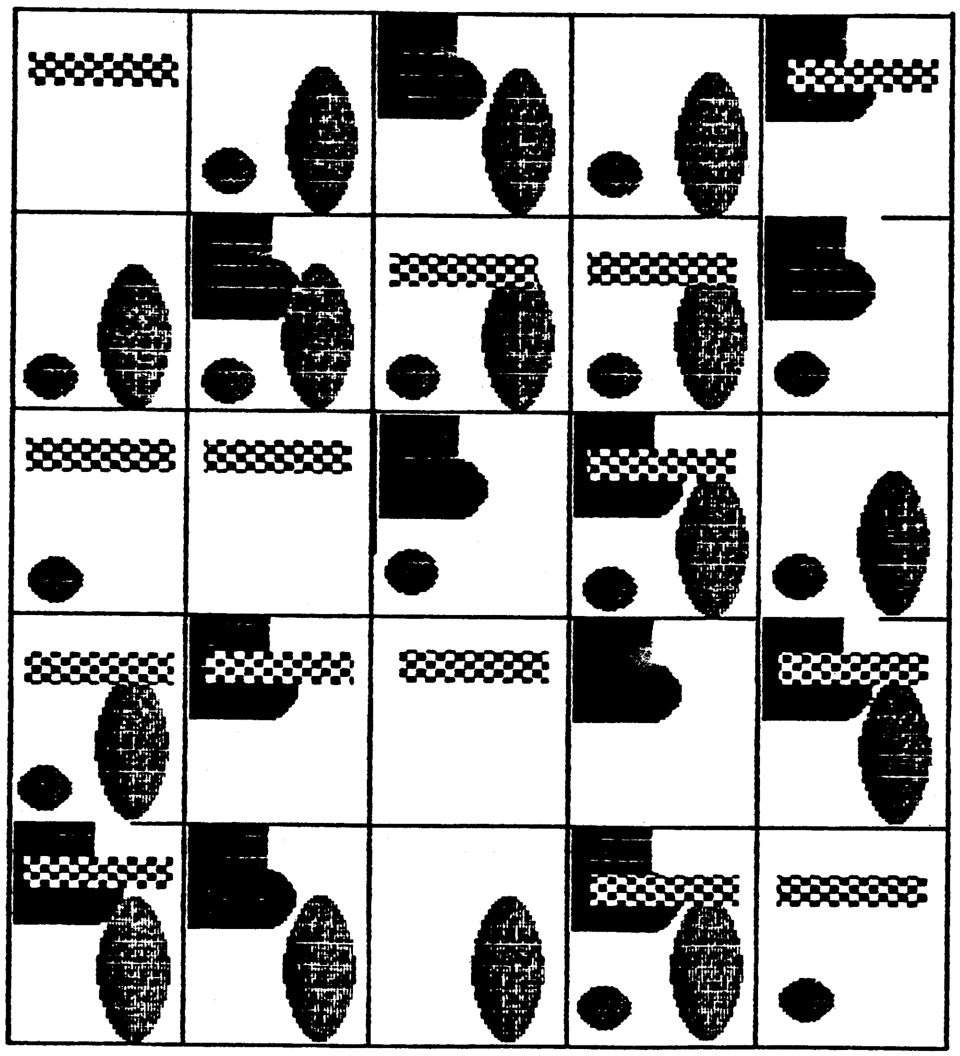


Randomness and Design - by Michael Eckersley

Michael Eckersely is an assistant professor in the Department of Design at the University of Maryland. His primary interest is the psychological study of design behavior. Professor Eckersley can be reached at the University of Maryland, College Park Campus, Dept of Housing and Design, Room 1401, Marie Mount Hall, College Park, Maryland 20742.

There exists among some artists and design theoreticians a particular fascination with random process as a kind of respite from formally contrived structures. Some aspects of my work examines the modern concept of randomness, and its potential for creative use in foundation design instruction. Students might understand the nature of formal design principles (e.g. harmony, balance, variety, structure...) by means of an introduction to the theory and process of randomization. For color slides and additional information, contact the author. Below are rules for the relatively simple design shown.





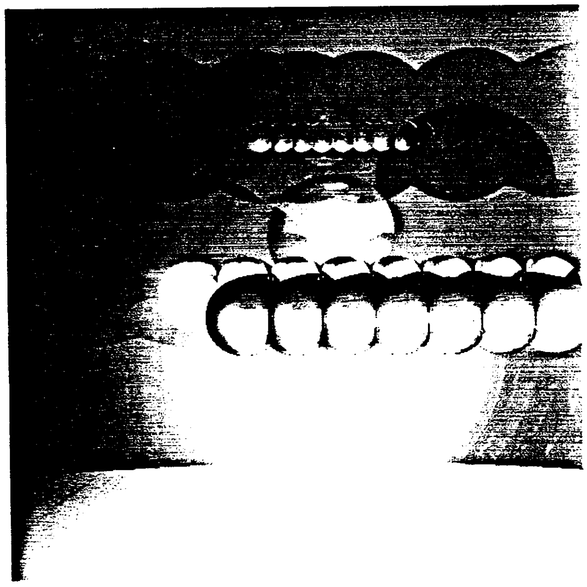
Ray Tracing - by Art Stein

Art Stein can be reached at the IBM Watson Research Lab, Yorktown Hts, NY 10598.

Creating realistic images on a computer requires simulating the physics of illumination on every object in the scene. *Ray tracing* in computer graphics allows the computer to generate a believable image given a collection of three-dimensional objects and a collection of light sources. Below is an example of a ray-traced picture, and the reader is referred to the references for more information on this type of approach.

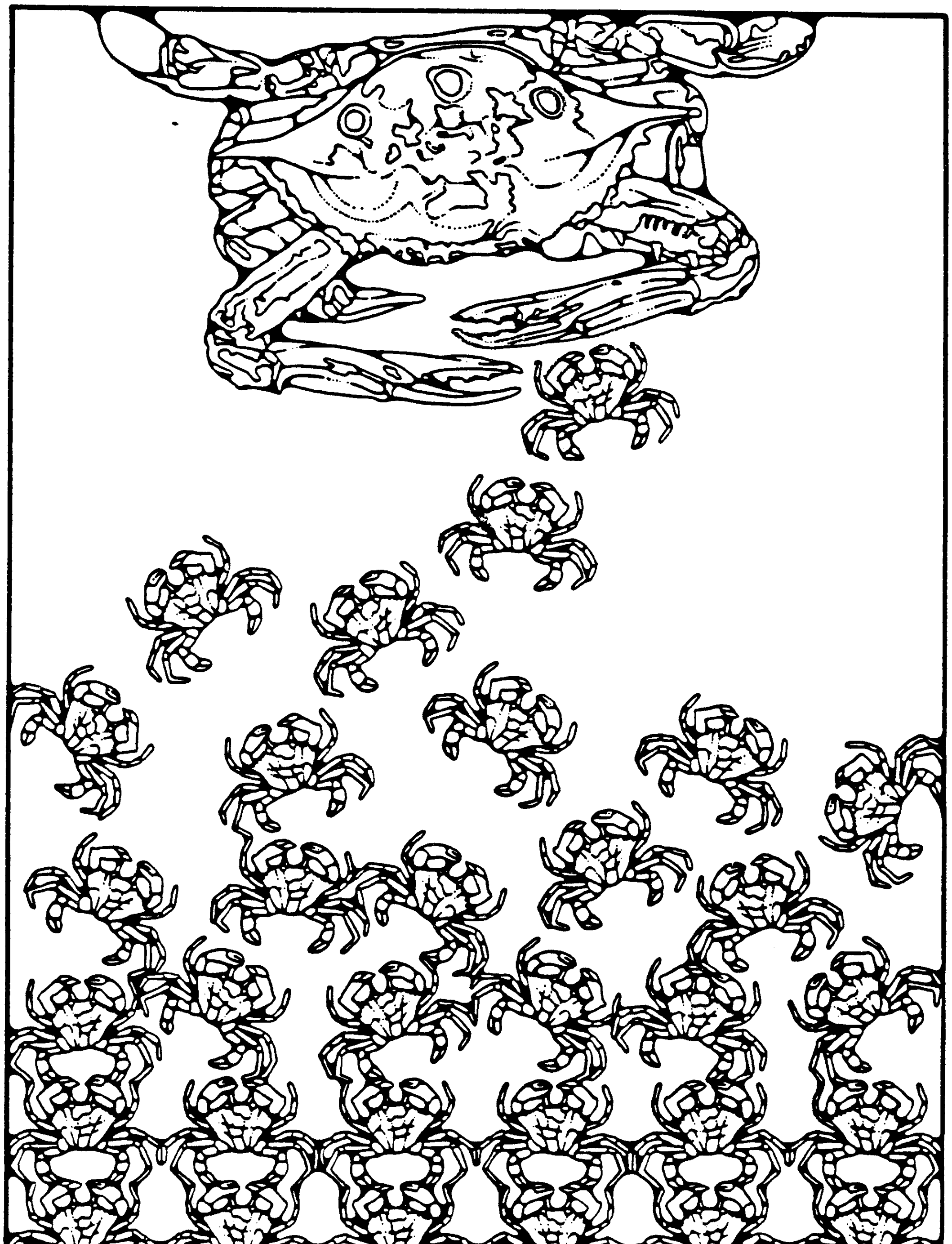
For Further Reading

1. Glassner, A. (1987) Ray tracing in computer graphics. *Computers in Science.* Sept./Oct. 18-25.
2. Heckbert, P. (1988) Ray tracing Jello-O brand gelatin. *Commun. ACM.* 32(2):131-135.



Order and Disorder in Art - by William Rowe

William Rowe is a master of pen and ink drawings which often contain ordered arrays with small disordered perturbations. This drawing is from his book "Flora and Fauna Design Fantasies" (Dover Publications). His book often displays nature's life forms in startling combinations: giant moths under the moon, repeating patterns of bats, fish spines with flowers ...



Cellular Automata Machines - Reviewed by David Chess

Dave Chess is a programmer in the Advanced Workstation Projects group at Yorktown Research, working on host-workstation cooperative applications. His interests include computer mediated communications (he ran the internal IBMPC conference for five years), philosophy (he has an AB degree in Philosophy, Princeton 1981), fractals, cellular automata, and computer-resident realities in general. He has a cellular-automaton package on PCTOOLS (DCCELAUT PACKAGE), and some fractal landscapes on IBMPIX. Mr. Chess can be reached at IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

Cellular Automata Machines: A New Environment for Modelling by Tommaso Toffoli and Norman Margolus MIT Press (Cambridge, Mass), 1987 250 or so pages

This book from MIT Press, should be of considerable interest to folks working in the cellular-automaton field (especially to those of us who have dreamed of having a box that could run interesting two-dimensional automata in real-time!), and to people in the physics community who have heard about cellular-automaton models, and want some concrete examples.

The book has two primary, interrelated, themes. The more theoretical one concerns various interesting classes of cellular automata, and how they may be used to model in significant ways various physical phenomena, including gas diffusion, growth processes, the heat equation, Ising models of magnetization and phase transitions and such, various noise phenomena, sound waves, the hydrodynamics of flow, spin glasses, and so forth; as well as interesting processes related to computation (especially in the area of energy-loss and information processing).

The other theme, which is used throughout to give actual examples of interesting automata, concerns a family of highly-parallel special-purpose processors which the authors have developed (are developing), called "Cellular Automata Machines" or CAMs. A CAM is specifically designed to allow extremely fast execution of reasonable-sized worlds running under user-specified cell-updating rules. The examples in the book are programs (in a variant of the FORTH language) written for the CAM6, a device of which several actually exist.

The book is basically a practical introduction and guide to using the CAM6 and cellular automata to model various computational and physical processes. The emphasis is on programs rather than equations, and implementation rather than theory (although theory is by no means ignored, and there is a differential equation or two; but readers not fond of diffeq's can skip them without much loss). Even without ready access to a CAM of one's own, the book is very interesting reading, and many of the concepts are sufficiently general to allow implementation and exploration on boring sequential computers.

I liked the book, and would recommended it to anyone with enough slack in the personal or departmental budget (it's a hardcover from MIT Press, and I think it was $30 or so). There are fifteen color plates of automata states, and good halftone black-and-white illustrations throughout. For those unfamiliar with FORTH, there is a good small tutorial in an appendix that should enable anyone to read the examples in the text. Hardware fans will also find an appendix on the CAM architecture; I'm not enough of a hardware person to tell how complete or informative it is! My only criticisms, and they are mild, are that I would have liked more color plates, and that the book occasionally discusses CAM6-specific issues in more detail than CAMless readers might want. This probably makes it of more use, on the other hand, to readers with CAMs!

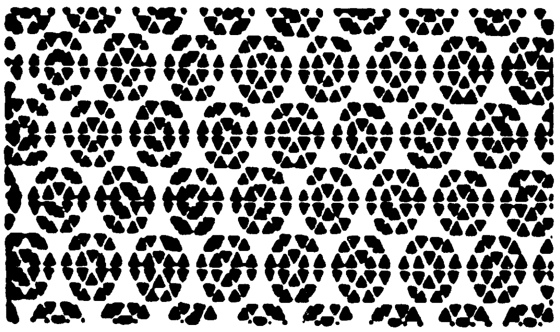
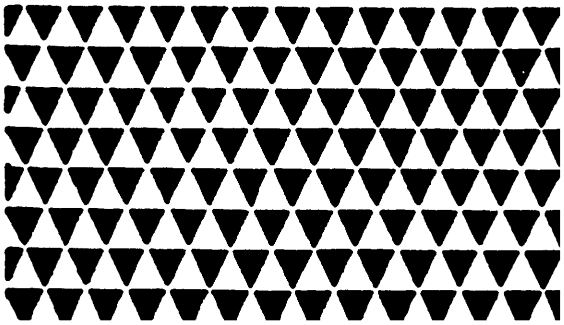
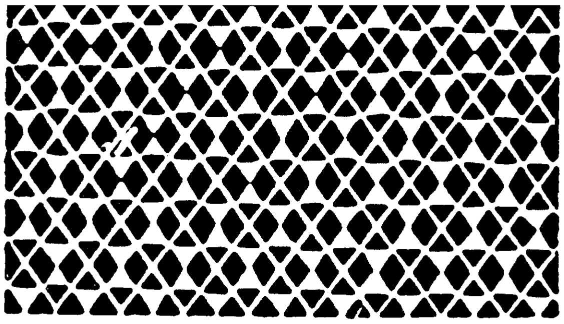
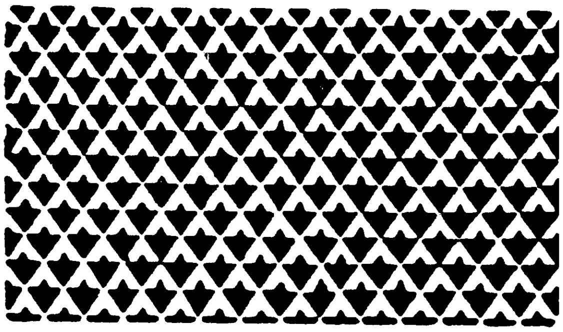
CAM6 itself is a board that plugs into an IBM PC, and communicates via the PC's keyboard and displays. It supports a 256x256-cell world (or, in some sense, four different 256x256-cell worlds), and its output is a 256x256 pattern on a standard color display. I've had one demo'd to me at the MIT CS Lab, and it really flies! The standard Conway's LIFE algorithm moves fast enough to make individual cell-events invisible; it normally runs (I believe) one generation per screen-refresh (with provision, of course, for slowing it down at user command). It supports various notions of "neighborhood", and various ways of using its four bit planes to produce cell-update rules. For more details, read the book! One thing that isn't mentioned is how to get a CAM of one's own; to quote the book

"This machine was originally developed at the MIT Laboratory for Computer Science. It is currently produced by SYSTEMS CONCEPTS (San Francisco, CA), from which it was commissioned with the explicit intention that, after fulfilling MIT's internal needs, further output of the production line would be made available to the scientific community at large, as inexpensively as possible."

Last time I talked to any of the CAM people, MIT's internal needs were not yet fulfilled, and there was no firm pricing or availability data, although I'd expect it to be in the $1-2,000 range (this is an order-of-magnitude-type guess).

A closing quote, from the book's introduction:

> "A cellular automata machine is a universe synthesizer. Like an organ, it has keys and stops by which the resources of the instrument can be called into action, combined, and reconfigured. Its color screen is a window through which one can watch the universe that is being 'played'. This book, then, is an introductory harmony and orchestration manual for 'composers' of cellular-automaton universes."

- a section containing short articles on mathematical curiosities, and interesting equations and their graphs.

A. Equation of the Month - Spirals

- a section on unusual equations with interesting behavior, submitted by readers.

Generally, plane curve spirals are of the form

$$r = f(\theta)$$

in polar coordinates (where f is monotonic), and they possess a simple beauty which humans have copied in their arts and tools, and nature has used in the creation of many structures of life. All the mathematical forms presented in this section were first discovered in the seventeenth and eighteenth centuries, except for the simplest form, the *Archimedes spiral* which was first discussed by Archimedes in 225 BC. The *Archimedes spiral* is expressed by the equation

$$r = a\theta$$

The most commonly observed spirals are of the Archimedian type: tightly wound springs, edges of rolled-up rugs and sheets of paper, and decorative spirals on jewelry. Practical uses of the Archimedes spiral include the transformation of rotary to linear motion in sewing machines (Gardner, 1969).

The *logarithmic spiral* (also known as the *equiangular spiral* or *Bernoulli spiral*) can be expressed as

$$r = ke^{a\theta}$$

This spiral was first discussed by Descartes in 1638. The angle between the straight line, $\theta =$ constant, and the tangent to the curve is constant. Other more exotic spirals include the *hyperbolic spiral* (or *reciprocal spiral*), which is of the form

$$r = \left(\frac{a}{\theta} \right)$$

A *littus* has the form

$$r^2\theta = a$$

A *Cornu spiral* (or *clothoid* or *Euler's spiral*) has a parametric representation:

$$x = a\sqrt{\pi} \int_0^t \cos\left(\frac{\pi t^2}{2} \right) dt$$

$$y = a\sqrt{\pi} \int_0^t \sin\left(\frac{\pi t^2}{2} \right) dt$$

This curve was discovered by Euler in 1744, and M.A. Cornu later used this curve in the representation of optical diffraction.

Some of these families of spiral curves can perhaps be more simply defined by

$$r^m = a^m\theta$$

which includes the Archimedes spiral ($m = 1$), Fermat's spiral ($m = 2$) (first discussed by Fermat in 1636), the hyperbolic spiral ($m = -1$) (first discussed by Pierre Varignon in 1704) and the littus ($m = -2$) (originated by Cotes in 1722).

The *involute of a circle* with parametric equations

$$x = a(\cos \phi + \phi \sin \phi)$$

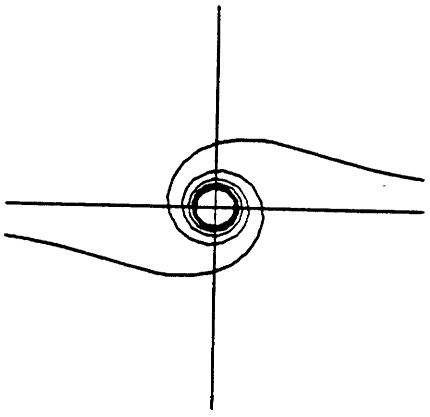
$$y = a(\sin \phi - \phi \cos \phi)$$

was first taken into account by Huygens when he was formulating his ideas for clocks without pendulums which might be of service on seagoing vessels. This is the curve described by the endpoint of a string as it unwinds from a circle of radius a while held taut. The curves traced by all points along the plank of a seesaw or the path of a goat tied to a cylindrical post as it winds tightly around it are both involutes of a circle.

Finally, the *cochleoid* (or snail form) is given by

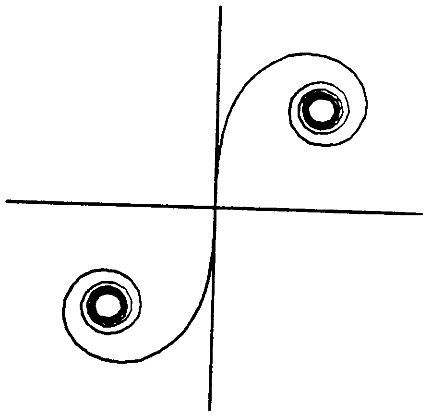
$$r = a\left(\frac{\sin \theta}{\theta} \right)$$

Apart from their mathematic differences, and also the varied natural forms these spirals help to describe, many of these spirals are quite different *visually*. For example, perhaps the most exotic looking of the group is *Euler's spiral*, which consists of two spirals connected together, giving it the appearance of a mustache with two curled ends. The only other spiral of the group with more than one center is the *Cochleoid* which contains two directly adjacent spirals. *Fermat's spiral* is the only member that consists of two concentric lines, and it resembles the paths of two tracks of a stereo record groove. Finally, the *lituus* is the only spiral of the group with a long, almost-linear section; it looks like a fern tendril with a very long stem. For more details, see Pickover, C. (1988) Mathematics and Beauty II: Spirals and "Strange" Spirals in civilization, nature, science, and art., ***Leonardo*** 21(2). Below are both traditional and exotic spirals from mathematics.



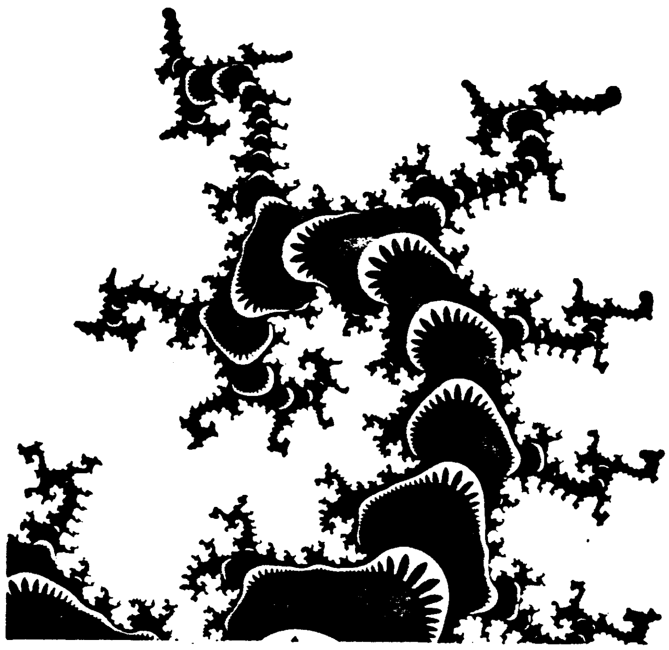
Lituus

$a = 2;\ m = -2$



Euler's Spiral

$a = 1$



B. Armstrong Numbers

One hundred fifty three is interesting because

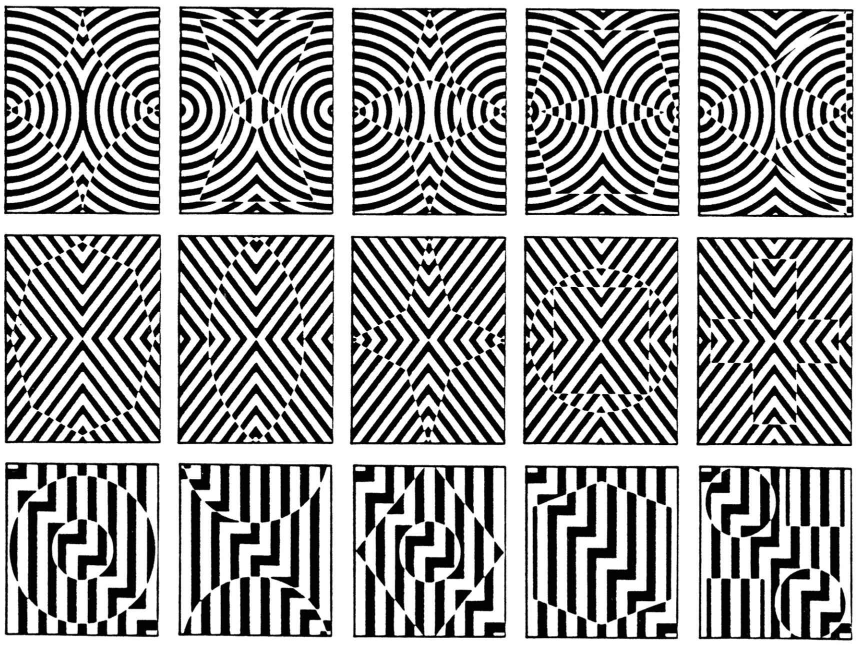
$153 = 1^3 + 5^3 + 3^3$

Numbers such as this are called Armstrong numbers. Any N digit number is an Armstrong number if the sum of the Nth power of the digits is equal to the original number. Another example is 370 (370 = 27 + 343 + 0). The following program prints all three-digit Armstrong numbers:

```
Print all Three-Digit Armstrong Numbers
For n = 100 to 999
  a = int(n/100)
  b = int(n/10)-10*a
  c = n-100*a - 10*b
  if n <> a**3 + b**3 + c**3 then goto next
  print ("Armstrong number",n)
  print ("equals",a**3,"+",b**3,"+",c**3)
next: Next n
End
```

For Further Reading

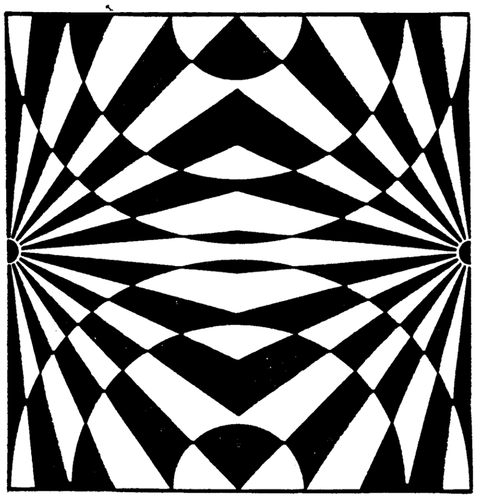
Spencer, D. (1982) Computers in Number Theory. Computer Science Press: Maryland.

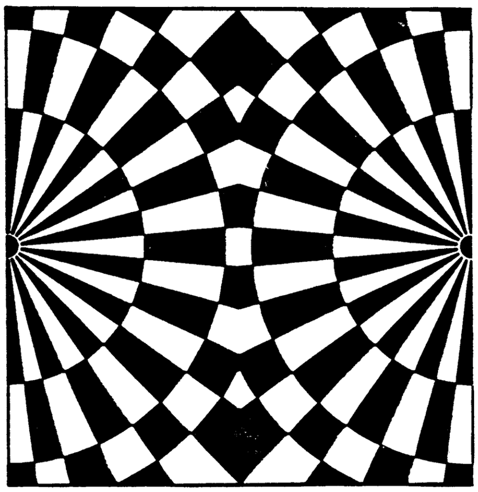


C. References of the Month

- a section on interesting references sent in by readers.

1. AMYGDALA, a fascinating newsletter on the Mandelbrot set. Write to AMYGDALA, Box 219, San Cristobal, NM 87564 for more information.
2. ART MATRIX, creator of postcards of beautiful mathematical shapes. Write to ART MATRIX, PO Box 880, Ithaca, NY for more information.
3. Recreational and Educational Computing Newsletter. Write to Dr. M. Ecker, 129 Carol Drive, Clarks Summit, PA 18411, for more information on this interesting newsletter.
4. The Dynamic Newsletter, Aerial Press, Inc. Santa Cruz, CA 95061-1360. (Ralph Abraham, Ed).
5. Barnsley, M. and Sloan, A. (1987) Chaotic compression (a new twist on fractal theory speeds complex image transmission to video rates). Computer Graphics World. November. 107-108.
6. Braden, B. (1985) Design of an oscillating sprinkler. Mathematics Magazine 58: 29-33.
7. Casey, S. (1987) Formulating fractals. Computer Lang. 4(4): 28-38.
8. Dunham, D., Lindgram, J. and Witte, D. (1981) Creating repeating hyperbolic patterns. ACM SIGGRAPH Computer Graphics 15(3): 215-220.
9. Donnini, R. (1986) The visualization of music: symmetry and asymmetry. Comp. and Maths. with Appls. 12B: 435-463.
10. Pickover, C. (1988) Pattern formation and chaos in networks, *Communications of the ACM* , February 31(2), 136-151.





Final Light Thoughts - Carol Grafton

This month's "Final Light Thoughts" are from Carol Grafton's book Bizarre and Ornamental Alphabets (Dover, 1981).

"Chaos"