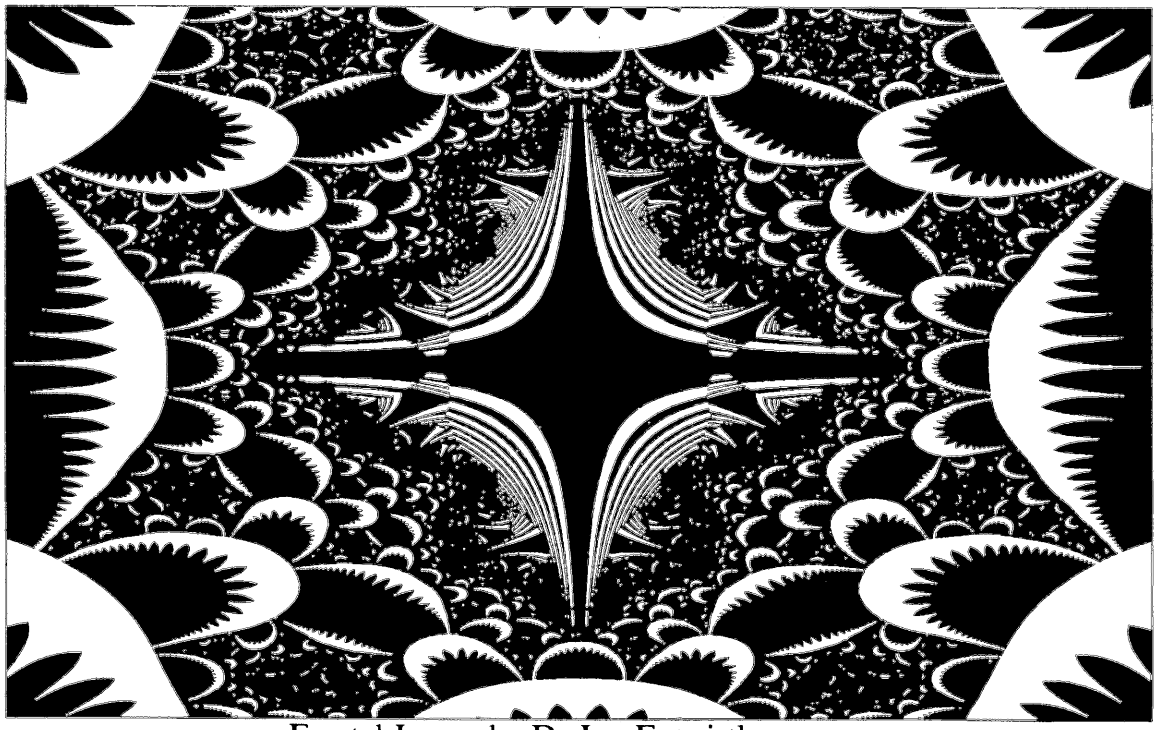
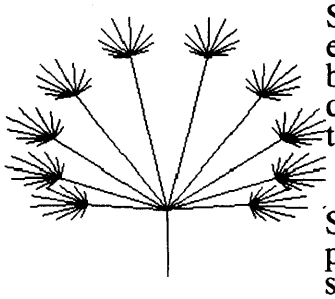
FRACTAL REPORT 14



Fractal Image by Dr Ian Entwistle.



Single copy rate £2. Subscription rates six issues: – £10 (UK only) £12 Europe £13 elsewhere. Cheques in British Pounds should be drawn on a UK bank and should be made payable to "Reeves Telecommunications Laboratories Ltd." Alternatively, dollar checks for $23 can be accepted if drawn on a U.S. bank and made payable to "J. de Rivaz".

Subscribers who are successful in getting one or more articles with programs published in a given series of six issues get the next volume of six issues free of subscription. All new subscriptions are backdated to the start of the current volume.

Fractal Report is published by Reeves Telecommunications Laboratories Ltd., West Towan House, Porthtowan, Truro, Cornwall TR4 8AX, United Kingdom.
Volume 3 no 14 First published April 1991. ISSN applied for.

THE SIERPINSKI CURVE

E J Turl

The excellent routine by W E Thomson in Issue 2 for producing a modified Sierpinski Curve prompted me to look up the original.
I have adapted Thomson's routine to produce this curve, and give the listing below in BBC BASIC.
PLOT1 is the BBC command for "draw relative to current point".

The curve has the amazing property that in its limit it has an infinite length, passes through every interior point of the bounding square, yet encloses an area a little less than half of the square. The figures for levels 1 & 2 show how at each stage, four shapes are reduced by a factor of 4 and joined in the middle. This middle section is the only increase in area from one level to the next (n), and can be verified to be $\frac{7}{8}4^{-n}A$, where A is the area of the bounding square. The area enclosed by level 1 is $\frac{11}{32}A$, from which it is a simple matter to show that the area inside the limit curve is $\frac{5}{12}A$.

The reference below also points out that a similar curve can be produced by building squares on alternate sides of an octagon. The routine can easily be changed to do this by replacing 2*u,2*v in each case by u,v. Other changes can be made, giving new space-filling curves. For example, figure 5, is the result of changing PLOT1,2*u,2*v to PLOT1,s,t:PLOT1,w,x Figure 6 is given as a puzzle for the reader. It needs 2 more parameters to be passed to the procedure, taken from W E Thomson's original routine.

Reference: Cundy & Rollett, "Mathematical Models", OUP

Listing:

```
  5 t$="The Sierpinski Curve"
 10 p=256:q=254:r=0
 20 INPUT"level: "n
 30 d=p/2^n
 40 MODE1:PLOT69,q+d/2,r+3*d/2
 50 PROCtom(n,d,d,0,d,-d,d)
 60 PLOT1,d,d
 70 PROCtom(n,d,-d,d,0,d,d)
 80 PLOT1,d,-d
 90 PROCtom(n,-d,-d,0,-d,d,-d)
100 PLOT1,-d,-d
110 PROCtom(n,-d,d,-d,0,-d,-d)
120 PLOT1,-d,d
130 FORc%=1TOLENt$
132    PRINTTAB(2.5+c%)MID$(t$,c%,1)
134    NEXT:REM title down LHS
136 PRINTTAB(0,28)"level ";n
190 END
200 DEFPROCtom(n,s,t,u,v,w,x)
210 IFn=1 PLOT1,s,t:PLOT1,2*u,2*v:PLOT1,w,x:ENDPROC
220 PROCtom(n-1,s,t,u,v,w,x)
230 PLOT1,s,t
240 PROCtom(n-1,-w,-x,v,-u,s,t)
250 PLOT1,2*u,2*v
260 PROCtom(n-1,w,x,-v,u,-s,-t)
270 PLOT1,w,x
280 PROCtom(n-1,s,t,u,v,w,x)
290 ENDPROC
```

Letter from Mr Larry Cobb:

I was interested to read Mike Parker's article about file compression in issue 13. Run length encoding (RLE) is a useful technique for reducing the size of graphics (and fractal images). It's great advantage is the relative simplicity of implementation but, as file sizes grow, other more powerful techniques should not be overlooked.

A logical extension of RLE, which recognises patterns data when they appear sequentially, is to build up a table of patterns and assign a reduced binary string to each so that this code can be substituted, no matter where the patterns occur in a graphics file. Such an algorithm is the Lempel-Ziv Welch, which has been used to great advantage by CompuServe in its Graphics Interchange Format (GIF), used for the transmission and storage of graphics. Some fractal investigation programs, not least DRAGONS and FRACTINT, and other more general graphics programs, such as Autodesk Animator, have adopted this standard for obvious reasons. It offers an increased compression of about two over RLE and, being a widely supported standard, it allows the interchange of files between computers, even between different types of machine. The GIF specification is freely available for study and use, without licence fees, from CompuServe International, who own the trademark.

It's interesting to speculate on where file compression can go from here. Can a more powerful standard be developed? There is clearly a need for better compression performance, particularly with the introduction of graphics cards like the IBM XGA. With 24 bits per pixel, these cards can give truly photographic results but where will we find room for the resulting super mega byte files? There is great interest at the moment in so-called "lossy" compression systems. In these a trade-off is sort between loss of picture quality (tiny in some cases) and improved compression performance.

The Joint Photographic Experts Group (JPEG) has developed a standard based of discrete cosine transforms - a method of resampling the data in proportion to the detail included in each area of the image. I can report that the results are very good for photographs but, so far, my tests have proved disappointing for fractal images.

Fractals themselves have been put forward as an alternative compression method - using the matching of fractal patterns to graphics images and then sending the resulting fractal equations.

These lossy systems are clearly a forward but they move the complexity from the transmission and storage media to the coding and decoding hardware. After all, I can "send" you a fractal with very little data:

$$zn+1 = zn + c$$

but no one has seen the end of that equation yet!

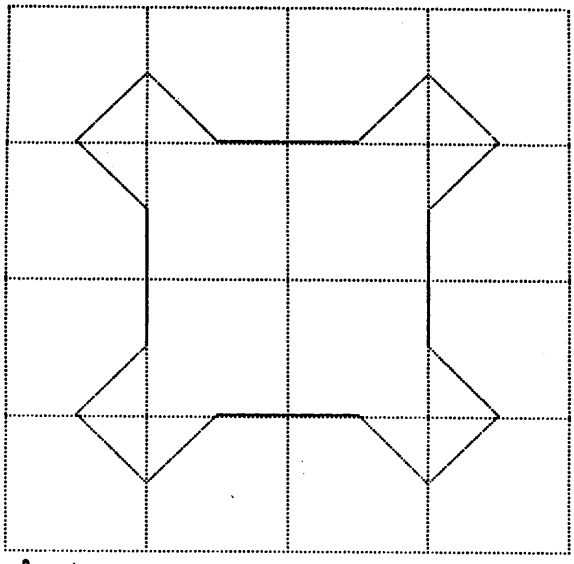


fig. 1 level 1

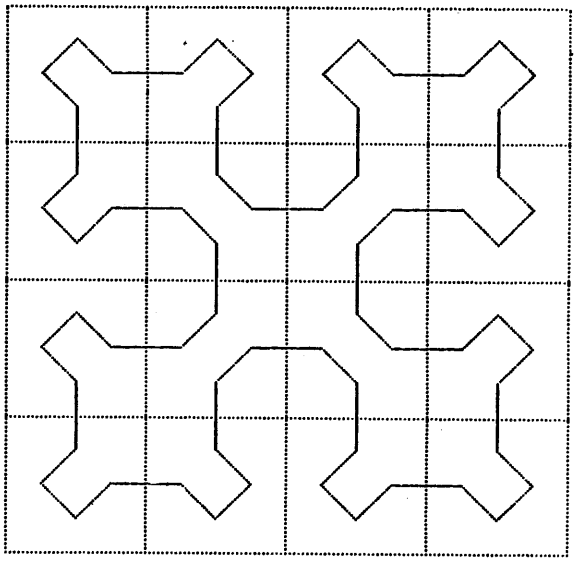
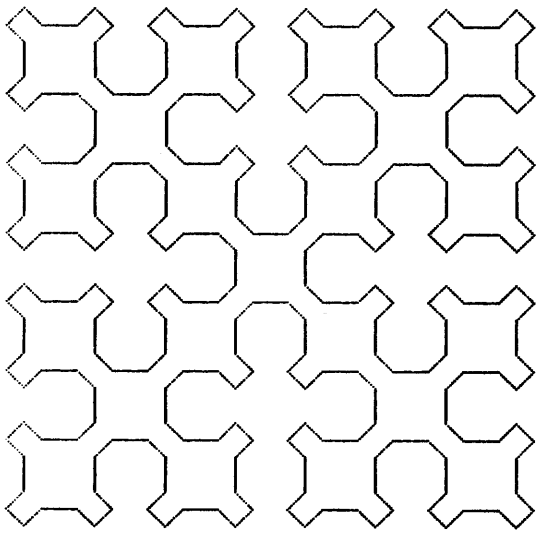
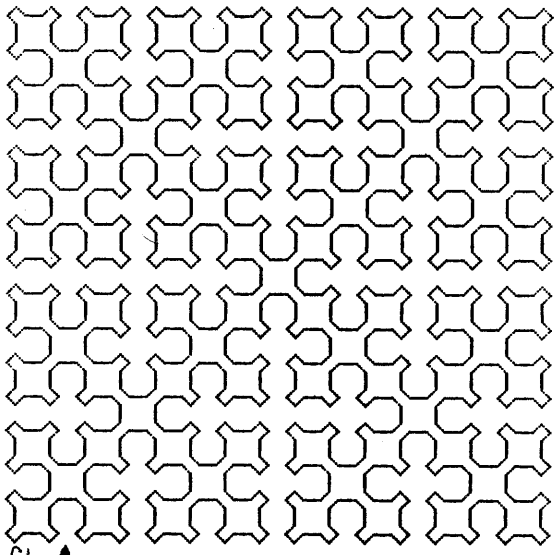


fig 2 level 2



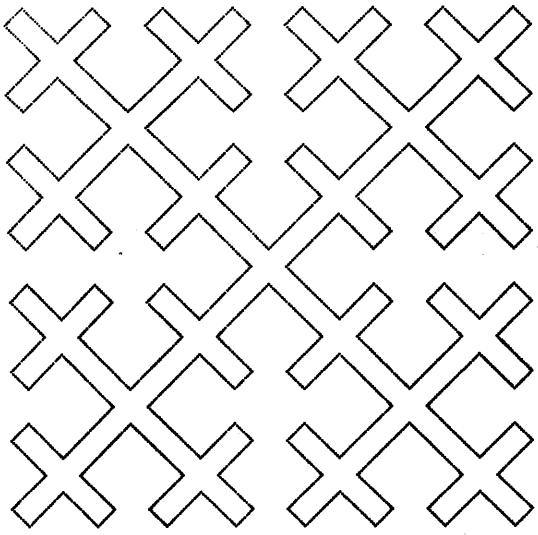
The Sierpinski Curve level 3

fig. 3



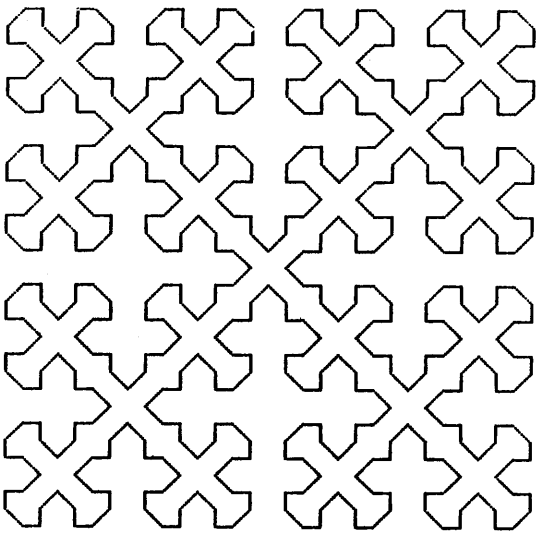
The Sierpinski Curve level 4

fig. 4



The Turl-Sierpinski Curve level 3

fig. 5



The Turl-Sierpinski Curve level 3

fig. 6

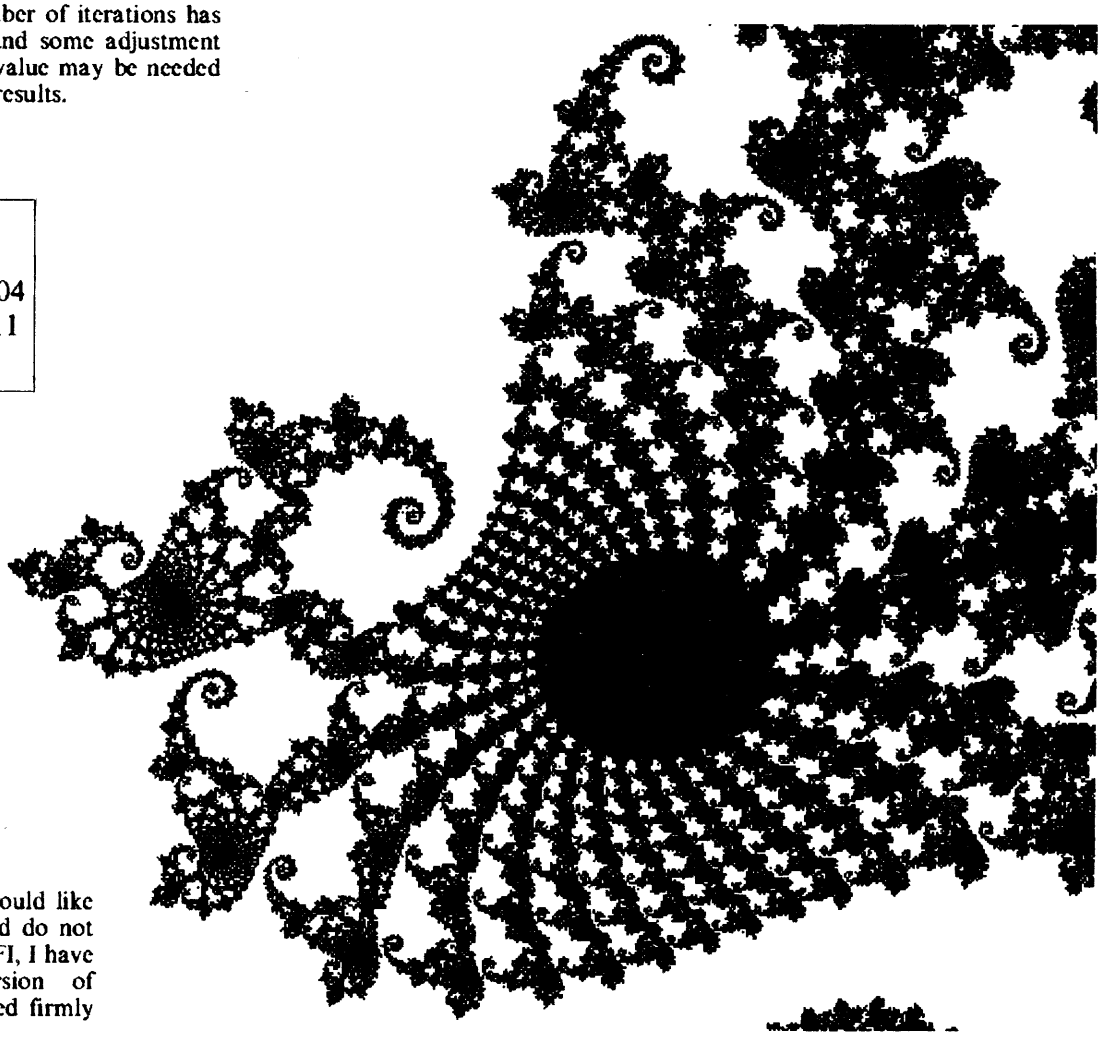
A BIT MORE DETAIL, PLEASE
by Graham Johnstone

I find that following a worked example removes some of the inadequacy I suffer from staring at mathematics described by the author as 'high school'. Well presented examples can illuminate theory to the point where I can at least grasp what is going on. This article, I hope, follows the precept that one example is more useful than five formulas.

The Science of Fractal Images (SFI)(1) contains many excellent examples, in the form of pseudo-code, that illustrate and amplify the text. In Chapter 4 of SFI Heinz-Otto Peitgen describes various methods for determining whether a point lies inside or outside the boundary of the Mandelbrot or Julia sets. I have developed the 'Distance Estimation' (DE) technique described to produce very acceptable (to me) renderings of parts of the Mandelbrot set. DE checks whether the point under calculation is within a limit - threshold - distance of being in the set. Therefore, there are now four possible outcomes for any point: inside the set, very close but not in the set, within the threshold but not in the set and outside the threshold. It is up to the user to decide how to employ this information in terms of output. There is also an extra constant - the threshold value - for reference within the program. The original essay does not have much to say on what might be reasonable threshold values to try although it depends partly on the screen resolution, number of iterations & other constants used and the maximum absolute value of numbers the system can handle. I used a trial and error approach to find suitable threshold values; Figure 2 demonstrates the key results. Note that in Figure 2 points 'in the set' are shown stippled, points 'within the threshold but not in the set' are shown solid black and 'outside the threshold' is left white (No points fell into the category 'very close to the set'). The same area of the Mandelbrot set (the southernmost tip) is shown using different values for the number of iterations and threshold. It is clearly shown that - as expected - increasing the number of iterations sharpens and contracts the points 'in the set' whilst decreasing the threshold reduces the contrast or definition of points 'within the threshold but not in the set'. Treating the threshold as a 'developer', significant improvement to the detail rendered can be achieved with many fewer iterations. As magnification is increased then certainly the number of iterations has to be increased and some adjustment to the threshold value may be needed to produce good results.

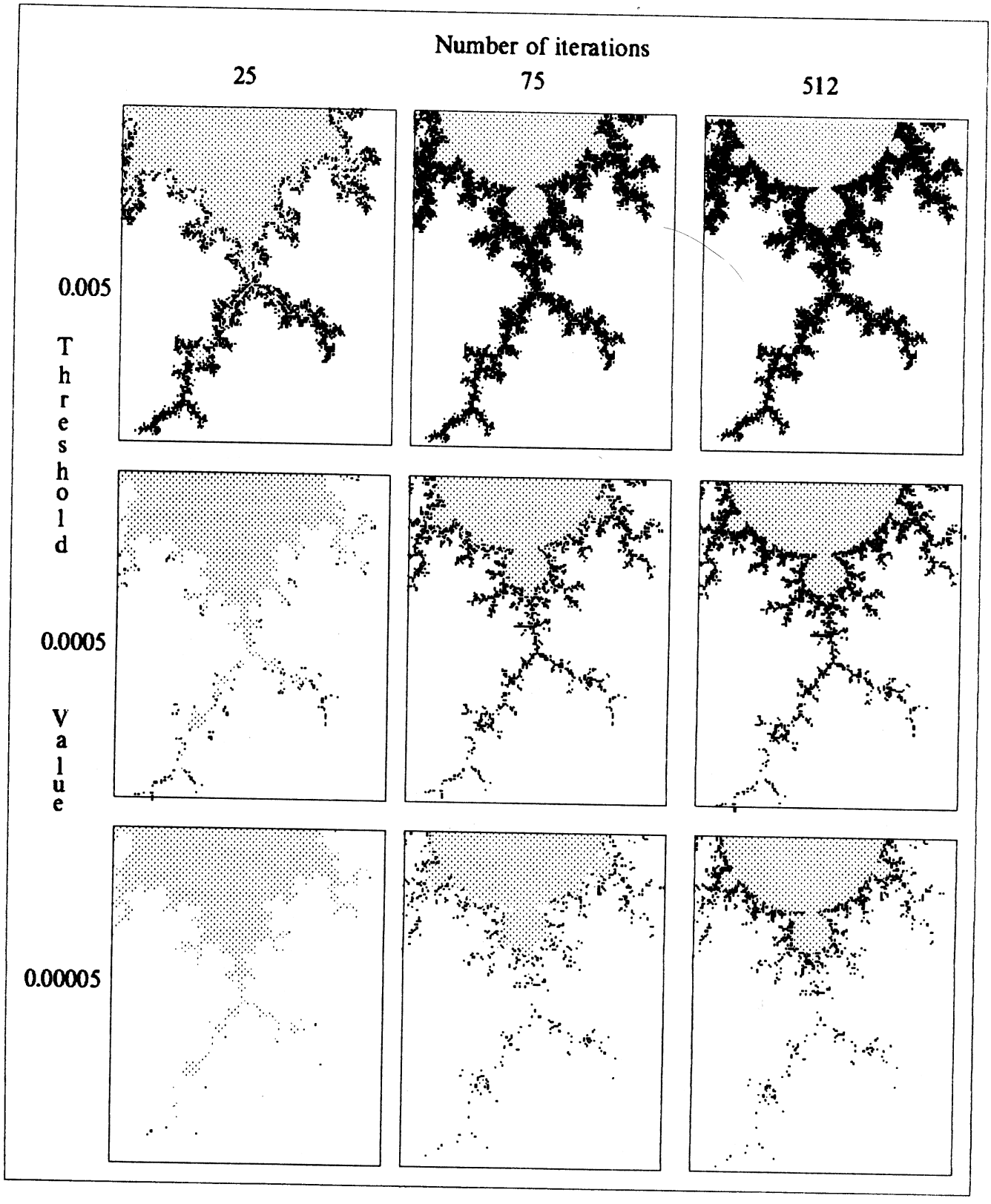
FIGURE 1

swx = -0.75104
swy = 0.10511
sid = 0.005



For those who would like to 'have a go' and do not have a copy of SFI, I have set out a version of pseudo-code based firmly on the original.

FIGURE 2 Output illustrating the effect of varying the threshold and the number of iterations.



swx = -0.25 , swy = -1.15 , sid = 0.25

PSEUDO-CODE for Distance Estimation

PROGRAM STRUCTURE

```
INPUTS:
  swx, swy      ;South West x & y values
  sid           ;length of side
  nx,ny         ;number of pixels in x & y axes
  maxiter       ;maximum number of iterations
  threshold     ;distance from M_set
                   estimate in pixel units

INITIALISE:
  SET delta = threshold * sid / (nx-1)
  SET huge = 100,000

M_SET LOOP:
  FOR iy = 0 TO ny - 1 DO
    cy = swy + (iy * sid) / (ny-1)
    FOR ix = 0 TO nx - 1 DO
      cx = swx + (ix * sid) / (nx - 1)

      PIXEL LOOP:
      SET x = y = x2 = y2 = dist = xorbit [ 0 ] = yorbit [ 0 ] = 0
       iter=0       ;iteration counter 1

        WHILE ( iter < maxiter ) AND (  (x2 + y2) < huge ) DO
          temp = x2 - y2 + cx
          y = 2 * x * y  + cy
          x = temp
          x2 = x * x
          y2 = y * y
          iter = iter + 1
          xorbit [ iter ] = x
          yorbit [ iter ] = y
        ENDWHILE

      IF ( iter = maxiter ) THEN  SET pix = 0 : JUMP PLOT
      IF ( x2 + y2 ) > huge  THEN

        SET  xder = yder = 0       ;derivatives of x & y
         i = 0     ;iteration counter 2

        WHILE ( i < iter ) AND NOT ( overflow ) DO
          temp =  2 * ( xorbit [ i ]  * xder  -  yorbit [ i ] * yder )  +  1
          yder =  2 * ( yorbit [ i ]  * xder  +  xorbit [ i ] * yder)
          xder = temp
          TEST xder NOT overflow
          TEST yder NOT overflow
           i = i + 1
        ENDWHILE

        IF ( overflow ) THEN  SET pix = -1 : JUMP PLOT
        A = x2 + y2  :  B = xder * xder  :  C = yder * yder
        dist = ( LOG ( A ) * SQRT ( A ) ) / SQRT (  B  +  C  )
        IF dist < delta THEN  SET pix = 1  ELSE  SET pix = 2

      PLOT:
        [ Choose COLOUR for output according to pix:
            0 - in the M_set
           -1 - very close to M_set
            1 - within the threshold
            2 - outside M_set ]
        SET POINT  [ ix , iy ]  TO COLOUR

    ENDFOR  [ ix ]

  ENDFOR  [ iy ]

END
```

NOTES

1. For each pixel, the first WHILE block carries out the standard M_set calculation and stores each x & y value for possible use in the second WHILE block. The first WHILE block terminates either through reaching maxiter or if the sum of the squares exceeds huge. (Trying other values of huge might be interesting).

2. In the second WHILE block TEST represents a mechanism to determine either – that overflow has just occurred or – that overflow will occur if the next multiplication is attempted and exit accordingly i.e. SET pix = –1 and jump to the PLOT routine; this result means that the point is very close to the M_set. [Maybe its my programming but I have not yet hit an overflow condition]. Otherwise the block will end when i reaches the value of iter . Now calculate the 'distance' from the M_set and compare it with delta to decide whether the point is within the threshold.

3. For points outside of the threshold, there are other colouring options to try based on the exit values of iter or i etc.

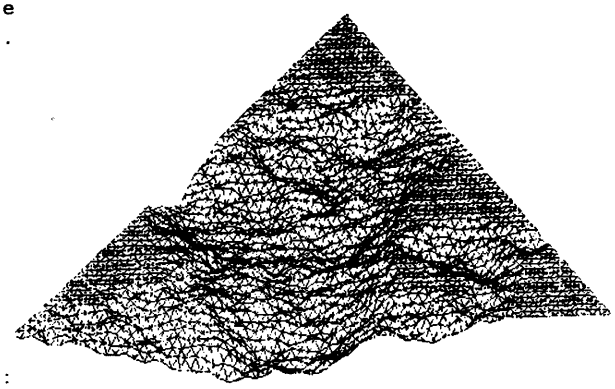
4. Everyone 'tunes' code to their own satisfaction. However, I do recommend getting a 'simple' version working first to ensure that all the loops and indexing are working correctly. During my program development an early failure was due entirely to over enthusiastic and premature 'tuning'. And it took a long time to find !

5. The figures were produced on an Archimedes 440 programmed in BASIC & Assembler and using the floating point emulation package for calculation in double precision arithmetic (17 significant digits). The article was produced using the Archimedes !Edit & !Draw applications and printed on a Fujitsu DPL 24 pin printer.

REFERENCES

1. The Science of Fractal Images.
   Edited by Peitgen & Saupe
   Published by Springer-Verlag
   Chapter 4:
   Fantastic deterministic fractals
   by Heinz-Otto Peitgen

```
Pograma:  Paisaje.bas
Función:  Generar paisajes fractales mediante latecnica de
          Desplazamiento de Medio Punto.
Autor:    desconocido. Adaptado a Qbasic por José E.
          Murciano

DEFINT A-N: KEY OFF: SCREEN 9: pi = 3.141592: CLS
DIM d(130, 110)

zzz = TIMER: RANDOMIZE zzz
INPUT "Nivel de recursion ?"; le
CLS
ds = 2: FOR n = 1 TO le: ds = ds + 2 ^ (n - 1): NEXT n
mx = ds - 1: my = mx / 2: rh = pi * 30 / 180: vt = rh * 1.2
FOR n = 1 TO le: l = 10000 / 1.8 ^ n
  LOCATE 2, 2: PRINT "Trabajando en el nivel "; n
  ib = mx / 2 ^ n: sk = ib * 2
  GOSUB 160: 'Altura a lo largo de x
  GOSUB 230: 'Altura a lo largo de y
  GOSUB 300: 'Altura en la diagonal
NEXT n
GOTO 650: 'Dibujo
'Altura en direccion x

160 FOR ye = 0 TO mx - 1 STEP sk
    FOR xe = ib + ye TO mx STEP sk
    ax = xe - ib: ay = ye: GOSUB 380: d1 = d: ax = xe + ib:
    GOSUB 380: d2 = d
    d = (d1 + d2) / 2 + RND(1) * l / 2 - l / 4: ax = xe:
    ay= ye: GOSUB 430
NEXT xe
NEXT ye: RETURN
'Altura en el eje y
230 FOR xe = mx TO 1 STEP -sk
    FOR ye = ib TO xe STEP sk
    ax = xe: ay = ye + ib: GOSUB 380: d1 = d: ay = ye - ib:
    GOSUB 380: d2 = d
    d = (d1 + d2) / 2 + RND(1) * l / 2 - l / 4: ax = xe:
    ay= ye: GOSUB 430
NEXT ye
NEXT xe: RETURN
'Altura en la diagonal
300 FOR xe = 0 TO mx - 1 STEP sk
    FOR ye = ib TO mx - xe STEP sk
    ax = xe + ye - ib: ay = ye - ib: GOSUB 380: d1 = d
    ax = xe + ye + ib: ay = ye + ib: GOSUB 380: d2 = d
    ax = xe + ye: ay = ye: d = (d1 + d2) / 2 + RND(1) * l /2 - l / 4: GOSUB 430
NEXT ye
NEXT xe: RETURN
'Obtencion de datos de la matriz
380 IF ay > my THEN 400
by = ay: bx = ax: GOTO 410
400 by = mx + 1 - ay: bx = mx - ax
410 d = d(bx, by): RETURN
'Escritura en la matriz
430 IF ay > my THEN 450
by = ay: bx = ax: GOTO 460
450 by = mx + 1 - ay: bx = mx - ax
460 d(bx, by) = d: RETURN
'Aqui se situa el nivel del mar
480 IF x0 <> -999 THEN 510
IF zz < 0 THEN GOSUB 1080: z2 = zz: zz = 0: GOTO 630
GOSUB 1100: GOTO 620
510 IF z2 > 0 AND zz > 0 THEN 620
IF z2 < 0 AND zz < 0 THEN z2 = zz: zz = 0: GOTO 630
w3 = zz / (zz - z2): x3 = (x2 - xx) * w3 + xx: y3 = (y2 - yy) * w3 + yy: z3 = 0
zt = zz: yt = yy: xt = xx
IF zz > 0 THEN 600
'Aqui ya agua
zz = z3: yy = y3: xx = x3: GOSUB 960
GOSUB 1080: zz = 0: yy = yt: xx = xt: z2 = zt: GOTO 630
'Emerge del agua
600 zz = z3: yy = y3: xx = x3: GOSUB 960
GOSUB 1100: zz = zt: yy = yt: xx = xt
620 z2 = zz
630 x2 = xx: y2 = yy: RETURN
'Presentacion en pantalla

650 GOSUB 1120: 'Inicializa la pantalla
xs = .05: ys = .05: zs = .05: 'Factores de escala
FOR ax = 0 TO mx: x0 = -999: FOR ay = 0 TO ax
GOSUB 380: zz = d: yy = ay / mx * 10000: xx = ax / mx * 10000 -yy / 2
GOSUB 950: NEXT ay: NEXT ax
FOR ay = 0 TO mx: x0 = -999: FOR ax = ay TO mx
GOSUB 380: zz = d: yy = ay / mx * 10000: xx = ax / mx * 10000 - yy / 2
GOSUB 950: NEXT ax: NEXT ay
FOR ex = 0 TO mx: x0 = -999: FOR ey = 0 TO mx - ex
ax = ex + ey: ay = ey: GOSUB 380: zz = d: yy = ay / mx * 10000
xx = ax / mx * 10000 - yy / 2: GOSUB 950: NEXT ey: NEXT ex
GOTO 1140: 'Acaba y sale del bucle

'Rotar
780 IF xx <> 0 THEN 810
IF yy <= 0 THEN ra = -pi / 2: GOTO 830
ra = pi / 2: GOTO 830
810 ra = ATN(yy / xx)
IF xx < 0 THEN ra = ra + pi
830 r1 = ra + rh: rd = SQR(xx * xx + yy * yy)
xx = rd * COS(r1): yy = rd * SIN(r1)
RETURN

870 rd = SQR(zz * zz + xx * xx)
IF xx = 0 THEN ra = pi / 2: GOTO 910
ra = ATN(zz / xx)
IF xx < 0 THEN ra = ra + pi
910 r1 = ra - vt
xx = rd * COS(r1) + xx: zz = rd * SIN(r1)
RETURN
'Moverse a (xp,yp)

950 GOSUB 480
960 xx = xx * xs: yy = yy * ys: zz = zz * zs
GOSUB 780: 'Rotar
GOSUB 870
IF x0 = -999 THEN pr$ = "M" ELSE pr$ = "D"
xp = INT(yy) + cx: yp = INT(zz)
GOSUB 1040
RETURN
'Dibujar

1040 xp = xp * 1.1: yp = yp + 260: IF pr$ = "M" OR f1 = 1 THEN x8 = xp: y8 = yp
PSET (x8, 350 - y8), f1: LINE -(xp, 350 - yp), f1: x8 = xp: y8 = yp: x0 = xp
RETURN
'Color

1080 f1 = 9: RETURN
'Color de la tierra

1100 f1 = 6: RETURN
'Iinicializa pantalla o plotter

1120 CLS : RETURN
'Salida

1140 a$ = INKEY$: WHILE LEN(a$) = 0: a$ = INKEY$: WEND
STOP
```



THE VOLTERRA-LOTKA SYSTEM

E J Turl

The Volterra-Lotka equations provide a simple mathematical model of a predator-prey relationship. In the absence of predators, the prey is assumed to grow at a rate x' proportional to its population x: $x' = ax$

If predators are present, the prey is consumed in proportion to the predator population y, and *also* (as a simplification) in proportion x. (The predators eat more & raise more young successfully when there's more to eat.) So:

$x' = ax - bxy$

In the absence of prey, the predators are assumed to decrease at a rate y' proportional to y. (The more there are, the more there are to die.): $y' = -cy$

But when there is prey, the predators will multiply more quickly than they die off, in proportion to their numbers y, and *also* (reason as before), to x, So:

$y' = -cy + dxy$

To process these differential equations on the computer, some method has to be chosen of calculating approximate jumps in the values of x & y. This is quite justifiable as a representation of reality because in ecological systems, this is often just what happens, as the system is subject to diurnal, seasonal, annual or other interruptions. In "The Beauty of Fractals", Peitgen & Richter give a method of 'discretising' the equations which combines two standard methods, the Euler & Heun methods. Representing $ax-bxy$ by $f(x,y)$, or f, and $aX-bXY$ by $f(X,Y)$, or F; $-cy+dxy$ by $g(x,y)$, or g, and $-cY+dXY$ by $g(X,Y)$, or G, the two methods can be described as follows. Euler's method is the simpler, in which x & y are incremented by fh & gh, where h is a period of time:

$X = x + fh : Y = y + gh$

Heun's method instead increments the variables using average values of the growth rates evaluated at the beginning and end of each step of Euler's method:

$X = x + (f+F)h/2 : Y = y + (g+G)h/2$

P & R use essentially the Heun method, but allow the possibility of a different time period, p, for the calculation of F, i.e. F might be evaluated before or after the end of the Euler step:

$X = x + (f + f(x+pf,y+pg))h/2 : Y = y + (g + g(x+pf,y+pg))h/2$

It can be seen that the process reduces to Euler's method if $p=0$, and Heun's method if $p=h$. P & R have explored the behaviour of the V-L system for $0 < h,p < 1$, using $a=b=c=d=1$, as these coefficients presumably only affect the scale of the process. Depending on the choice of h & p, and the initial values of x & y, the system tends, except for a few special cases, towards either an *invariant cycle*, or a *periodic attractor*, or a *strange attractor*, or infinity. The figures show results for $h=p=0.72$. Fig.1 shows initial values leading to the invariant cycle, fig.2, to the periodic attractor. Fig.3 shows superimposed growth cycles for a range of initial values (incrementing x & y by 0.125, and plotting 30 steps). Fig.4 shows the *attractor basins*, i.e. which initial values lead to which attractor.

The textbook example of this type of predator-prey cycle is that of the snowshoe hare and lynx in Canada, which has a period of about 10 years, but there is now some doubt about whether this is due to predation or food toxins.

Xo=1.010 Yo=0.990

fig.1

Volterra – Lotka System
h = 0.72 : p = 0.72
(0,0) to (5,4)
image reduced in relation to fig 4

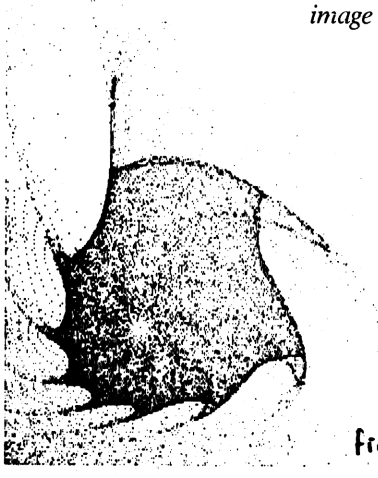


fig.3

Listing:

The listing below in BBC BASIC gives the program required to generate fig.4 (with alterations in REMs for fig.3). Note that the value of 'invmax' has been obtained by observing the maximum values of y in the invariant & periodic attractors in figs.1 & 2, and choosing a value roughly halfway between. To improve the speed of the program, the equations have been reduced to their simplest form from those given above, and intermediate variables defined to avoid duplication. Thus the working formulae are:

$$X = x + fh(1+p/2) - H \quad : \quad Y = y + gh(1-p/2) + H$$
$$\text{where } H = (xy(x-y) + pfg)hp/2$$

```
  0 REM"V-Lsystem"
 10 MODE1:REM or 0 for fig.3
 20 ONERROR att%=2:GOTO230:REM generated by X,Y "Too Big"
 30 h=0.72:p=0.72
 40 Xlo=0:Xhi=5:Ylo=0:Yhi=4
 50 Xg=1280:Yg=1024:REM screen range
 60 I%=50:REM 'settling' iterations
 70 J%=10:REM 'testing' iterations
 80 invmax=2.35
 90 step%=4:REM or 32 for fig.3
100 Xrange=Xhi-Xlo:Yrange=Yhi-Ylo:P=Xrange/Xg:Q=Yrange/Yg
110 q=h/2*p:r=h+h*p/2:s=h-h*p/2:t=q*p
120 H%=0
130 V%=0
140 X=Xlo+H%*P:Y=Ylo+V%*Q
150 att%=0:REM or 2 for fig.3
160 FORC%=1TOI%:PROCiterate
170   REM include PLOT69,(X-Xlo)/P,(Y-Ylo)/Q for fig.3
180   NEXT
190 C%=0:REPEAT:REM omit REPEAT loop for fig.3
200 PROCiterate:C%=C%+1
210 IFABS(X)>10 ORABS(Y)>10 att%=2 ELSEIFY>invmax att%=1 ELSEIFY<.1 att%=-1
220 UNTILC%>J%ORatt%>0
230 GCOL0,1+att%:REM attractor type selects colour
240 PLOT69,H%,V%
250 V%=V%+step%:IFV%<Yg GOTO140
260 H%=H%+step%:IFH%<Xg GOTO130
270 END
280 DEFPROCiterate
290 D=X-Y:E=X*Y:F=X-E:G=E-Y:H=q*D*E+t*F*G:X=X+r*F-H:Y=Y+s*G+H
300 ENDPROC
```

Xo=1.000 Yo=2.500

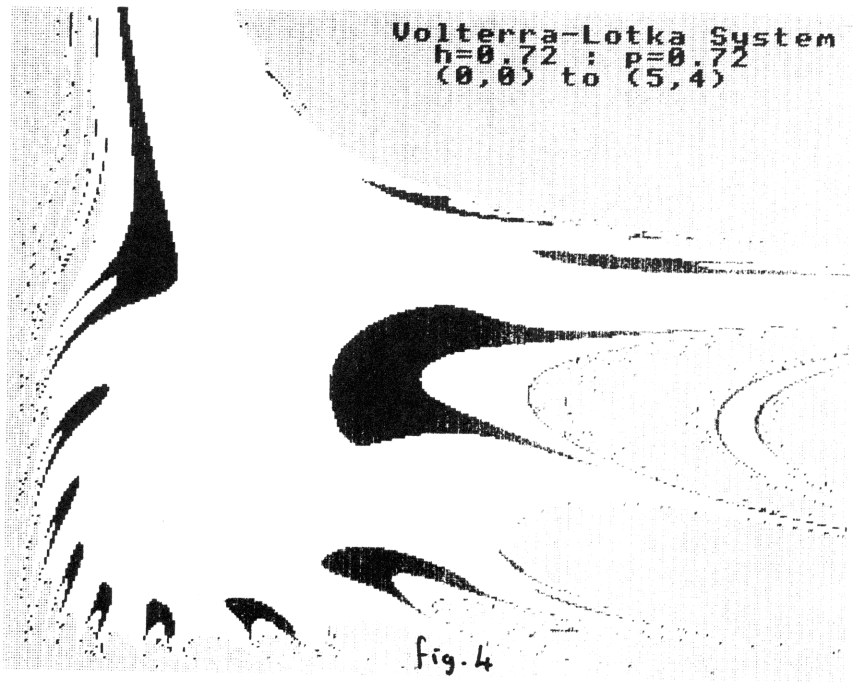


fig.2

fig.4

The Bifurcation Diagram and Chebyshev Polynomials

John Topham

The bifurcation diagram (figure 1) is a graph showing the onset of chaos in a population growth model when the growth factor 'R' increases beyond a certain point. See Peitgen and Richter 'The Beauty of Fractals' for further details.

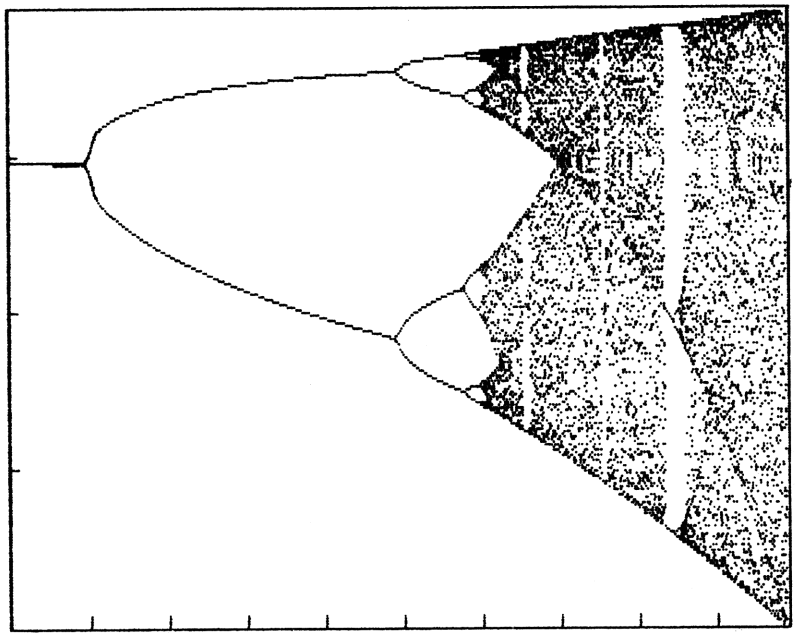


Figure 1
Bifurcation Diagram from system $y_{n+1}=y_n+Ry_n(1-y_n)$

Interval of R: [1.9,3]
Interval of y: [0,1.333]

The model takes the equation $y_{n+1} = y_n + Ry_n(1-y_n)$ to calculate each year's population level.

So at year 1 or when n=0 we have: $y_1 = y_0 + Ry_0(1-y_0)$.
('y_0' is an initial value given to the population)

At year 2 or n=1 we have: $y_2 = y_1 + Ry_1(1-y_1)$

or $y_2 = (y_0+Ry_0(1-y_0)) + R(y_0+Ry_0(1-y_0))(1-(y_0+Ry_0(1-y_0)))$

Re-arranging for 'R' and sudstituting $a = y_0(1-y_0)$:
$y_2 = y_0 + Ra + R^2a(1+2y_0) - R^3a^2$

At year 3 or n=2: $y_3 = y_2 + Ry_2(1-y_2)$

or $y_3 = R^7l(a) + R^6k(a) + R^5j(a) + R^4i(a) + R^3h(a) + R^2g(a) + Rf(a) + y_0$

where $f(a)=3a$, $g(a)=a(3-4y_0)$, $h(a)=a(1-4y_0-5a)$, $i(a)=-a^2(5+6y_0)$, $j(a)=a^2(1+2y_0)-2a^3$, $k(a)=-2a^3(1+2y_0)$, $l(a)=a^4$

and so on ....

Clearly writing the equation out in full for each iteration is laborious. However, with this way it can be seen that if 'y_0' is the same, i.e. a constant, for each value of 'R' then a polynomial in 'R' is generated. The degree of each polynomial for each successive iteration increases by powers of 2. That is, the degree of polynomial $P_n(R)$ is $n=(2^k)-1$, $k=1,2,3,.....$ It can then be seen that even after a few iterations the details of the oscillations are lost in complexity. They cannot be drawn as lines, only as dots. (see figure 1)

If, however, only a small number of low level iterations are generated (i.e. only 10 to 20 iterations) and a small interval of 'R' is taken (see figure 2), then the behaviour of the oscillations of each iteration as 'R' increases can be observed. On certain intervals some of these low level oscillations display fractal-like patterns (figure 3). These show a striking similarity to a certain collection of 'Chebyshev' polynomials.

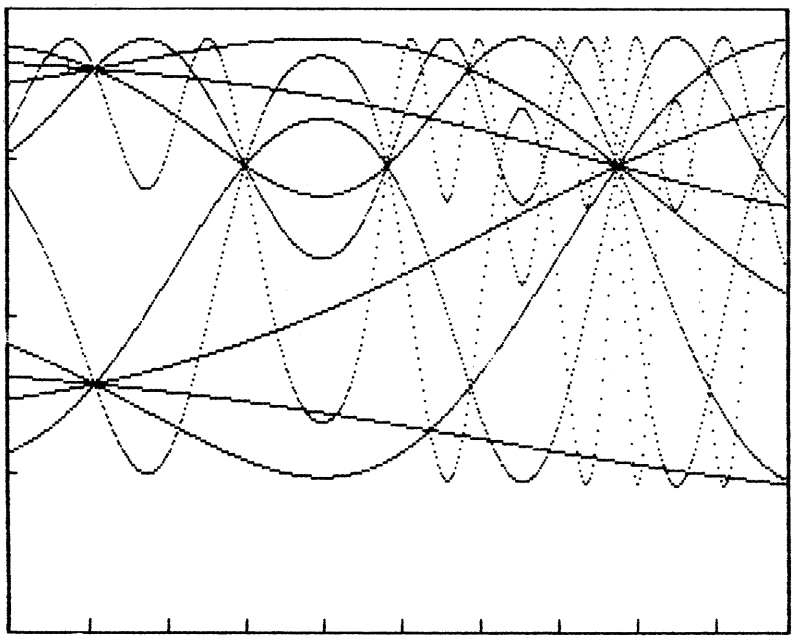


Figure 2
General Low iteration small interval window of the Bifurcation Diagram of the above system.

Max No. of Iter: 18
Min No. of Iter: 9
Interval of R:
[2.705,2.74]
Interval of y:
[0,1.333]

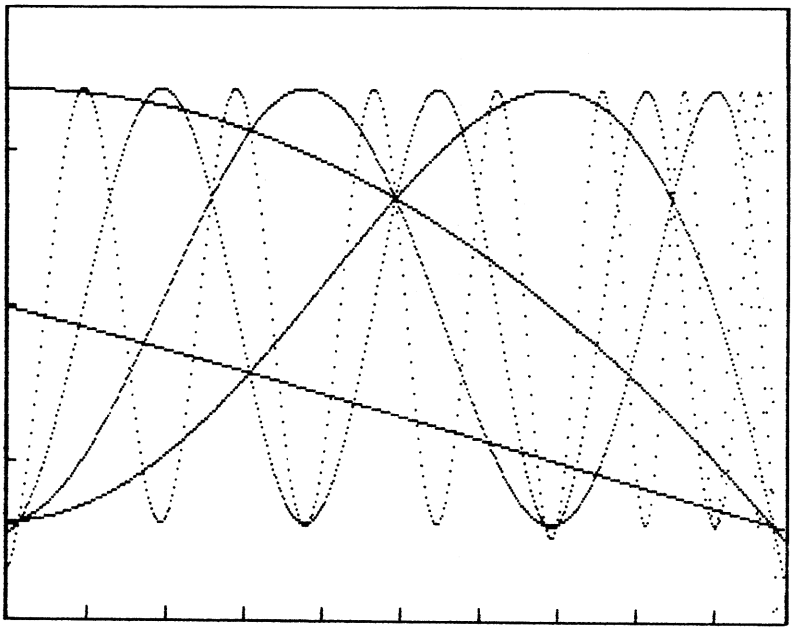


Figure 3
Low iteration, small interval window showing fractal structure.

Max No. of Iter: 18
Min No. of Iter: 13

Interval of R:
[3.001933,3.002013]
Interval of y:
[-.3,1.6]

Chebyshev polynomials $\{T_n(x),\ n = 1,2,3,....\}$ are a set of linearly independent functions that are used to approximate other more complex functions over a certain interval.

They are defined as follows:
$T_1(x) = 1,\ T_2(x) = x,\ T_3(x) = 2x^2 - 1,\ T_4(x) = 4x^3 - 3x$
$T_5(x) = 8x^4 - 8x^2 + 1,$

or in general terms;
$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad n >= 1$

Figure 4 shows a collection of Chebyshev polynomials $T_0(x)$ to $T_6(x)$ in the interval [-1,1]. If a collection of '$T_n(x)$' are drawn where $n = 2^k$, $k = 1,2,3,....$ (figure 5) patterns emerge that look similar to figure 3. It is odd that this is so since the degree of each set of polynomials increases in a different way (The degree of the growth model polynomials increases by $(2^k)-1$ and the degree of the Chebyshev polynomials increase by 2^k) and you would expect each polynomial to behave in a different way. Also when other functions are studied - functions that are seemingly unrelated to polynomials, such as sines, cosines and logarithmic functions - the same similarity occurs (Figures 7 to 10).

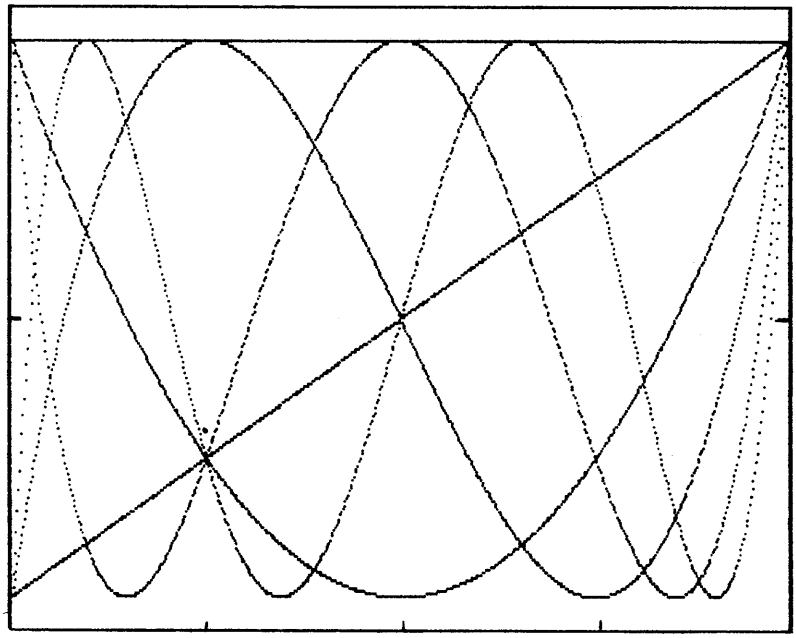


Figure 4
Set of Chebyshev Polynomials - $T_0(x)$ to $T_5(x)$ - on interval [-1,1].

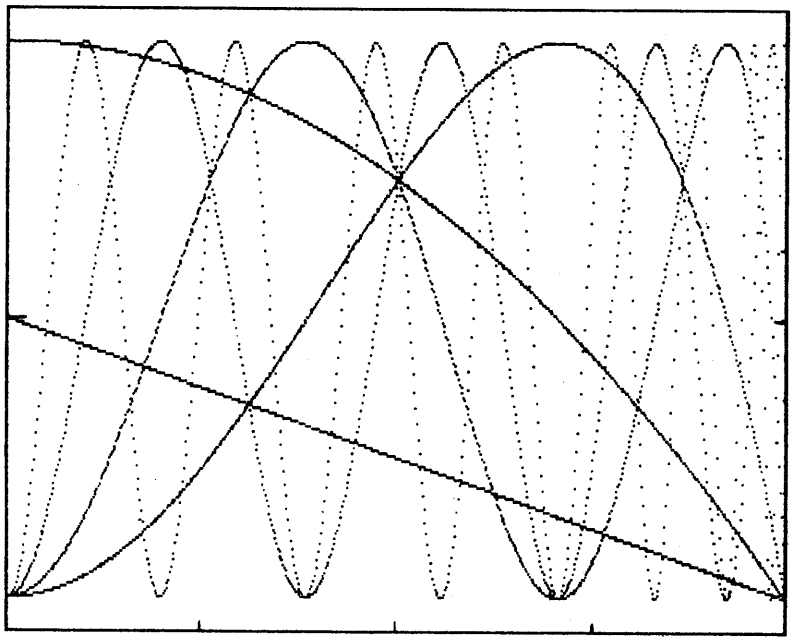


Figure 5
Subset of Chebyshev Polynomials: $\{-T_n(x)\}$ $n=2^k$, $k=0,1,2,3,4,5$ on interval [0,1]

Chebyshev polynomials can also be generated by the equation:

$T_n(x) = \cos(n \arccos(x))$ $\quad n = 1,2,3,....$ $\quad x$ in $[-1,1]$

Using $n=2^k$, $k=1,2,3,...$ produces the same effect.

There is also another collection of functions that produce similar curves to those shown in the figures especially figure 10. (see figure 6)

These are simply the set: $G_n(x) = \cos(nx)$ x in [-pi,0]
Figure 6 shows a subset where $n = 2^k$, k=-1,0,1,2,3,4.

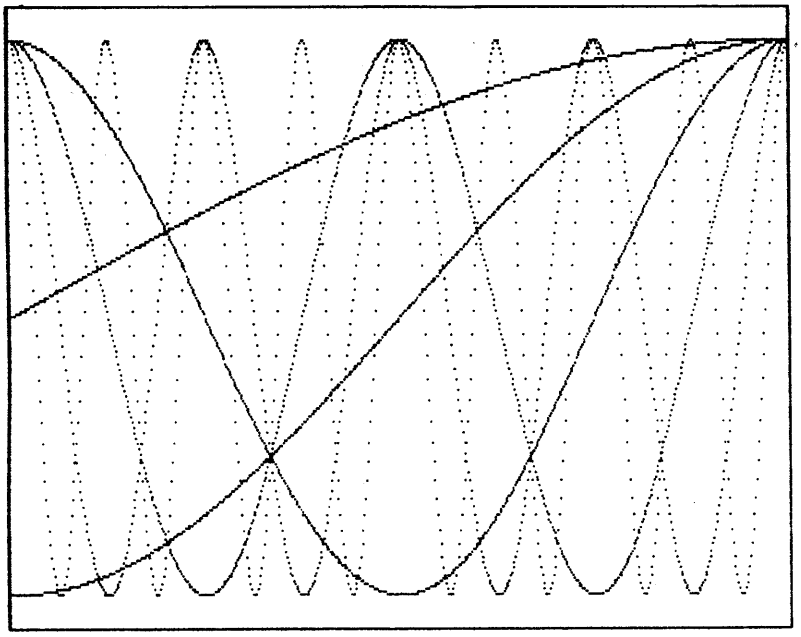


Figure 6
Set $G_n(x)$ $n = 2^k$,
k = -1,0,1,2,3,4
on interval [-pi,0]

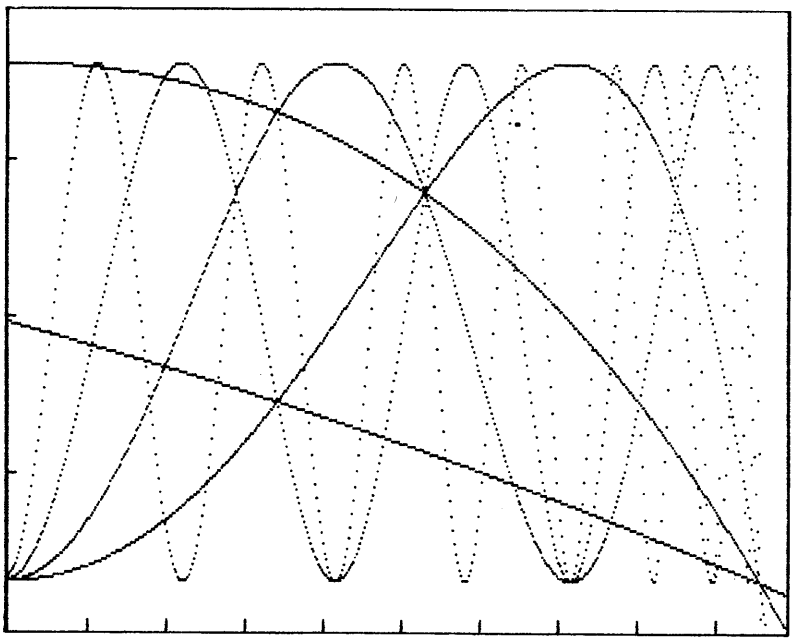


Figure 7
Diagram of system:
$y_{n+1} = R(1-y_n)y_n$

Max No. of Iter: 12
Min No. of Iter: 7

Interval of R:
[3.99985,4.000005]
Interval of y:
[-.1,1.1]

$y_0 = 0.5$

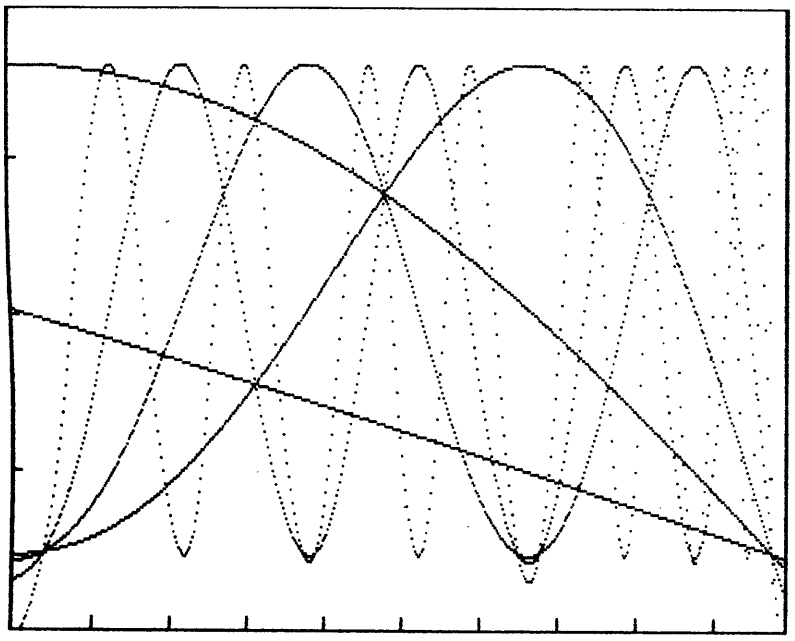


Figure 8
Diagram of system:
$y_{n+1} = R\sin(y_n)$

Max No. of Iter: 13
Min No. of Iter: 8

Interval of R:
[3.1471,3.1476]
Interval of y:
[-.5,3.5]

$y_0 = 0.5$

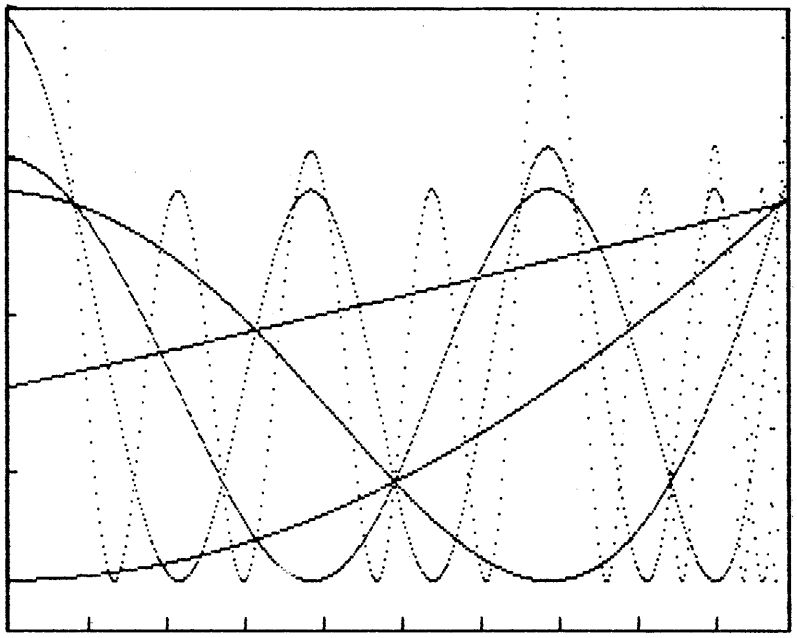


Figure 9
Diagram of system:
$y_{n+1}=R\cos(y_n)$

Max No. of Iter: 12
Min No. of Iter: 7

Interval of R:
[4.2135,4.219]
Interval of y:
[-4.5,-1]

$y_0 = 0.5$

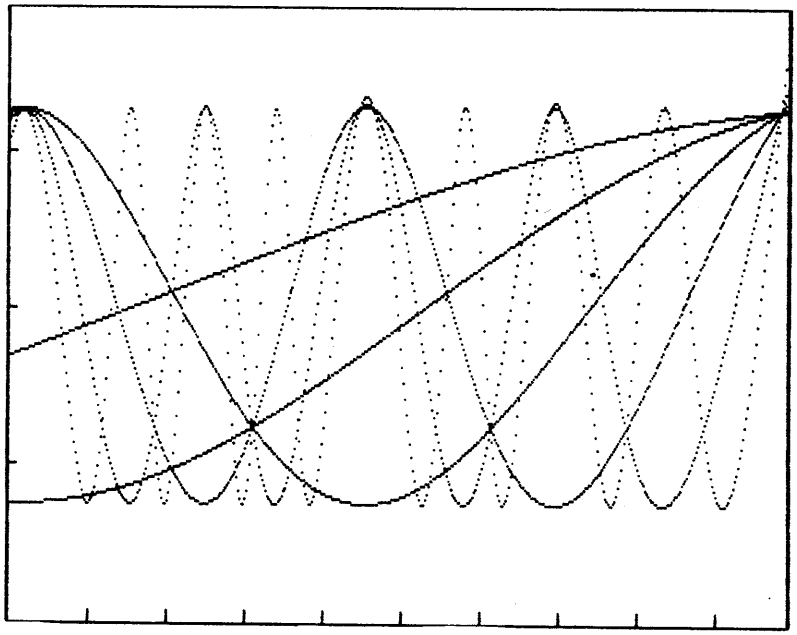


Figure 10
Diagram of system:
$y_{n+1}=Ry_n(\ln((y_n)^2-1)$

Max No. of Iter: 24
Min No. of Iter: 13

Interval of R:
[1.28991,1.29069]
Interval of y:
[-1.7,-1]

$y_0 = 0.5$

Of course, there is no particular use for any of these coincidences or observations, however, it is interesting to discover a form of unity in mathematics which is hitherto unsuspected. If anyone is interested there are a couple of programs written in Basic to investigate these similarities further. The first program enables you to look at close intervals of the bifurcation diagram and the second program draws Chebyshev polynomials of any degree you wish.

Reference: The Beauty of Fractals - H. Peitgen & P. Richter
Springer-Verlag

```
100 REMark PERIOD DOUBLING
105 REMark SCENARIO For
110 REMark Yn+1 = Yn+R(1-Yn)Yn
120 :
130 CLS
140 :
150 Max_Iteration=20
160 Min_Iteration=10
170 :
180 Rmin=1.9    :Rmax=3
190 Y_lower=0   :Y_upper=1
200 Yo=.5
210 :
220 scr_width=500
230 scr_height=200
240 :
245 R_width=Rmax-Rmin
250 Y_scale_fac=scr_height/1.333
260 r_scale_fac=scr_width/R_width
270 incmt=R_width/scr_wdth
280 :
290 FOR R=Rmin TO Rmax STEP incmt
300  Yn=Yo
310  FOR n=0 TO Max_Iteration
320   Ym=Yn+R*Yn*(1-Yn)
330   u=(R-Rmin)*r_scale_fac
340   v=Ym*Y_scale_fac
350   IF n>=Min_Iteration
360     PLOT_POINT u,v
370   END IF
380   Yn=Ym
390  END FOR n
400 END FOR R
410 STOP
```

```
100 REMark CHEBYCHEV POLYNOMIALS
110 :
120 CLS
130 :
140 x1=-1:x2=1
150 X_width=x2-x1
160 :
170 Scr_width=500
180 Scr_ht=200
190 :
200 X_scale_fac=Scr_width/X_width
210 Y_scale_fac=Scr_ht/2
220 :
230 incrmt=X_width/Scr_width
240 Mid_Scr=Scr_ht/2
250 :
260 FOR n=1 TO 4
270   :
280   FOR x=x1 TO x2 STEP incrmt
290     y=COS(n*ACOS(x))
300     xp=(x-x1)*X_scale_fac
310     yp=y*Y_scale_fac+Mid_Scr
320     IF x=x1:LINE 0,yp:END IF
330     LINE TO xp,yp
340   END FOR x
350  :
360 END FOR n
370 STOP
```

Fractal Shareware for the Acorn Archimedes

by Phil Edmonds

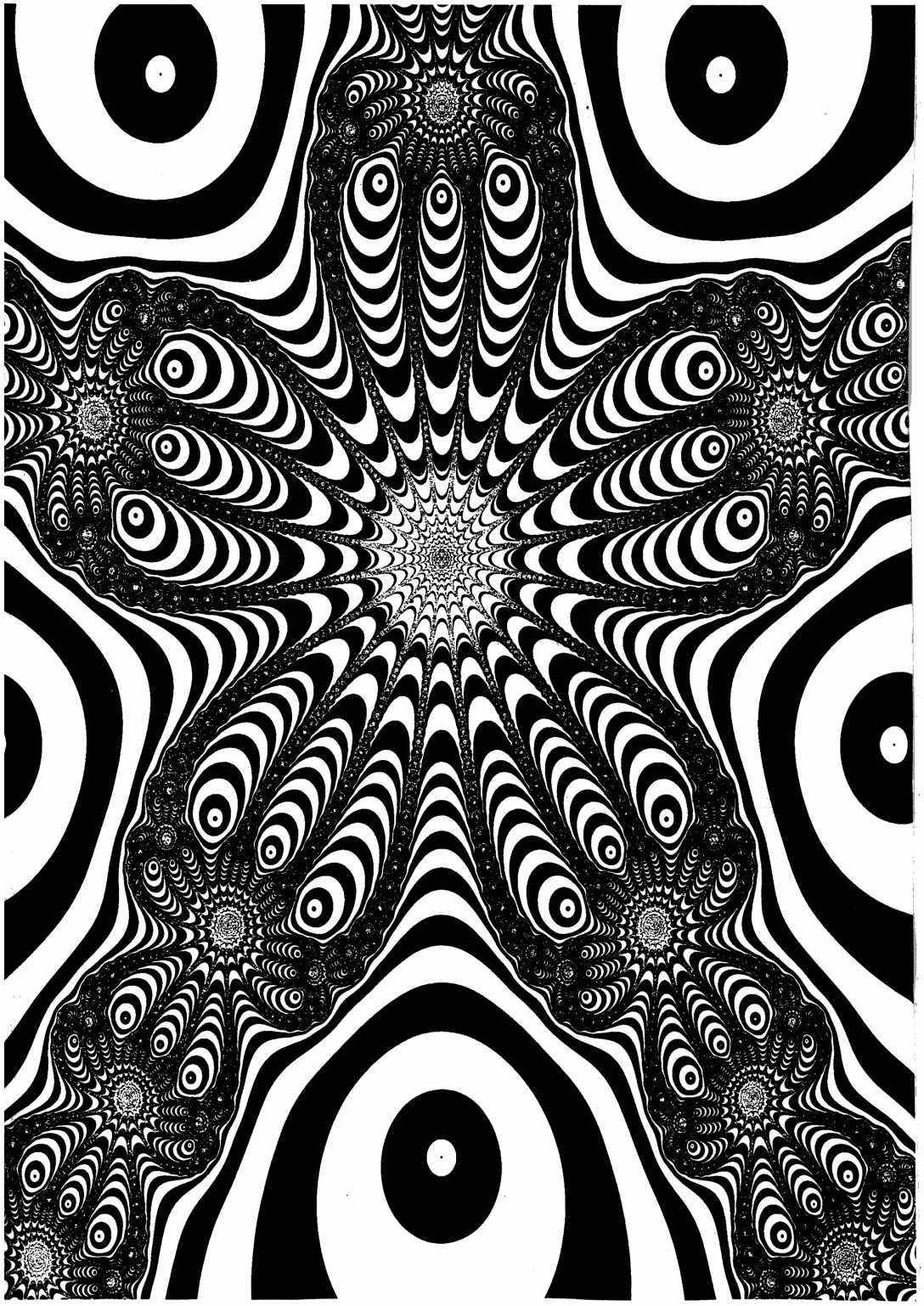
I have recently obtained some excellent fractal shareware to run on the Acorn Archimedes. This consists of three programs on Careware disk no 9, available for Norwich Computer Services (see ads in Acorn magazines) for around £6.00. The three programs are supplied in a compressed archived format and must be uncompressed using a utility on the same disk. Once uncompressed, they are run as applications in the normal way, although only two of the applications run in the desktop environment.

The first application is called (imaginatively!) *Fractal*; it is a fractal pattern generator with operates in the background via a small window. It installs itself on the icon bar and once started by double-clicking on the icon produces fractal-like patterns in a random manner, including some that are similar to Martin's Mappings. It is possible to specify one's own parameters to produce new patterns via a menu system and to save the patterns produced to disk.

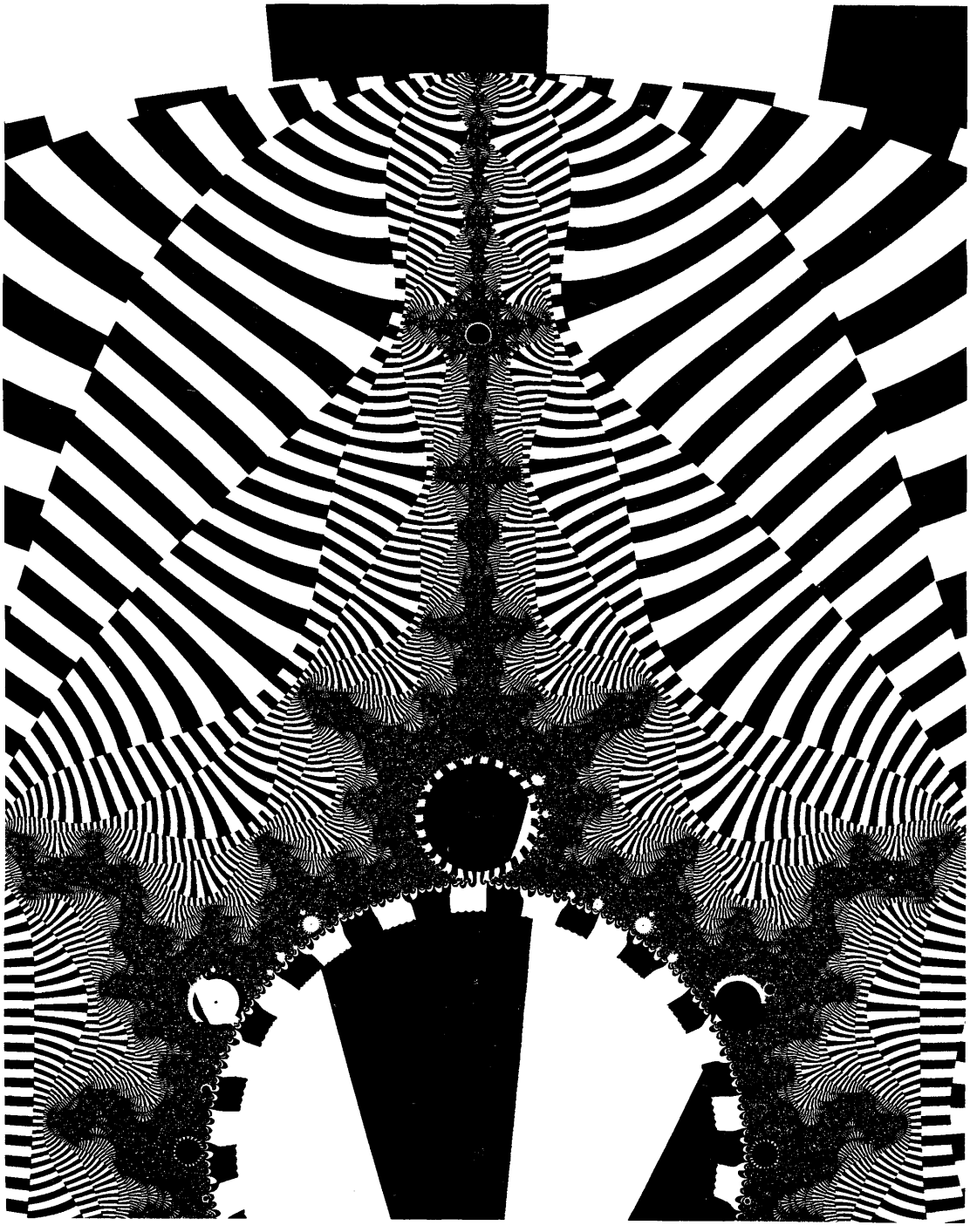
The second application is an extremely fast Mandelbrot/Julia set generator called SPEM. This program does not multitask via the desktop, but operates vis its own mouse/menu system. The usual Mandelbrot beetle is generated in about 15 seconds, and while more complex zooms take longer the program is reasonably fast. The program will operate to 64 digit precision, and it is possible to select the number of iterations used up to a maximum of 999. Zooming in is accomplished simply by selecting an area using the mouse and restarting calculation, and the results are displayed as they are calculated. The colours used may be selected by cycling through the 256 colour "palette". By simply holding one of the mouse buttons down amazing pseudo-random colour effects may be obtained. Parameters used can be saved to disk.

The third application is a 3D Mandelbrot/Julia generator/ray tracer. The application multitasks from the desktop and all functions are accessed via a menu system.

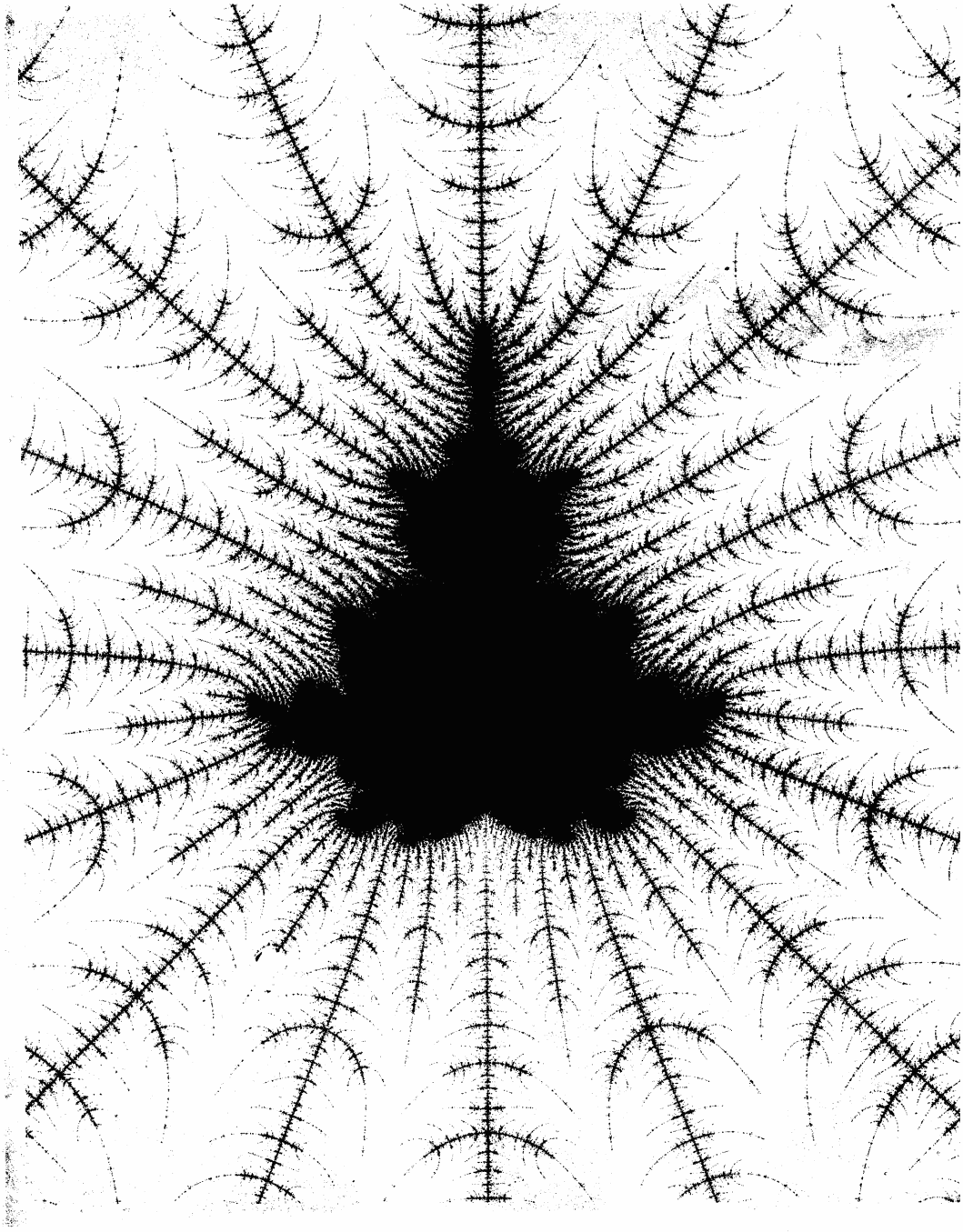
The program produces 16 grey scale ray-traced renderings of fractal scenes which may be saved as standard sprite files. Scenes are described by text files created in !Edit or similar. Parsing and ray-tracing are started simply by dragging the icon of the description file on to the application window. The simple command language used in the description files allows one to specify which type of scene is required (Mandelbrot/Julia), the complex number coordinates used, position and intensity of lamps, whether or not shadows are to be produced and much more besides. In common with other ray-tracers, image production takes a fairly long time, but one can watch the image appear in a window on the screen as it is calculated. It is possible to produce some quite stunning images with this program and the disk is worth obtaining for this alone.



$x^5 - 1$ *Newton Plot, by Ian Entwistle 1990*



A Mandelbrot Image by Cade Roux



A Mandelbrot Image by Cade Roux

Mandelbrot With Moving Colours

by José E. Murciano

This program draws the Mandelbrot set, with colours constantly pulsating.

```
DEFINT A-Z
DECLARE SUB ShiftPalette ()
DECLARE SUB WindowVals (WL%, WR%, WT%, WB%)
DECLARE SUB ScreenTest (EM%, CR%, VL%, VR%, VT%, VB%)
CONST FALSE = 0, TRUE = NOT FALSE
CONST MAXLOOP = 30, MAXSIZE = 1000000
DIM PaletteArray(15)
FOR I = 0 TO 15: PaletteArray(I) = I: NEXT I

WindowVals WLeft, WRight, WTop, WBottom

ScreenTest EgaMode, ColorRange, VLeft, VRight, VTop, VBottom

VIEW (VLeft, VTop)-(VRight, VBottom), 0, ColorRange
WINDOW (WLeft, WTop)-(WRight, WBottom)

LOCATE 24, 10: PRINT "Pulse una telca para salir.";

XLength = VRight - VLeft
YLength = VBottom - VTop
ColorWidth = MAXLOOP \ ColorRange

FOR Y = 0 TO YLength

   LogicY = PMAP(Y, 3)

   PSET (WLeft, LogicY)
   OldColor = 0

   FOR X = 0 TO XLength

      LogicX = PMAP(X, 2)

      Mandel X& = LogicX
      Mandel Y& = LogicY

      FOR I = 1 TO MAXLOOP
         RealNum& = Mandel X& * Mandel X&
         ImagNum& = Mandel Y& * Mandel Y&
         IF (RealNum& + ImagNum&) >= MAXSIZE THEN EXIT FOR
         Mandel Y& = (Mandel X& * Mandel Y&) \ 250 + LogicY
         Mandel X& = (RealNum& - ImagNum&) \ 500 + LogicX
      NEXT I

      PColor = I \ ColorWidth

      IF PColor <> OldColor THEN
         LINE -(LogicX, LogicY), (ColorRange - OldColor)
         OldColor = PColor
      END IF

      IF INKEY$ <> "" THEN END
   NEXT X

   LINE -(LogicX, LogicY), (ColorRange - OldColor)

   IF EgaMode THEN ShiftPalette
NEXT Y

DO

   IF EgaMode THEN ShiftPalette
LOOP WHILE INKEY$ = ""

SCREEN 0, 0
WIDTH 80
END

BadScreen:
   EgaMode = FALSE
   RESUME NEXT

SUB ScreenTest (EM, CR, VL, VR, VT, VB) STATIC
   EM = TRUE
   ON ERROR GOTO BadScreen
   SCREEN 8, 1
   ON ERROR GOTO 0

   IF EM THEN
      VL = 110: VR = 529
      VT = 5: VB = 179
      CR = 15

   ELSE
      SCREEN 1, 1
      VL = 55: VR = 264
      VT = 5: VB = 179
      CR = 3
   END IF

END SUB

SUB ShiftPalette STATIC
   SHARED PaletteArray(), ColorRange

   FOR I = 1 TO ColorRange
      PaletteArray(I) = (PaletteArray(I) MOD ColorRange) + 1
   NEXT I
   PALETTE USING PaletteArray(0)

END SUB

SUB WindowVals (WL, WR, WT, WB) STATIC
   CLS
   PRINT "Este programa hace la representacion grafica completa"
   PRINT "del Conjunto de Mandelbrot. Por defecto la ventana es"
   PRINT "de (-1000,625) hasta (250,-625). Un Zoon de una parte"
   PRINT "de la figura. Se efectua entrando nuevas coordenadas."
   PRINT
   PRINT "Pulse <ENTER> para la ventana por defecto. Pulse otra"
   PRINT "tecla para entrar la nuevas coordenadas de la ventana: ";
   LOCATE , 1
   Resp$ = INPUT$(1)

   IF Resp$ <> CHR$(13) THEN
      PRINT
      INPUT "X coordenada superior izquierda: ", WL
      DO
         INPUT "X coordenada inferior derecha: ", WR
         IF WR <= WL THEN
         PRINT "Coordenada derecha debe de ser mayor que c.izquierda."
         END IF
      LOOP WHILE WR <= WL
      INPUT "Y coordenada superior izquierda: ", WT
      DO
         INPUT "Y coordenada inferior derecha: ", WB
         IF WB >= WT THEN
         PRINT "Coordenada inferior debe ser mayor que c.superior."
         END IF
      LOOP WHILE WB >= WT
   ELSE
      WL = -1000
      WR = 250
      WT = 625
      WB = -625
   END IF
END SUB
```

Programa: Mandel1.BAS
Función: Desarrolar Conjunto de Mandelbrot
Autor: Dan Lufkin Colleage & José Enrique Murciano
Tiempo de ejecución: 4 minutos 17 segundos
Compilador: QBasic.
Con unas muy ligeras modificaciones se puede compilar con Turbo Basic de Borland.

Editorial

The article pile is now very low, and I would be grateful for some contributions for the next issue, preferably before the last week of April.

RTL offers Low Cost Video Standard Conversion

By the time this appears, RTL will be offering low cost digital video conversions on VHS from any tv standard to any other. Thus if you see a cassette you want that is only available in an alien standard, then send it to us and we'll send it back to you in your local standard. Please note that this is not a copying service – you get one cassette back! (Your original is erased and re-cycled.) You can't break the copyright laws by selling the original and keeping a standard-converted copy.

Anyone wishing to use this service, please write to us detailing what you want converted and we'll quote a price. If you are a provider of specialist low volume videos, then we would be pleased to negotiate a drop-ship arrangement to despatch your products to the rest of the world, converted to run at the addressee's location.

Announcements

Reader's Hall of Fame

What? No entries this time! Readers are reminded that if you get an article on fractals published in any mainstream magazine, please try and mention *Fractal Report*. Tell us, and we'll give your article a mention in this spot. To ensure that your mention of *Fractal Report* isn't edited out, include it as rem statements in a listing! These are usually reproduced photographically, and are therefore no so easily edited out.

Amygdala Issue 22

This issue had more of the flavour of *Fractal Report* than previous issues of *Amygdala* that I have seen. This is probably because the main article was one from a *Fractal Report*'s contributor – Dr Ian Entwistle. He discussed Julia sets where the maximum count is relatively small, and there is an unusual loop termination condition. This is when an iterate escapes from a square rather than the more usual circle. As usual he gives the essential lines of the program and also expands the iterations into real and imaginary parts for readers – although he then goes on to say that a "deliberate mistake" gives prettier pictures!

The second article was a description of a lecture given to various university departments on the Mandelbrot Set. The rest of the newsletter was snippets of news. The second issue of *Merz*, their ad sheet of $30 rectangles, included many repeats, but readers of *Fractal Report* may be interested in an *Amygdala* spin off *The Cellular Automatist*. This has just started with no 1, and it costs $25 per 10 issues (USA), $30 (Rest of North America), $45 (rest of world.) [Box 219, San Cristobal, New Mexico 87564, USA: Visa and MasterCard accepted.]

A Chaos Application

Mr V.A. Crawley of Hailsham sent us a cutting from *The Daily Telegraph* concerning an invention made by the paper's science correspondent. He suggests a new version of the "one time pad" where a sequence of numbers is used to code and decode a secret message. A disadvantage of this is that users must keep large volumes or computer files containing lists of numbers to encipher their messages. Mr Robert Matthews suggests that a fractal formula is iterated to produce a string of numbers, the formula and starting point being the key to the code.

Of course, the RND generator in any computer behaves in a similar manner. Fed the same seed, it generates the same string of numbers! (The system may vary between versions of the high level language using it – look at your manual to find out how to get the same sequence every time.)

Nevertheless, Mr Brian Winkel, editor of *Cryptologica* is quoted in *The Daily Telegraph* as being enthusiastic about Mr Matthews' idea.

Rec Suggests Answer to π Problem

Recreational and Educational Computing, an American newsletter produced by Dr Mike Ecker • 909 Violet Terrace • Clarks Summit • PA18411 • USA, mentions the Chudnovsky algorithm in its issue 5,8. This seems to be the answer to my query as to how one can work out the digits of π from starting at a particular digit without going back to the beginning. However the item just mentions the name and the fact that two readers independently wrote in and asked about it, and goes on to ask other readers to detail the algorithm. So if anyone out there knows about the Chudnovsky algorithm, please let us all in on the secret! It would be nice if *Fractal Report* could beat *Rec* to publishing details.

Also in this issue of *Rec* the phenomena of Fibonacci music was discussed, with a GW Basic program to play it. "Wallpaper for the Mind" was also covered(!), but to a lesser extent to the coverage already given in *Fractal Report*. Anyone interested in *Rec* is referred back to their advertisement in *Fractal Report* 12, page 17.

P.oem – 12 hours of PC video from a CD ROM

Mr P. Moon sent us some details of Michael F. Barnsley's fractal video compression system for the PC. The full kit costs about £5,000 and consists of a hard wired card plus software. However clients can send their films on video cassettes and have them encoded. They can then play them on their PCs using a video player program costing about £100. Single frames can also be compressed, thus complex pictures can be included in programs using trivial amounts of memory. The program to reproduce single frames (Targa formatted images or photographs) costs £50 and the cost for encoding a single frame is £12, or £50 for ten, or £125 for 50. An intriguing feature of the system is that when images are magnified, detail not present in the original is shown. Of course, the detail shown may not be present in the original scene.

Obviously these prices indicate a corporate market, but if these techniques could be developed to run in real time, there seems no reason why they shouldn't be used to allow real time video recording or transmission on audio media. With a mass market, the equipment could have a domestic price tag. Who would have guessed a few years ago that a multi-standard VCR and tv standards convertor would become a domestic item?

Machine Code Integer Mandelbrot for QL

C.G.H. Services have introduced an integer Mandelbrot program for the QL, written by Kenneth Murray. It costs £8.80 by post from C.G.H. Services, Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA. Please state media required. The program uses 32 bit integer arithmetic, and performs the usual zooms using a menu system. There are provisions for altering the iteration limit, saving the screen and printing it on 9,600 baud Epson printers. Coordinates can be entered by a moveable box or numerically. The program uses the QL's "true" windows and multitasking features. They sent me a copy for evaluation, but unfortunately my usual apalling problems with time made it impossible for me to review it as throughy as it deserves. I can say that it gave quick (for the QL) 8 colour displays. Within the constraints of time, I attempted to get the SuperQboard to print the screen, with "par use ser" but my NECP2200 just printed a series of line feeds.

They also offer 20 programs for the QL, together with two newsletters, *QL Leisure Review* and *QL Technical Review*, and a QL shareware service.

PC Mandelbrot Animation Program from Australia

Not content with sending us *Cell Block H* and *Neighbours*, we are now getting computer programs from those enterprising people on the other side of the world. A shareware example is from the Computer Gallery, PO Box 179, Maylands, WA6051, Australia. It is another Mandelbrot display program called *The Mandelbrot Viewer* and its animation facilities make it stand out from some others. It runs on all common video standards, and comes with a 25 page manual on disk. Registration is $Australian 20, in Australian currency on an Australian bank only. This entitles the owner to receive the latest version and go on a mailing list. I have sent copies to Frachaos and PC Star (both of whom have previously been mentioned in *Fractal Report*) and shareware enquiries can therefore be directed to either.

Fractint 15.1 Now Available

José Murciano kindly sent me a copy of *Fractint* 15.1 all the way from Spain. There are a number of new features, vastly improved presentation, a slight speed-up, and a manual of over 100 pages. I have sent copies to the aforementioned shareware dealers, to whom enquiries may be directed.