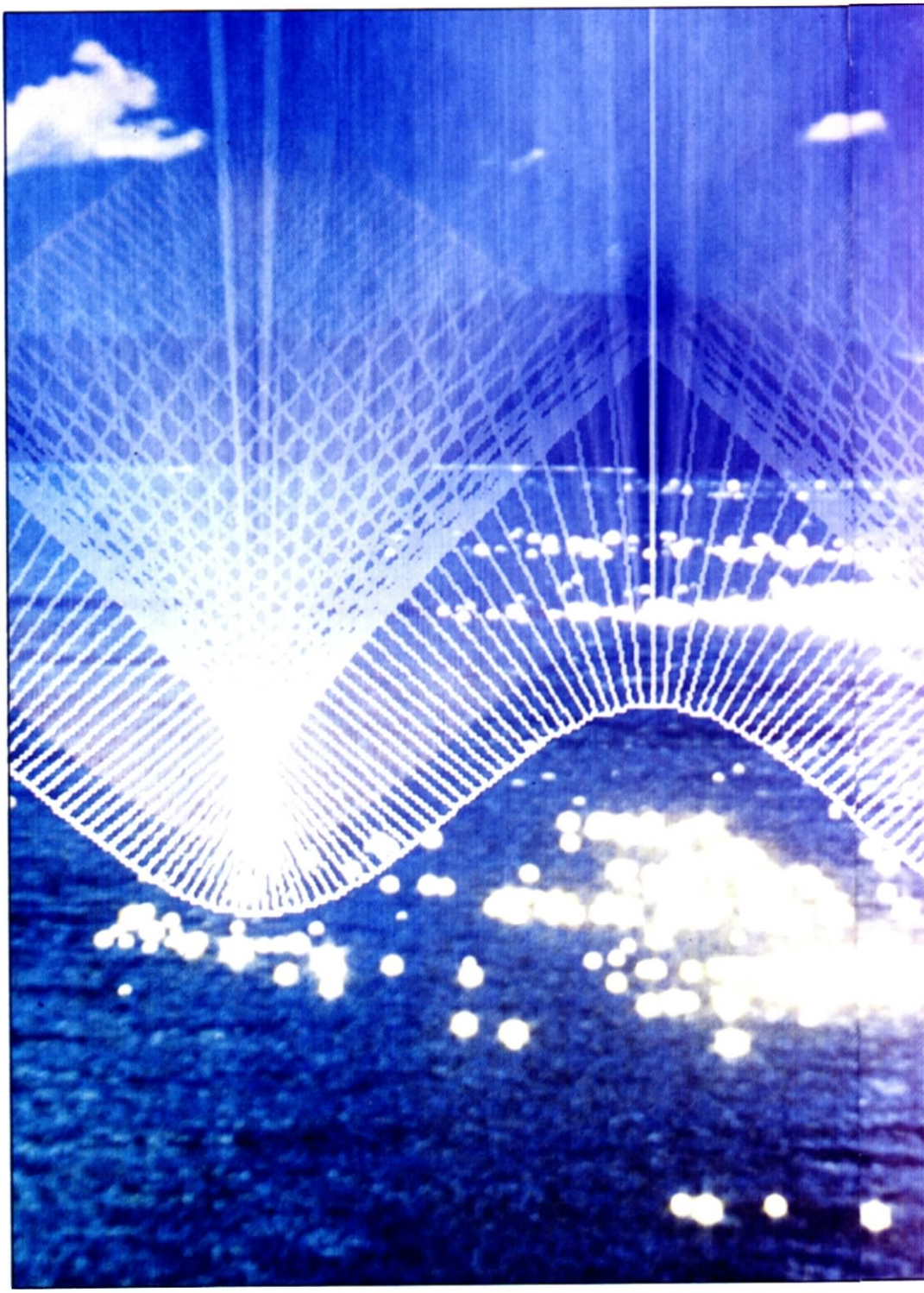
# PATTERNS FROM NATURE

**A surprising variety of patterns can be produced using very simple graphics routines that superimpose curves or build up patterns of dots. Here are a few ideas to try**



The way in which computers can be used to plot the orbit or trajectory of an object falling under gravity was described in the articles on pages 740 to 747 and 797 to 803. Such programs illustrate the simplest aspects of the old science of dynamics. Orbits can, however, generate patterns much richer and more interesting than the parabolas, circles and ellipses in which projectiles and planets move. This article will explain how some of these patterns can be produced by very elementary programming and graphics routines.

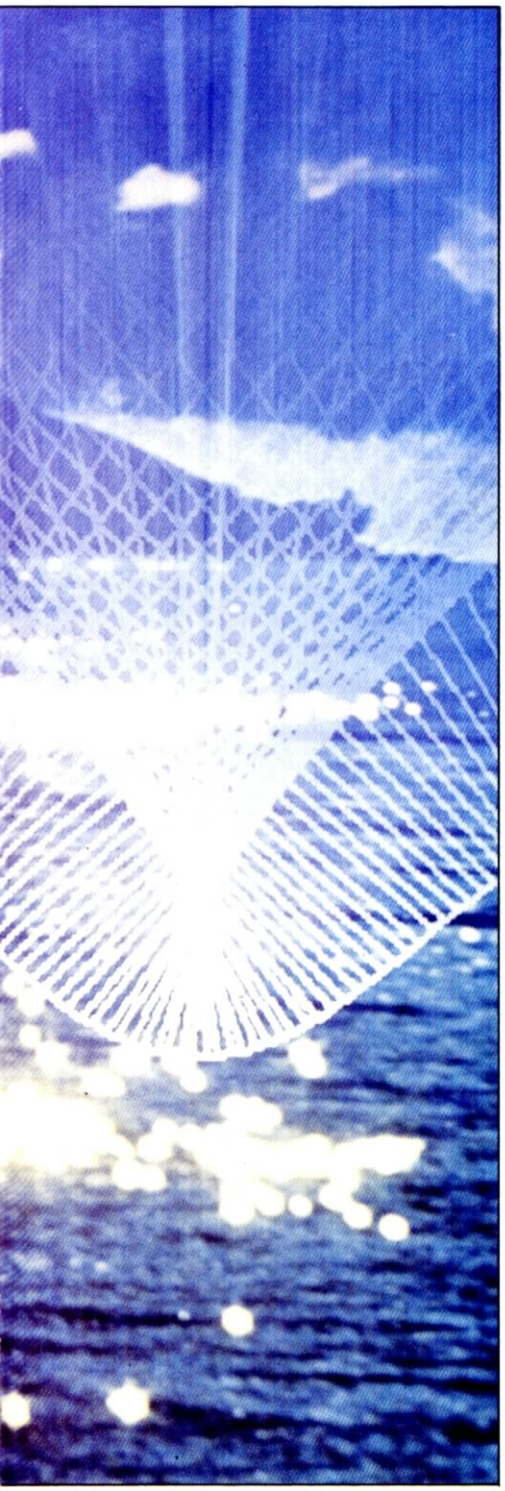
Nowadays dynamics is once again a rapidly developing field of research. One reason for this is the realization that there are mathematical ideas underlying dynamics that can be applied much more widely than simply to the motion of bodies acted on by forces. Optical scientists interested in the deviation of starlight by refraction in the atmosphere, industrial chemists studying the progress of a reaction, and biologists concerned with the growth of populations of competing species, all find themselves using the mathematics of dynamics. Another reason is that computers—by enabling simple operations to be repeated many times—have led to the discovery of structural complexity often unsuspected on the basis of the restricted calculations previously possible. A pattern or curve may need to be drawn many, many times before any structure begins to appear. The programs below show how this can be done.

All the programs for the Commodore need a Simons' Basic cartridge or *INPUT*'s hi-res utility. The programs for the Vic need a Super Expander cartridge.

## FAMILIES OF ORBITS

When assembled into a collection or *family*, orbits that are individually simple can display striking patterns. To see this for parabolas, enter and RUN the first program.

```
5 LET A$="": FOR N=1 TO 64: LET
  A$=A$+"□": NEXT N
10 BRIGHT 1: BORDER 0: INK 5: PAPER 0:
  CLS
20 LET NMAX=81
30 LET DELT=.05
```



```
40 LET SX=170/SQR 3: LET SY=175
50 FOR N=1 TO NMAX
60 LET A=PI*(-1+2*N/NMAX)
70 PLOT 128,80
80 FOR T=0 TO 3 STEP DELT
90 LET X=T*COS A: LET Y=T*(SIN A-T/2)
100 IF Y<=-.4 THEN GOTO 120
110 LET DX=SX*X+128: LET
    DY=SY*Y+80
111 IF DX<0 OR DX>255 OR DY<0 OR
    DY>175 THEN GOTO 120
115 DRAW DX-PEEK 23677,DY-PEEK 23678
117 PRINT AT 19,0;A$
120 NEXT T
130 NEXT N
```

```
10 HIRES 6,3:COLOUR 6,3
15 BLOCK 0,160,319,199,1
20 NM=81
30 DE=.05
40 SX=160/SQR(3):SY=200
50 FOR N=1 TO NM STEP 2
60 A=π*(-1+2*N/NM)
70 XX=160:YY=100
80 FOR T=0 TO 3 STEP DE
90 X=T*COS(A):Y=T*(SIN(A)-T/2)
100 IFY>-.4THENLINE XX,YY,SX*X+160,
    100-SY*Y,1:XX=SX*X+160:YY=
    100-SY*Y:GOTO110
105 T=3
110 NEXT T,N
130 GOTO 130
```

```
10 GRAPHIC 2:COLOUR 6,1,1,1
15 DRAW 1,0,000 TO 1023,800:
    PAINT 1,0,808
20 NM=81
30 DE=.05
40 SX=512/SQR(3):SY=1000
50 FOR N=1 TO NM STEP 2
60 A=π*(-1+2*N/NM)
70 POINT 0,512,512
80 FOR T=0 TO 3 STEP DE
90 X=T*COS(A):Y=T*(SIN(A)-T/2)
100 IFY>-.4THEN:DRAW 1 TO SX*X+512,
    512-SY*Y:GOTO110
105 T=3
110 NEXT T,N
130 GOTO 130
```

```
10 MODE0
20 NMAX=81
30 DELT=.05
40 SX=800/SQR(3):SY=1300
50 FOR N=1 TO NMAX
60 A=PI*(-1+2*N/NMAX)
70 MOVE 640,341
80 T=0:REPEAT
90X=T*COS(A):Y=T*(SIN(A)-T/2)
100 IF Y>-0.4 THEN DRAW SX*X+640,
    SY*Y+341
110 T=T+DELT:UNTIL Y<=-.4 OR T>3
120 NEXT
```

```
10 PMODE4,1:PCLS1:SCREEN1,0
20 NM=81:PI=4*ATN(1)
30 DE=.05
40 SX=160/SQR(3):SY=230
50 FOR N=1 TO NM
60 A=PI*(-1+2*N/NM)
70 DRAW"BM127,118"
80 FOR T=0 TO 3 STEP DE
90 X=T*COS(A):Y=T*(SIN(A)-T/2)
100 IF Y>-.4 THEN LINE-(SX*X+127,
    118-SY*Y),PRESET ELSE T=3
110 NEXT T
120 NEXT N
130 GOTO 130
```

This program simulates the paths of falling drops of water sprayed from the head of a garden sprinkler, or, in a more modern application, neutrons squirted from a thin pipe connected to a reactor. In this case the family of orbits consists of all the parabolic paths that emerge from a particular point in different directions but with the same speed. The pattern formed by this family is the outer curve which each orbit touches. This curve, called the *envelope* of the family, happens, itself, to be a parabola in this case (in gunnery it is called the 'bounding parabola' because it is the boundary of the region that can be reached by projectiles of a fixed initial speed). It is important to realize that the envelope is a property of the whole family of parabolic orbits and has no meaning for any single one of them. Thus envelopes are perfect illustrations of the fact that the whole can be

greater than the sum of its parts.

In the program, the equation for the orbits appears on Line 90. X is horizontal distance, Y is vertical distance, T is time (starting from zero at the instant of emission), and each orbit in the family is labelled by A, which is an angle giving the direction in which it starts out. There are NMAX or NM such directions; try changing the value of NMAX or NM in Line 20.

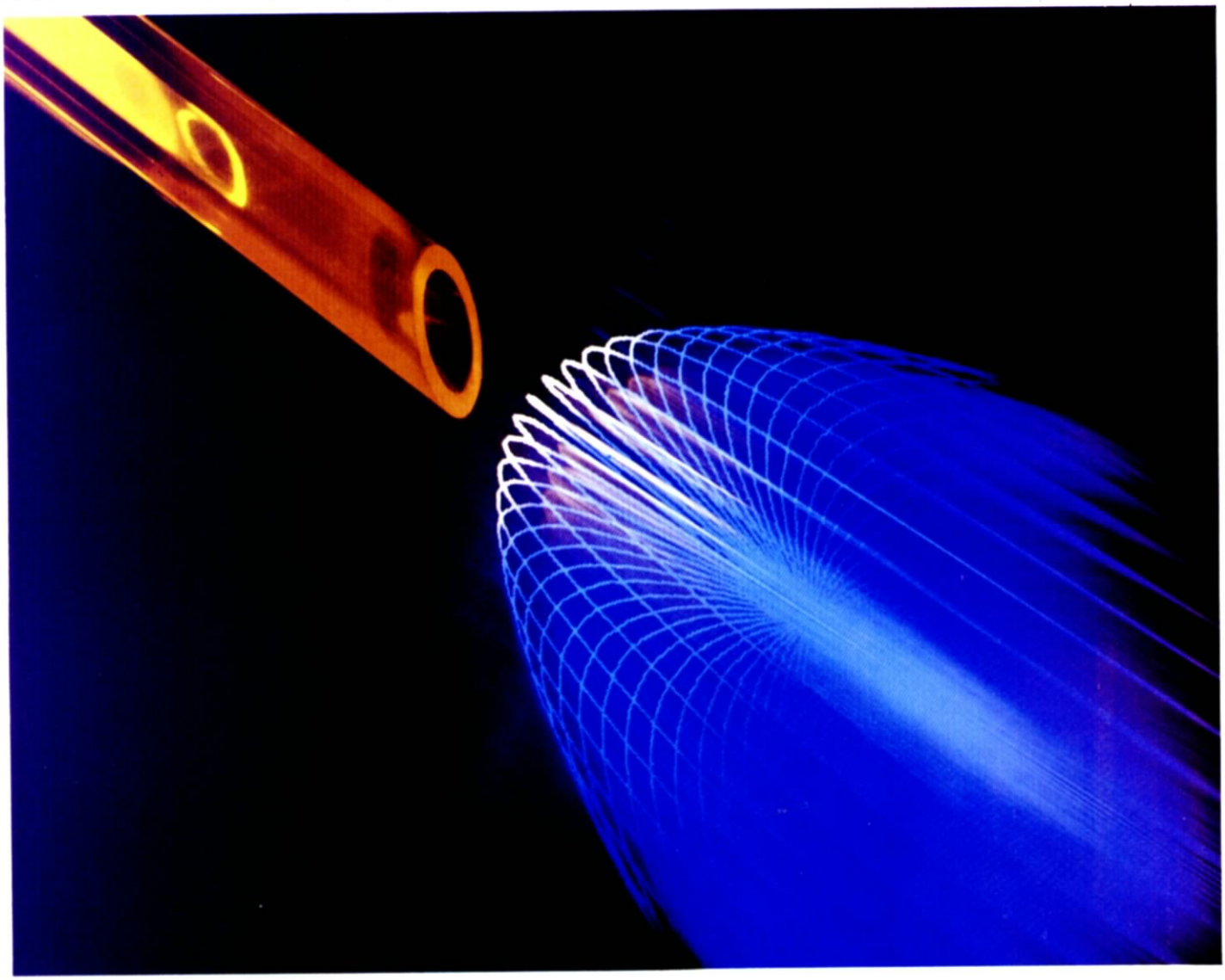
## FOCUSING

Even straight lines can form families with interesting envelopes, as the second program shows.

```
10 BORDER 0: INK 7: PAPER 0: CLS:
   LET N=2
30 LET Y0=80/N/N
40 PLOT 0,-6-Y0*2
50 FOR X=0 TO 255 STEP 2
60 LET Y=169-Y0-Y0*COS (N*2*PI*X/255)
70 DRAW X-PEEK 23677,175-Y-PEEK
   23678
80 LET XT=X+2*Y0*N*PI*Y/255*SIN (N*2*
   PI*X/255)
85 LET Y1=0: IF XT<0 THEN LET XT=0:
   LET Y1=Y+255*X/(2*N*PI*Y0*SIN (N*
   2*PI*X/255))
86 IF XT>255 THEN LET XT=255: LET
   Y1=Y-(255-X)*255/(2*N*PI*Y0*SIN
   (N*2*PI*X/255))
90 DRAW XT-PEEK 23677,175-Y1-PEEK
   23678
100 PLOT X,175-Y
110 NEXT X
120 GOTO 120
```

```
10 HIRES 6,3:COLOUR 6,3
20 N=2
30 Y0=80/N/N
40 XX=0:YY=INT(160-Y0*2)
50 FOR X=0 TO 319 STEP 2
60 Y=160-Y0-Y0*COS(N*2*π*X/319)
70 LINE XX,YY,X,Y,1:XX=X:YY=Y
80 XT=X+2*Y0*N*π*Y/319*SIN(N*2*π*
   X/319)
85 Y1=0:IF XT<0 THEN XT=0:Y1=Y+
   199*X/(2*N*π*Y0*SIN(N*2*π*X/199))
86 IF XT>319 THEN XT=319:Y1=Y-
   (199-X)*199/(2*N*π*Y0*SIN(N*2*π*
   X/199))
90 LINE XX,YY,XT,Y1,1
110 NEXT X
120 GOTO 120
```

```
10 GRAPHIC 2:COLOR 6,1,1,1
20 N=2
30 Y0=200/N/N
40 XX=0:YY=INT(512-Y0*2)
50 FOR X=0 TO 1023 STEP 20
60 Y=512-Y0-Y0*COS(N*2*π*X/1023)
70 DRAW 1,XX,YYTO X,Y:XX=X:YY=Y
90 XT=X+2*Y0*N*π*Y/1023*SIN(N*2*π*
```



```
X/1023)
85 Y1=0:IF XT<0 THEN XT=0:Y1=Y+
   1023*X/(2*N*π*Y0*SIN(N*2*π*X/1023))
86 IF XT>1023 THEN XT=1023:Y1=Y-
   (1023-X)*1023/(2*N*π*Y0*SIN(N*2*π*
   X/1023))
90 DRAW 1,XX,YYTO XT,Y1
110 NEXT X
120 GOTO 120
```

```
10 MODE0
20 N=2
30 Y0=400/N/N
40 MOVE0,Y0*2+30
50 FOR X=0 TO 1283 STEP 9
60 Y=Y0+30+Y0*COS(N*2*PI*X/1279)
70 DRAW X,Y
80 XTOP=X+2*Y0*N*PI*(1023-Y)/1279*
   SIN(N*2*PI*X/1279)
90 DRAW XTOP,1023
100 MOVEX,Y
110 NEXT
```

```
10 PMODE4,1:PCLS1:SCREEN1,0
20 N=2:PI=4*ATN(1)
30 Y0=80/N/N
40 DRAW"BM0,"+STR$(INT(186-Y0*2))
50 FOR X=0 TO 255 STEP 2
60 Y=186-Y0-Y0*COS(N*2*PI*X/255)
70 LINE-(X,Y),PSET
80 XT=X+2*Y0*N*PI*Y/255*SIN(N*2*PI*X/
   255)
85 Y1=0:IF XT<0 THEN XT=0:YZ=Y+
   255*X/(2*N*PI*Y0*SIN(N*2*PI*X/255))
86 IF XT>255 THEN XT=255:Y1=Y-
   (255-X)*255/(2*N*PI*Y0*SIN(N*2*PI*X/
   255))
90 LINE-(XT,Y1),PRESET
100 DRAW"BM"+STR$(X)+","+STR$
   (INT(Y))
110 NEXT
120 GOTO120
```

This program simulates light rays bent by refraction through a wavy, curved surface, for example sunlight refracted by ripples on water in a swimming pool. The family of orbits consists of all the straight lines at right angles to a wavy curve of sine form. For light rays the envelope is the curve corresponding to *focusing*. Focusing is particularly intense near the troughs (minima) of the sine wave, where the envelope has sharp points called *cusps*. These cusps are the bright points of light you see amongst the ripples. The existence of cusps is predicted by the recently developed mathematics of envelopes, called *catastrophe theory* (whose dramatic name originated in applications of the same mathematics to the collapse of bridges and the capsizing of ships).

In the program, the wavy initial curve is specified in Line 60 and the rays that start out from it are defined in Line 80. The number of troughs of the wavy curve is N; try changing the value of N in Line 20 (N = 1 is especially recommended).

Trajectories that are themselves wavy sine curves are assembled into a family in the third program.

```
10 BORDER 0: PAPER 0: INK 7: CLS
20 LET QM=SQR (2*LN (3))
30 FOR Q=-QM TO QM*1.001 STEP QM/30
40 PLOT 0,75+Q*75/QM
50 FOR T=0 TO 3*PI STEP .2
60 LET X=Q*COS (EXP (-Q*Q/2)*T)
70 DRAW (T*240/3/PI)-PEEK
   23677,75+(X*75/QM)-PEEK 23678
80 NEXT T
90 NEXT Q
```

```
10 HIRES 0,1:COLOUR 1,6:MULTI 4,3,7
20 QM=SQR(2*LOG(3))
30 FOR Q=-QM TO QM*1.001 STEP
   QM/30
40 XX=0:YY=INT(100-Q*90/QM)
50 FOR T=0 TO 3*π STEP .2
60 X=Q*COS(EXP(-Q*Q/2)*T)
70 LINE XX,YY,T*17/3*π,100-X*90/QM,RND
   (1)*3+1
75 XX=T*17/3*π:YY=100-X*90/QM
80 NEXT T,Q
90 GOTO 90
```

```
10 GRAPHIC 1:COLOR 6,1,3,7
20 QM=SQR(2*LOG(3))
30 FOR Q=-QM TO QM*1.001 STEP
   QM/30
40 POINT 0,0,INT(512-Q*500/QM)
50 FOR T=0 TO 3*π STEP .6
60 X=Q*COS(EXP(-Q*Q/2)*T)
70 DRAW RND(1)*3+1 TO T*116/ 3*π,
   512- X* 500/ QM
80 NEXT T,Q
90 GOTO 90
```

```
10 MODE0
20 QM=SQR(2*LN(3))
30 FOR Q=-QM TO QM*1.001 STEP QM/30
40 MOVE 0,Q*450/QM+500
50 FOR T=0 TO 3*PI STEP .2
60 X=Q*COS(EXP(-Q*Q/2)*T)
70 DRAW T*1200/3/PI,X*450/QM+500
80 NEXT
90 NEXT
```

```
10 PMODE4,1:PCLS1:SCREEN1,0
20 QM=SQR(2*LOG(3)):PI=4*ATN(1)
30 FOR Q=-QM TO QM*1.001 STEP
   QM/30
40 DRAW"BM0,"+STR$(INT(100-Q*90/
   QM))
50 FOR T=0 TO 3*PI STEP .2
60 X=Q*COS(EXP(-Q*Q/2)*T)
70 LINE-(T*240/3/PI,100-X*90/QM),
   PRESET
80 NEXT T
90 NEXT Q
100 GOTO 100
```

This program simulates light rays passing along a thin glass optical fibre whose refractive index varies across its width, or (on a much smaller scale) electrons winding between planes of atoms in a crystal placed in the beam of an electron microscope. The family consists of orbits starting out parallel to each other at the left-hand edge of the screen; each orbit undulates about the central line at a rate that depends on its starting point. Although this family is very different from the last program, the envelope curves also display intense focusing at cusp catastrophe points.

In the program, orbits are specified in Line 60 by giving their distance X from the centre line at time T. Different orbits in the family are labelled by Q. There are 30 orbits in the program as written; to get more or fewer, change the number at the end of Line 30.

## DOT PATTERNS

Assembling orbits into families is not the only method of obtaining interesting patterns. Another way is to follow the orbits for *very long times*, with the result that considerable complexity can develop even when the mathematics of the orbit is relatively simple. In displaying these orbits it is not usually advisable to plot the continuous curve giving the position at every instant, because over long times this would just fill the screen with a mess like tangled wool. Instead the orbit is plotted at regular intervals (once every second, for example), as though the trajectory were viewed in the flashing light of a stroboscope. Examples of the resulting dot patterns are given in the next three programs. The position of dots on the screen does not always correspond to the real, spatial position of physical objects whose motion the programs simulate; sometimes it is an abstract representation, in which horizontal screen position corresponds to position and vertical screen position to speed.

The fourth program will take several minutes to RUN.

```
10 BORDER 0: PAPER 0: INK 7: CLS
30 LET A=76.11
40 LET ALF=A*PI/180: LET C=COS (ALF)
50 LET S=SIN (ALF)
60 LET NMAX=200
70 LET M=52
80 FOR J=1 TO M
90 LET X=0: LET Y=J/M
100 FOR N=1 TO NMAX
110 LET W=X
120 LET X=X*C-(Y-X*X)*S: LET
    Y=W*S+(Y-W*W)*C
130 IF ABS (X)>4 OR ABS (Y)>4 THEN
    GOTO 860
135 IF X>1 OR Y>1 THEN GOTO 150
140 PLOT X*128+128,Y*85+85
150 NEXT N
160 NEXT J
```

```
10 HIRES 0,1:COLOUR 1,6:MULTI 5,3,7
20 A=76.1
30 AL=A*π/180:C=COS(AL)
40 S=SIN(AL)
50 NM=200
60 M=52
70 FOR J=1 TO M
80 X=0:Y=J/M:CL=RND(1)*3+1
90 FOR N=1 TO NM
100 W=X
110 X=X*C-(Y-X*X)*S:Y=W*S+(Y-
    W*W)*C
120 IF ABS(X)>4 OR ABS(Y)>4 THEN 160
125 IF ABS(Y)>1 OR ABS(X)>1 THEN 140
130 PLOT 80+X*79,100-Y*99,CL
140 NEXT N,J
150 GOTO 150
```

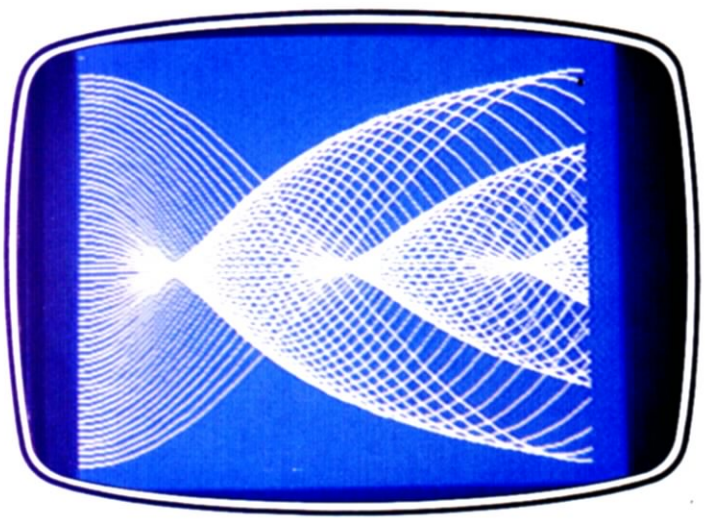
```
10 GRAPHIC 1:COLOR 6,1,3,7
20 A=76.1
30 AL=A*π/180:C=COS(AL)
40 S=SIN(AL)
50 NM=200
60 M=52
70 FOR J=1 TO M
80 X=0:Y=J/M:CL=RND(1)*3+1
90 FOR N=1 TO NM
100 W=X
110 X=X*C-(Y-X*X)*S:Y=W*S+
    (Y-W*W)*C
120 IF ABS(X)>4 OR ABS(Y)>4 THEN 160
125 IF ABS(Y)>1 OR ABS(X)>1 THEN 140
130 POINT CL,512+X*512,512-Y*512
140 NEXT N,J
150 GOTO 150
```

```
10 MODE0
20 A=76.11
30 ALF=A*PI/180:C=COS(ALF)
40 S=SIN(ALF)
50 NMAX=200
60 M=52
70 FOR J=1 TO M
80 X=0:Y=J/M
90 FOR N=1 TO NMAX
100 W=X
110 X=X*C-(Y-X*X)*S:Y=W*S+
    (Y-W*W)*C
120 IF ABS(X)>4 OR ABS(Y)>4 THEN END
130 PLOT 69,X*640+640,Y*512+512
140 NEXT
150 NEXT
```

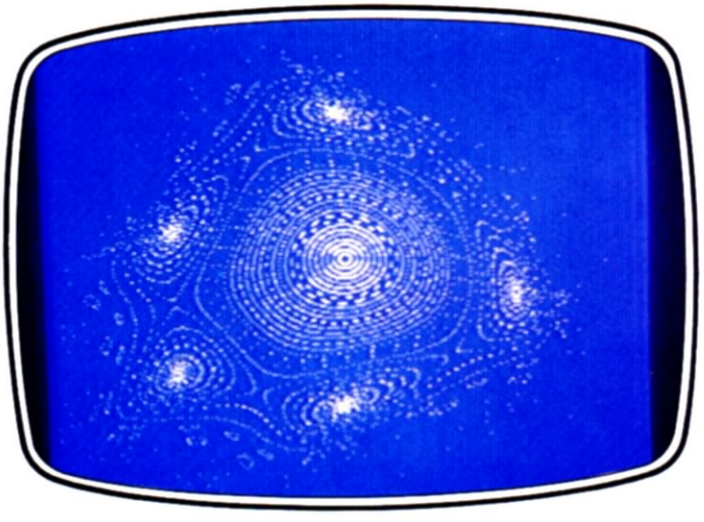
```
10 PMODE4,1:PCLS1:SCREEN1,0
20 A=76.11:PI=4*ATN(1)
30 AL=A*PI/180:C=COS(AL)
40 S=SIN(AL)
50 NM=200
60 M=52
70 FOR J=1TOM
80 X=0:Y=J/M
90 FORN=1TONM
100 W=X
110 X=X*C-(Y-X*X)*S:Y=W*S+
    (Y-W*W)*C
120 IF ABS(X)>4 OR ABS(Y)>4 THEN 160
125 IF ABS(Y)>1 OR ABS(X)>1 THEN140
130 PRESET(128+X*128,96-Y*96)
140 NEXT N
150 NEXT J
160 GOTO160
```

This program simulates motion of subatomic particles (such as protons or electrons) in an accelerator, or the windings of a line of force in the magnetic bottle of a fusion power device. Mathematically, what the program does is to take a series of initial points and move them about the screen by repeating a rule that says: 'rotate about the middle of the screen by a fixed angle, apart from a slight modification'. Without the 'slight modification' the orbits of the points would all be circles, and indeed those near the centre are roughly circles. But the modification has a dramatic effect on points initially far from the centre: their orbits may be a series of 'islands', or they may escape from the screen altogether. More refined computer graphics, using high magnification, reveals tiny islands everywhere, distributed amongst the large ones.

In the program, the number of initial points is M, specified in Line 60; if you get tired of waiting for the picture you can reduce this number. Each of these points is plotted for NMAX or NM repetitions of the transform-



**Focusing light in a glass fibre**



**Islands of dots or lines of force**

ation rule; NMAX or NM is specified in Line 5Ø. The rule itself is contained in Lines 1ØØ and 11Ø, which describe how the horizontal and vertical screen coordinates X and Y are altered at each repetition. The angle of the unmodified rotation, in degrees, is A, specified in Line 2Ø; you should experiment with different values of A (try 9Ø).

An unexpected pattern made of dots generated by a single initial point is produced by the next program which you should now enter and run:

```
20 LET K=51.3: BORDER Ø: INK 7: PAPER Ø:
   CLS
30 LET X=1/PI: LET P=Ø
40 LET A=1/SQR 5
50 FOR N=1 TO 10000
60 LET Y=X-.5: LET X=X+A-INT (X+A)
70 LET P=P-Y
80 PLOT X*255,P*K+85
90 NEXT N
```

```
10 HIRES Ø,1:COLOUR 1,6:
   MULTI 5,3,7:K=6Ø
20 X=1/π:P=Ø
30 A=1/SQR(5)
40 FOR N=1 TO 10000
50 Y=X-.5:X=X+A-INT(X+A)
60 P=P-Y
70 PLOT X*159,1ØØ-P*K,RND
   (1)*3+1
80 NEXT N
90 GOTO 90
```

```
10 GRAPHIC 1:COLOR 6,1,3,7:
   K=300
20 X=1/π:P=Ø
30 A=1/SQR(5)
40 FOR N=1 TO 10000
50 Y=X-.5:X=X+A-INT(X+A)
60 P=P-Y
70 POINT RND(1)*3+1,X*1Ø23,
   512-P*K
80 NEXT N
90 GOTO 90
```

```
10 MODEØ:K=3ØØ
20 X=1/PI:P=Ø
30 A=1/SQR(5)
40 FOR N=1 TO 10000
50 Y=X-.5:X=X+A-INT(X+A)
60 P=P-Y
70 PLOT 69,X*1279,P*K+512
80 NEXT
```

```
10 PMODE4,1:PCLS1:SCREEN1,Ø:
   K=6Ø
20 X=1/(4*ATN(1)):P=Ø
30 A=1/SQR(5)
40 FOR N=1 TO 10000
50 Y=X-.5:X=X+A-INT(X+A)
60 P=P-Y
70 PRESET(X*255,128-P*K)
80 NEXT N
90 GOTO9Ø
```

This program is an abstract simulation of *resonance*, where the frequencies of two physical effects may clash. For example, one frequency might be that of an asteroid's motion round the sun, and the other might be the frequency with which the asteroid is disturbed by the gravitational pull of the planet Jupiter. In this case an important question is: do the disturbing pulls mount up and throw the asteroid out of its orbit, or do they force it into a stable orbit? The answer is that this depends on the *ratio* of the two frequencies (that is, on one divided by the other). The particles in Saturn's rings are affected by similar forces.

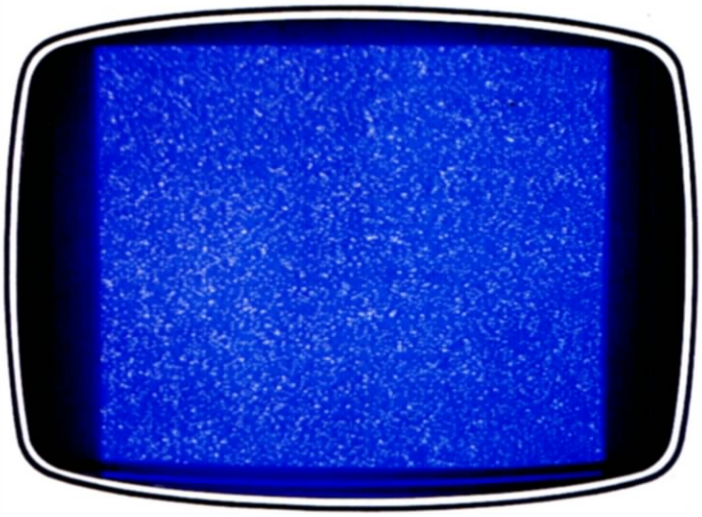
In the program, the ratio is called A and its value (which must be less than 1) is specified in Line 3Ø. The number of repetitions of the resonance transformation is 10000 and is specified at the end of Line 4Ø; reduce this value if you get tired of waiting. The transformation itself is on Lines 5Ø and 6Ø, and is the rule for changing the horizontal (X) and vertical (P) positions of the dot on the screen.

As written, the delicate curtain pattern is produced by A=1/SQR(5) which equals 0.4472136. This is not the ratio of two whole numbers (it is an 'irrational number') and corresponds to *nonresonance*. To get resonance you should try a number that is the ratio of two whole numbers, such as A=9/2Ø which equals 0.45. Then try a different irrational number, such as A=1/PI. (If you want to reduce the vertical scale to display more of the picture, diminish the value of K.

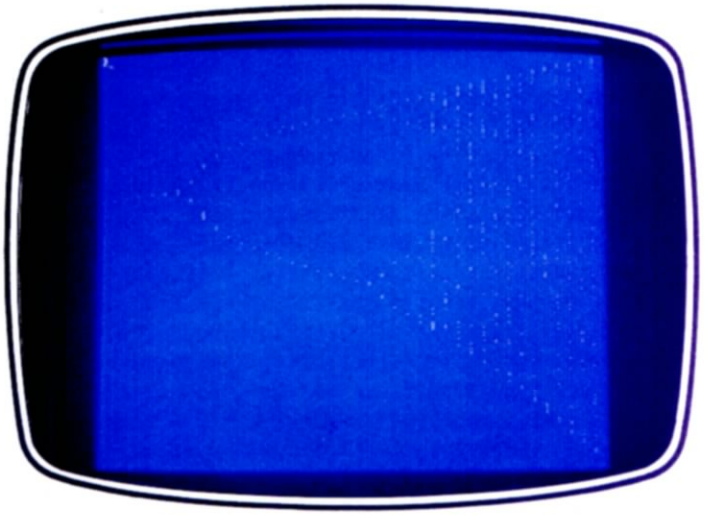
## CHAOS

It is a remarkable fact that some simple rules produce *no pattern at all*. Such an example is the program below which you should enter and RUN:

```
10 BORDER Ø: PAPER Ø: INK 7: CLS
20 LET X=1/PI
30 LET Y=1/PI
40 FOR M=1 TO 10000
50 LET X=X+Y-INT (X+Y)
60 LET Y=X+Y-INT (X+Y)
70 PLOT X*255,Y*175
80 NEXT M
```



**Some simple rules produce chaos**



**Population changes in a fish pond**

```
10 HIRES 0,1:COLOUR 6,6:
   MULTI 5,3,7
20 X=1/π
30 Y=1/π
40 FOR M=1 TO 10000
50 X=X+Y-INT(X+Y)
60 Y=X+Y-INT(X+Y)
70 PLOT X*159,199-Y*199,
   RND(1)*3+1
90 NEXT M
90 GOTO 90
```

```
10 GRAPHIC 1:COLOR 6,1,3,7
20 X=1/π
30 Y=1/π
40 FOR M=1 TO 10000
50 X=X+Y-INT(X+Y)
60 Y=X+Y-INT(X+Y)
70 POINT RND(1)*3+1,X*1023,
   1023-Y*1023
80 NEXT M
90 GOTO 90
```

```
10 MODE0
20 X=1/PI
30 Y=1/PI
40 FOR M=1 TO 10000
50 X=X+Y-INT(X+Y)
60 Y=X+Y-INT(X+Y)
70 PLOT69,X*1279,Y*1023
80 NEXT
```

```
10 PMODE4,1:PCLS1:SCREEN1,0
20 PI=4*ATN(1):X=1/PI
30 Y=1/PI
40 FORM=1TO10000
50 X=X+Y-INT(X+Y)
60 Y=X+Y-INT(X+Y)
70 PRESET(X*255,192-Y*191)
80 NEXT M
90 GOTO 90
```

This program simulates the erratic bouncing of metal spheres in a pinball machine, or the motion of molecules in a gas, or of a roulette wheel, or indeed any dynamic system whose orbits are so unpredictable as to be indistinguishable from what would be generated by purely random processes. And yet the ten thousand points on the screen are not the result of random sprinkling but are generated by the program from a single point by repeating a rule that is purely deterministic—that is, it contains no random element.

The rule is based on regarding the screen as the 'unit square' on which horizontal distance X and vertical distance Y range from zero to one, and operates in three stages, constituting Lines 50 and 60 of the program. First, the X and Y coordinates of a point are added to produce a new X; second, this new X is added to Y to produce a new Y; third, if either of the new X or Y lies outside the range 0 to 1, an appropriate whole number is subtracted in order to bring the transformed point back into the unit square (in the program this subtraction is implemented by the INT function in Lines 50 and 60).

## FISH POPULATIONS

How does the population of fish in a pond change over many generations? This depends on the rule that determines how the population changes from one generation to the next. Such a rule must incorporate both the tendency of the population to increase because each set of parents produces more than two offspring, and the tendency of the population to decrease when it gets so large that the finite food supply in the pond cannot sustain it. Depending on the precise balance between these two dependencies a fish population may settle down to a stable fixed value, or alternate regularly between two or more values, or change apparently randomly between successive generations.

The final program employs graphics and sound to illustrate these different possibilities. Horizontal screen position, denoted by A, corresponds to the balance between breeding and food supply and hence to the rule relating successive generations. Vertical screen position, denoted by Y, corresponds to the population, represented by a point jumping up or down at each generation. The population Y is plotted on the screen only when it has settled down to its stable value or set of alternating values, called the *attractor set*. But the way in which the population homes in on the attractor can be heard because the program makes the computer emit a sound whose pitch (frequency) is proportional to the population.

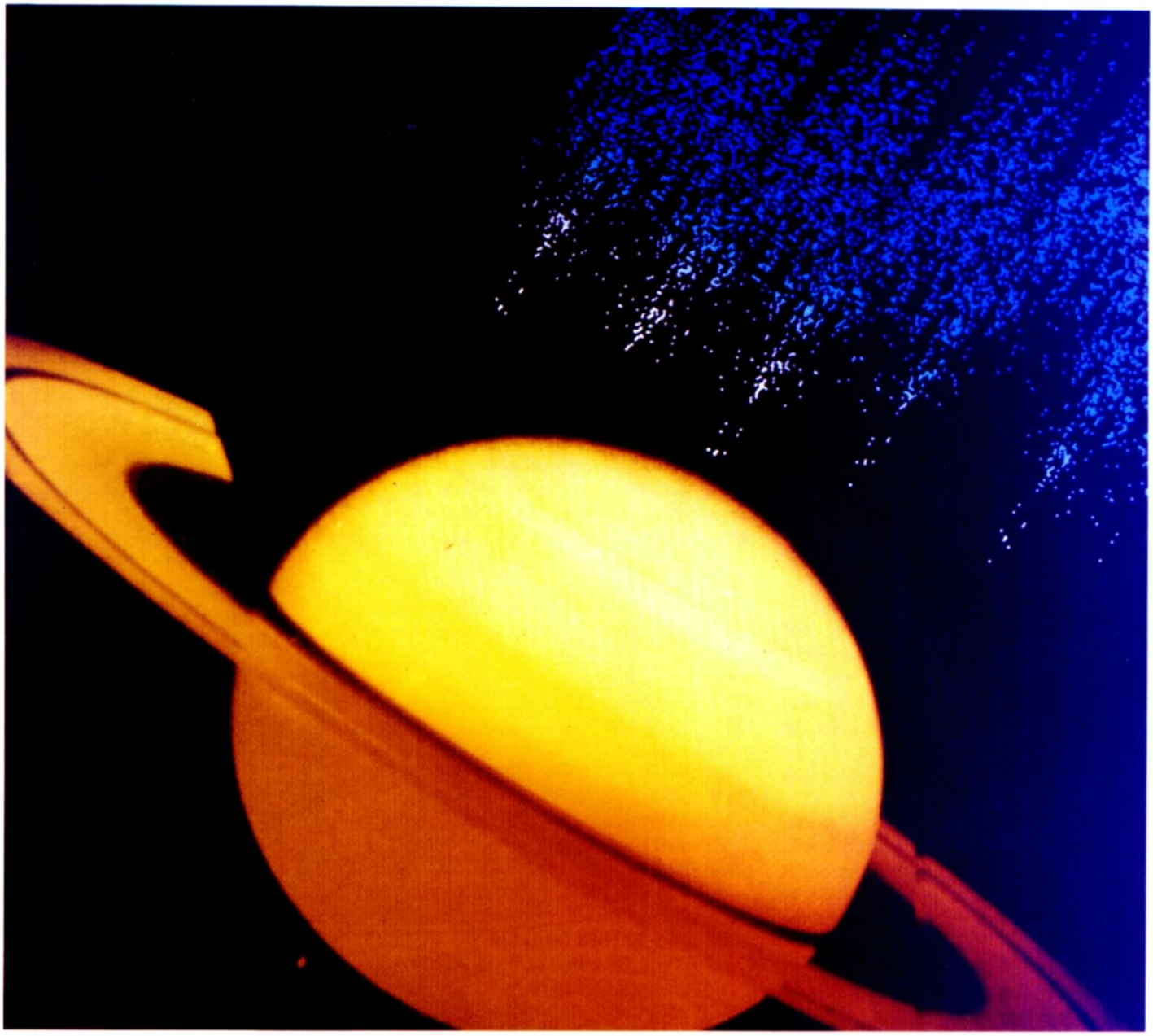
```
10 BORDER 0: INK 7: PAPER 0: CLS
20 LET S=.03 LET NMIN=50: LET
   NMAX=80
30 FOR A=2.8 TO 4 STEP S
40 LET Y=1/PI
50 FOR N=1 TO NMAX
60 LET Y=A*Y*(1-Y)
70 IF N>NMIN THEN PLOT 255*
   (A-2.8)/1.2,Y*175
80 BEEP .0075,Y*20
90 NEXT N
100 NEXT A
```

```
10 HIRES 0,1:COLOUR 1,6:
   MULTI 5,3,7
15 POKE 54296,15:POKE 54277,64
20 S=1/160:NN=50:NX=80
30 FOR A=2.8 TO 4 STEP S
40 Y=.25/ATN(1)
45 POKE 54276,33
50 FOR N=1 TO NX
60 Y=A*Y*(1-Y)
70 IF N>NN THEN PLOT 159*(A-2.8)/1.2,
   199-Y*199,RND(1)*3+1
80 POKE 54273,1+255*Y
90 NEXT N
95 POKE 54276,32
100 NEXT A
110 GOTO 110
```

```
10 GRAPHIC 1:COLOR 6,1,3,7
15 POKE 36878,15
20 S=.01:NN=50:NX=80
30 FOR A=2.8 TO 4 STEP S
40 Y=.25/ATN(1)
50 FOR N=1 TO NX
60 Y=A*Y*(1-Y)
70 IF N>NN THEN:POINT RND(1)*3+1,
   1023*(A-2.8)/1.2,1023-Y*1023
80 POKE 36876,128+127*Y
90 NEXT N
95 POKE 36876,0
100 NEXT A
110 GOTO 110
```

```
10 MODE0
20 S=.03:NMIN=50:NMAX=80
30 FOR A=2.8 TO 4 STEP S
40 Y=1/PI
50 FOR N=1 TO NMAX
60 Y=A*Y*(1-Y)
70 IF N>NMIN THEN PLOT69,1279*
   (A-2.8)/1.2,Y*1023
80 SOUND1,-15,255*Y,1
90 NEXT
100 NEXT
```

```
10 PMODE4,1:PCLS1:SCREEN1,0
20 S=1/127:NN=50:NX=80
30 FORA=2.8 TO4 STEP S
40 Y=.25/ATN(1)
50 FORN=1TONX
60 Y=A*Y*(1-Y)
70 IF N>NN THENPRESET(255*(A-2.8)/1.2,
   192-Y*191)
80 SOUND1+255*Y,1
90 NEXT N
100 NEXT A
110 GOTO110
```



It is clear that the attractor undergoes drastic changes as the amount of food is slowly altered. To start with (on the left of the screen), the attractor is a single point, indicating an eventually stable fish population. This means there is enough food for the fish to live healthily and breed well. If the population increases too much there will be less food to go around so some fish will die of starvation. This leaves more food for the survivors so they grow and breed well again. Eventually the numbers in each generation settle down to a constant stable population. In the program you'll hear the sound oscillating between each generation but settling down to a stable value each time.

Suddenly, at a certain value of A, as the food is increased past a certain level, the attractor is seen and heard to branch into two or *bifurcate*, indicating a population eventually alternating between two values. Later, with even more food, the attractor bifurcates again, indicating four alternating population values. More and more divisions occur, forming an infinite sequence of which only the first few are resolved by this program. The successive bifurcations accumulate at a finite A value, corresponding to a fish population that fluctuates among infinitely many values without ever settling down.

Much excitement has been generated by the discovery that this 'bifurcation tree', ending in chaos, arises in a wide range of mathematical contexts. An important application is to the way in which the motion of a flowing liquid changes in stages from smooth (stable attractor) to turbulent (chaotic attractor) as the speed increases (from the lazy river to the raging torrent). The same thing can be seen as smoke from a cigarette rises gently then suddenly spirals and swirls into turbulent eddies.

In this program the population evolution rule is specified in Line 6Ø. Use of the sound command gives a vivid impression of how the orbit reaches the attractor, but does slow the program down. To get a clearer picture of the attractor itself in a reasonable time first delete Line 8Ø. You can increase the resolution by changing Line 2Ø so that S = Ø.ØØ5, NMIN or NN = 2ØØ and NMAX or NX = 3ØØ.