Computers in Physics

Simple Programs Create 3D Images

J. C. Sprott

Citation: Computers in Physics **6**, 132 (1992); doi: 10.1063/1.4823057
View online: http://dx.doi.org/10.1063/1.4823057
View Table of Contents: http://scitation.aip.org/content/aip/journal/cip/6/2?ver=pdfcov
Published by the AIP Publishing

Simple Programs Create 3-D Images

J. C. Sprott

Static or animated two-color computer graphics provide a three-dimensional effect when viewed through red/cyan glasses

Modern computers produce output in such abundance that for many problems the bottleneck in understanding lies in the graphical limitations of the computer screen, which is inherently two-dimensional. Most interesting calculations involve more than two variables, and the loss of information that results when only two of these are exhibited is enormous.

A number of visualization techniques for capturing additional information are in widespread use. Color is commonly used to repesent an additional dimension. Animation is an especially appropriate method when one of the variables is time. Much progress has recently been made in improved rendering techniques to provide the illusion of depth through the use of ray tracing to produce realistic reflections and shadows. Objects can be displayed in perspective and rotated. In some cases, sound can be used to convey additional information. Many of these techniques can be used in combination. Most tax the memory and computation speed of desktop computers, and some are difficult to program.

An alternative approach is the binocular stereogram, in which the parallax produced by separate images in each eye creates the illusion of depth. The concept dates back to Socrates in the 4th century BC, and the earliest stereograms were produced in the mid-1800s by Charles Wheatstone (with the mirror stereoscope) and David Brewster[1] (with the lenticular stereoscope), using spatially non-overlapping images viewed through appropriate optics. The inception of motion pictures in the early 1900s was accompanied by 3-D movies using overlapping red-green images, which were viewed through red-green glasses to produce a black-and-white image in what is called the anaglyphic process.[2-4] Anaglyphs have also been widely used in comic books. Color 3-D movies using cross-polaroids were briefly popular in the 1950s. There has been a recent resurgence of interest in 3-D images using the alternating-field television process, in which left and right images flash alternately while viewed through synchronously-driven liquid-crystal shutter glasses. A number of other techniques, such as holography or the Pulfrich effect, offer promise for 3-D imaging in special applications.

Of the various techniques, the anaglyphic process offers distinct advantages in computer visualization.[5] The hardware requirements are minimal (a color monitor and a pair of 50¢ glasses[6]), the programming is surprisingly simple, and the results can be impressive. The technique can be combined with animation, which greatly enhances its effectiveness. The main drawback is that the images produced are usually monochromatic, although a gray scale and some limited coloration is possible. What follows is a discussion of binocular stereopsis, an appropriate algorithm for adaptation to computer graphics, and some examples.

Binocular Stereopsis

Our perception of depth arises from a number of psychological and physiological processes.[7] Many of these processes are induced by visual cues that do not depend on binocular vision, such as the relative size and motion of objects, interposition, illumination, shadows, and focal accommodation. Others require the parallax attendant on stereoscopic vision. When some of the usual visual cues are absent or contradictory, a conflict arises that demands time and effort for our brain to resolve. It is remarkable that with the single cue of binocular stereopsis, most people can quickly perceive a vivid three-dimensional image. This is fortunate, because it leads to a relatively straightforward computer implementation.

Consider a point at a distance D from the midpoint of our eyes, which themselves are separated by the distance e (typically 6.5 cm), as shown in Fig. 1a. The point might be a single illuminated pixel on a computer screen. Each eye must swivel inward through an angle Θ in order for

J. C. Sprott (sprott@juno.physics.wisc.edu) is a Professor of Physics at the University of Wisconsin - Madison and a specialist in plasma physics and nonlinear dynamics. His Chaos Demonstrations program was a winner in the first annual Computers in Physics *software contest.*

the two images to fuse into a single point, where Θ is given by:

$$\Theta = \tan^{-1}(e/2D) . \quad (1)$$

It is this muscular response of the eyes that provides the brain with the relevant depth information. Normal eyes can comfortably accommodate values of Θ in the range of zero (an object at infinity) to about 10° (slightly cross-eyed).

Now suppose the observer is to perceive the point to be at a distance $D - z$ (i.e., a distance z in front of the computer screen) as shown in Fig. 1b. We must then plot two points on the computer screen separated horizontally by Δx. From the similarity of the two triangles, we calculate

$$\Delta x = ez / (D - z) . \quad (2)$$

The formula also works for negative z. This choice for the sign of z yields a right-handed coordinate system when x and y are in the plane of the screen with $+y$ upward and $+x$ to the right.

To achieve the proper perspective, we also need to arrange for the nearby (positive-z) points to be separated more than the far (negative-z) points. Fig. 1c shows that a point with coordinates $(0, y, z)$ should actually be plotted on the computer screen at $(0, y')$, where it is assumed that the origin $(x = y = 0)$ is at the center of the screen. From the similarity of triangles, we see that

$$y' = Dy / (D - z) . \quad (3)$$

For the x-coordinate, the transformation is similar:

$$x' = (Dx \pm ez/2) / (D - z) \quad (4)$$

where the $\pm$ refers to the different colors.

Note that in the limit $z \ll D$ the transformation reduces to

$$y' = y \quad (5)$$

$$x' = x \pm ez/2D . \quad (6)$$

This approximation introduces some expansion of the image for negative z (behind the screen) and compression of the image for positive z (in front of the screen). The compression is sometimes desirable so as to keep the image always in front of the viewer's face rather than to let it pass behind the head. Note that in this limit, the only relevant scaling parameter is the distance to the screen divided by the distance between the eyes (D/e). For a computer screen viewed at arm's length, this parameter is, for most individuals, within a few percent of 10.0. In practice, the viewing distance is not very critical. The perceived depth of the image is increased by viewing from a greater distance, but it usually takes longer for the brain to accommodate, and so it is often best to view first from close up. It sometimes speeds the adjustment to move the head from side to side.

A computer display optimized for viewing at arm's length is actually very effective when projected on a large screen and viewed in an auditorium. Were this not the case, 3-D movies could not be shown to theater audiences. This result seems to contradict the unique dependence on the parameter D/e. In such a case, the brain perceives a scaled version of the image at a shorter distance. The same effect occurs when viewing 2-D movies. The actors are not perceived as giants at a large distance from the viewer. Similarly, the brain is able to compensate almost without limit for other distortions if the objects are familiar. A movie viewed from the right-most seat in the front row appears normal after a brief period of adjustment.

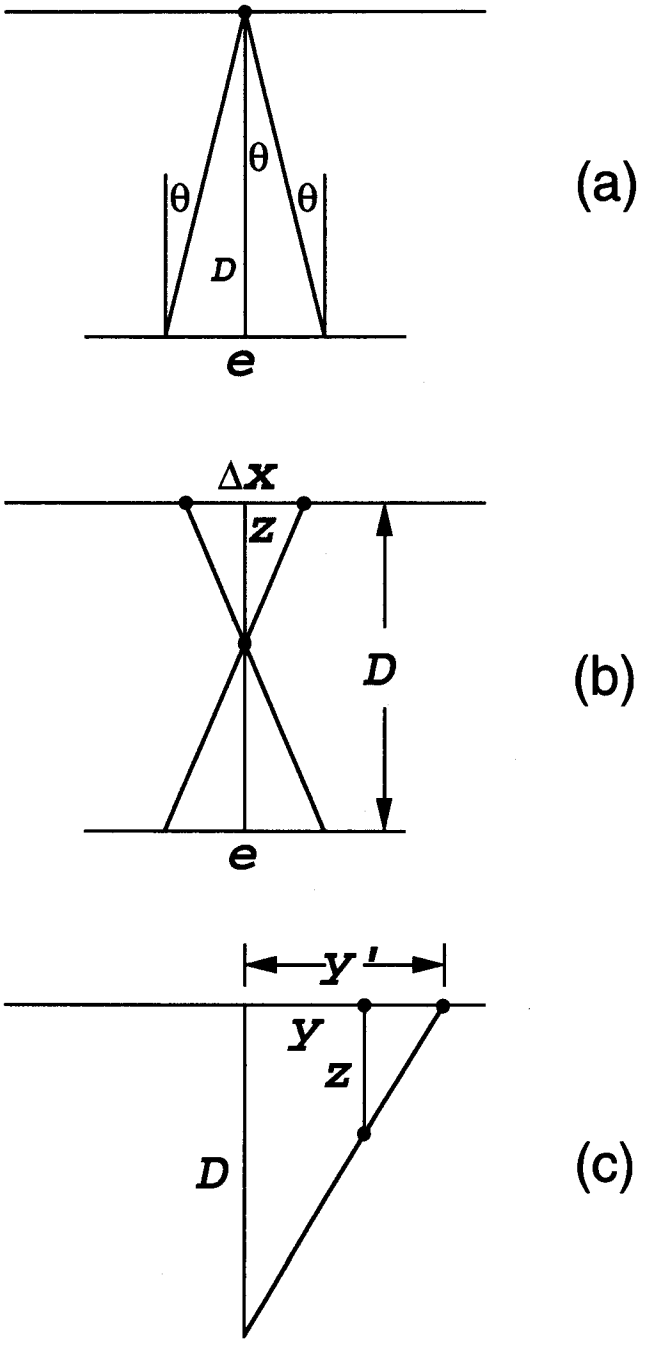

Fig. 1: (a) When viewing a point on a computer screen a distance D away, each eye toes-in by an angle Θ; (b) in order for a point to appear to be at a distance z in front of the computer screen, the eyes must see two points on the screen separated by a distance Δx; (c) objects that are to appear a distance z closer than the distance D to the computer screen are plotted not at their position y, but at position y'.

Box 1. BASIC code draws the anaglyph of two parallel lines shown in Fig. 2.

```
10 REM PARALLEL LINES
20 SCREEN 12                        'assume VGA graphics mode
30 WINDOW (-1, -1)-(1, 1)           'define screen coordinates
40 D = 5                            'distance from screen
50 E = .5                           'eye separation
60 WH = 15                          'white color
70 BK = 8                           'black color
80 RD = 12                          'red color
90 CY = 11                          'cyan color
100 PAINT (0, 0), WH                'paint the screen white
110 Y = -1: Z = 0                   'initial values
120 DT = .01                        'step size
130 WHILE INKEY$ = ""               'test keyboard
140    X = -1: GOSUB 200            'left line
150    X = 1: GOSUB 200             'right line
160    Y = Y + DT / 10              'increment position
170    Z = Z - DT
180 WEND                            'loop while no key is pressed
190 END

200 REM ROUTINE THAT PLOTS RED AND CYAN POINTS TO FUSE AT (X,Y,Z)
210 YP = D * Y / (D - Z)
220 XP = (D * X - E * Z / 2) / (D - Z)
230 P = POINT(XP, YP)
240 IF P = WH THEN PSET (XP, YP), RD ELSE IF P = CY THEN
                                        PSET (XP, YP), BK
250 XP = (D * X + E * Z / 2) / (D - Z)
260 P = POINT(XP, YP)
270 IF P = WH THEN PSET (XP, YP), CY ELSE IF P = RD THEN
                                        PSET (XP, YP), BK
280 RETURN
```

It is important to maintain a somewhat limited depth and field of view. Leonardo da Vinci recommended that a painting be viewed from an optimal distance equal to three times its width. Most computer screens satisfy this criterion, approximately at least, when viewed at arm's length. An object as deep as it is wide will thus require that the two images be separated by up to ± 1 cm, causing the eyes to toe-in by about 3 ± 0.5 degrees.

The computational task, therefore, is to plot each point that makes up the object twice, with a horizontal separation related to the distance at which that point is to appear in front of or behind the screen, and to arrange that one set of points be visible only to the left eye and the other only to the right eye. In the anaglyphic process, this is done by plotting one set of points in red and the other in the complementary cyan (blue-green), and viewing them through appropriate color-filtered glasses. By convention, the left eye should respond only to red and the right eye only to cyan. Note that individuals who are color blind should encounter no difficulty since it is unnecessary (and indeed undesirable) to perceive the individual colors; it is only necessary that the eyes be sensitive to them. Certain other eye defects, particularly those resulting in ocular asymmetry, are more problematic.

One has the choice of plotting the points on either a black or a white background. With a black background, the images fuse into white (additive process); with a white background, the images fuse into black (subtractive process). The sense of z is reversed with the choice of background. With a black background, the red is seen through the red filter on the left eye, while for a white background, red is seen through the cyan filter on the right eye. In practice, the white background is usually more satisfactory. Wherever a red and cyan point overlap, they should be plotted as a single black point if the background is white, or as a single white point if the background is black.

Because of the large variation of computer monitor colors and spectacle filters, ghost images are common. Manipulation of the computer color palette is of limited use, because the monitor ultimately constructs its colors from three distinct phosphors (red, green, and blue). The usual problem is inadequate rejection of the green by the red filter, resulting in a red ghost image when viewed against a white background. Suppression of the green by using only red and blue on a magenta background

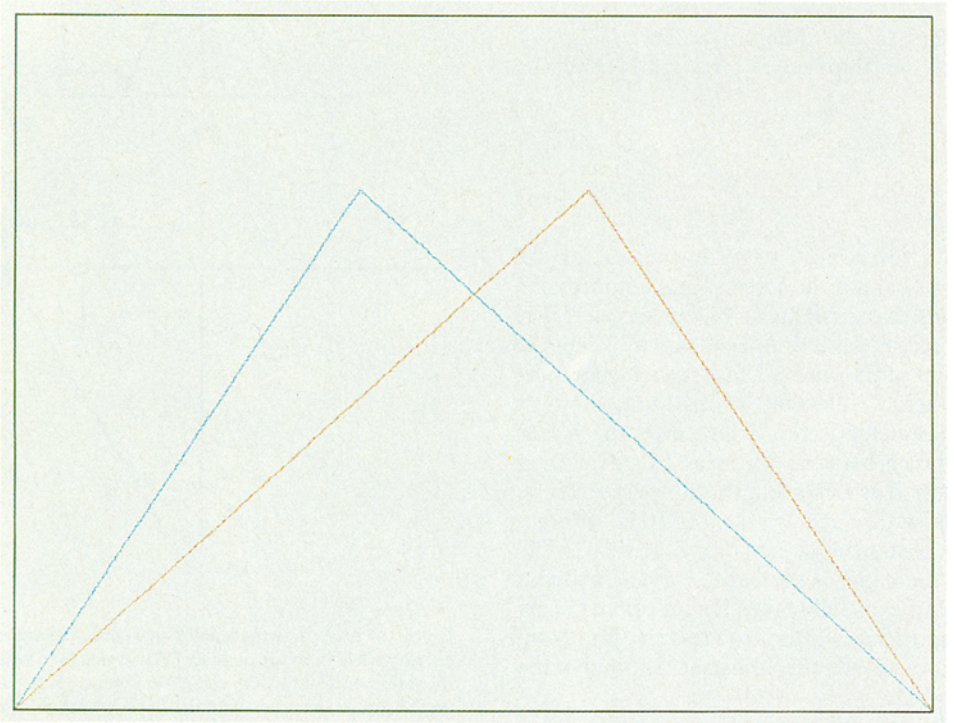

Fig. 2: Two parallel lines stretch to infinity in anaglyphic representation.

eliminates this problem, but yields poor contrast of the resulting image. The more expensive plastic frame glasses tend to have better filters than the cheaper cardboard frame types. In some cases, the ghost images can be suppressed by viewing through multiple pairs of glasses.

Some individuals experience inordinate difficulty acclimating to stereograms. A number of things can be done to ease the adjustment. One should first practice looking at familiar objects such as cubes and spheres. The objects should be relatively small and of limited depth, and they should initially be viewed close-up. Animated rotation helps immensely. It also helps to frame the picture with lines perceived to be in the plane of the screen, or to draw axes which bisect the picture vertically and/or horizontally. With some images which use only binocular cues, it is helpful to reverse the glasses (red over right eye), which reverses the sense of the z-axis.

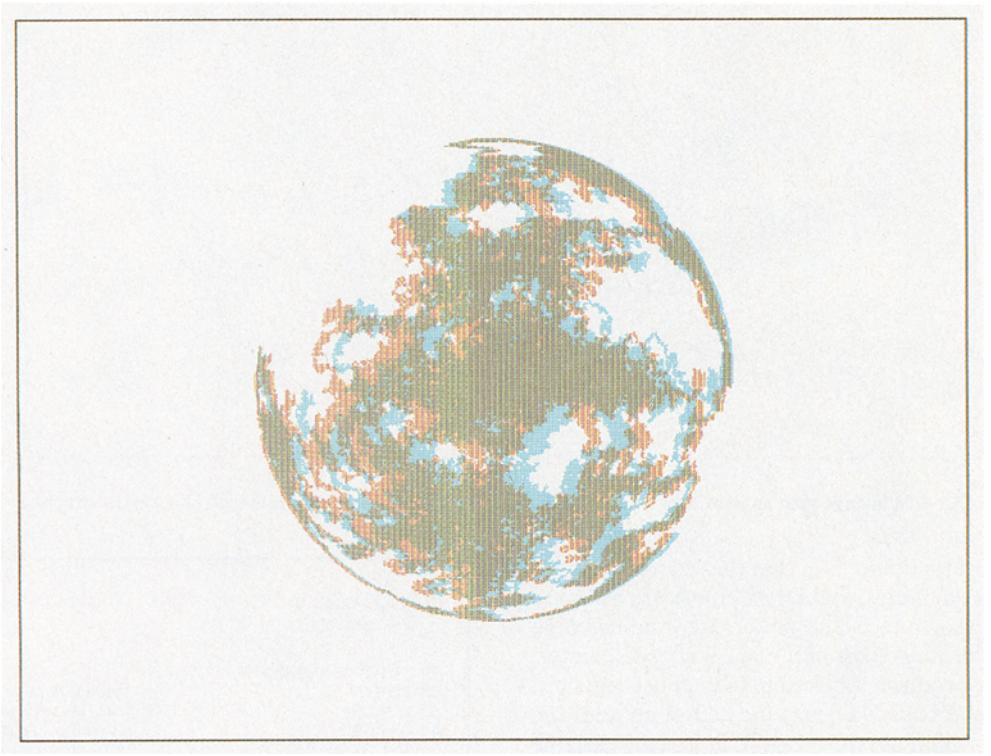

Fig. 3: An object executes a random walk on a sphere in anaglyphic representation.

Computer Algorithm

A simple computer code that implements the above ideas is given in Box 1. It is written in BASIC, and assumes an MS-DOS machine with VGA graphics. It should run without modification under TurboBASIC, PowerBASIC, QuickBASIC, or the QBASIC interpreter included with MS-DOS 5.0. BASIC was chosen to make the code widely accessible and to facilitate translation to other languages. The program draws two parallel lines that vanish in the distance like a pair of railroad tracks. The result of the calculation is shown in Fig. 2. This case is visually demanding because of its extreme depth. It is especially important in this case that the eyes be accurately horizontal.

Without the glasses, one should confirm that the red and cyan lines converge on points that are horizontally separated by a distance equal to the eye separation. This will of course depend on the size of the monitor in use. It may be necessary to alter E in line 50 to obtain the correct separation at infinity, and then to change D in line 40 so that the ratio D/E is equal to the viewing distance divided by the eye separation.

With EGA graphics, it is necessary to change SCREEN 12 in line 10 to SCREEN 9. Users with CGA graphics, or who have only the primitive BASIC interpreter provided with earlier versions of MS-DOS (BASICA or GW-BASIC), will need to use SCREEN 1 and change the colors to WH = 3, BK = 0, RD = 2, and CY = 1. The CGA color palette is rather limited, and the resolution is poor, but a vivid 3-D image should be observable.

Box 2. Change in the code of Box 1 plots a random walk on a sphere as shown in Fig. 3.

```
10 REM RANDOM WALK ON A SPHERE
20 SCREEN 12                          'assume VGA graphics mode
30 WINDOW (-2, -1.5)-(2, 1.5)         'define screen coordinates
40 D = 10                             'distance from screen
50 E = 1                              'eye separation
60 WH = 15                            'white color
70 BK = 8                             'black color
80 RD = 12                            'red color
90 CY = 11                            'cyan color
100 PAINT (0, 0), WH                  'paint the screen white
110 PH = 1: TH = 0                    'initial values
120 DT = .01                          'step size
130 WHILE INKEY$ = ""                 'test keyboard
140    PH = PH + (RND - .5) * DT
150    TH = TH + (RND - .5) * DT / SIN(PH)
160    X = SIN(PH) * COS(TH): Y = SIN(PH) * SIN(TH): Z = COS(PH)
170    GOSUB 200
180 WEND                              'loop while no key is pressed
190 END
```

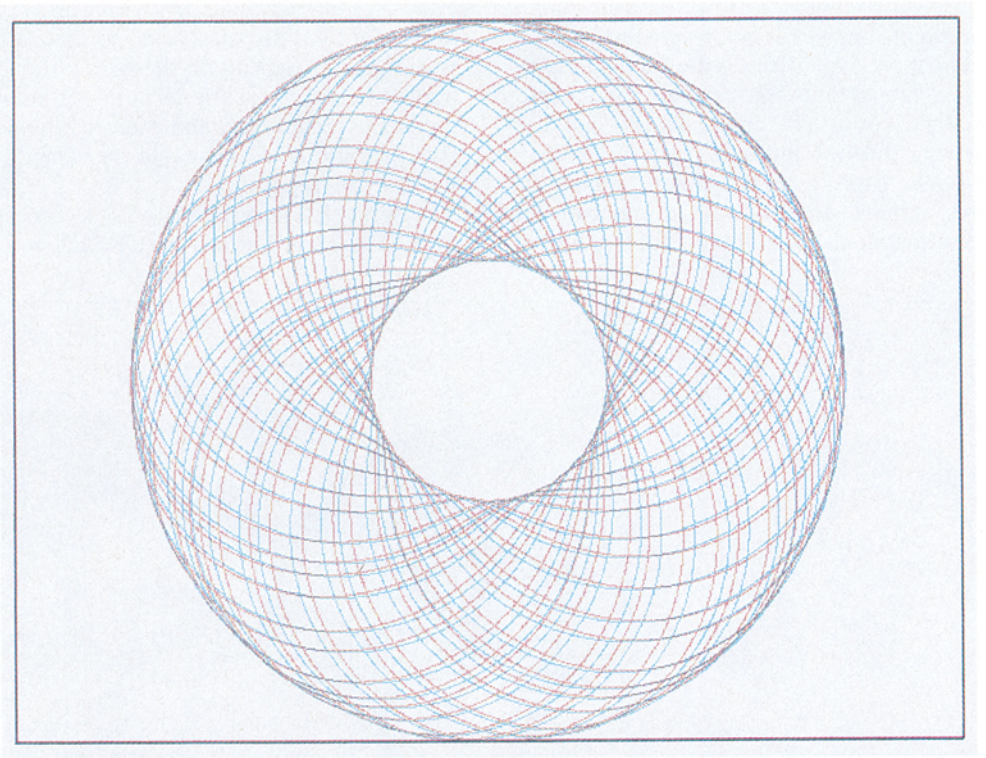

Fig. 4: Windings with an irrational winding number cover a toroidal surface in anaglyphic representation.

Macintosh users should note that the red and cyan included in the eight default QuickDraw colors are not pure (they don't add to white). The colors were adjusted to match the colors produced on the ImageWriter II printer, and thus they produce objectionable ghost images. However, a table of color values is located at an address, the pointer for which is stored in the global variable QDColors at $8B0. The red and cyan can be made pure with the BASIC statement

```
Q& = PEEKL(&H8B0): POKEL Q& + 26,&HFFFF: POKEL Q&
    + 28,0: POKEL Q& + 30,0:
POKEL Q& + 34,0: POKEL Q& + 36,&HFFFF:
    POKEL Q& + 38,&HFFFF
```

Examples

The techniques outlined above can be extended to a number of more interesting and visually attractive examples. Four such cases are described below.

Random Walk on a Sphere

To test the accuracy of the method as a visualization tool, it is best to use a familiar example so that any distortions introduced by the process are readily apparent. For this purpose a sphere is appropriate, and a random walk on its surface is chosen for programming simplicity. The position on a unity-radius sphere is specified in spherical coordinates by two angles Θ and ϕ for which the cartesian coordinates are

$$x = \sin\phi \cos\Theta \tag{7}$$

$$y = \sin\phi \sin\Theta \tag{8}$$

$$z = \cos\phi\,. \tag{9}$$

Box 3. Change in the code of Box 1 plots the torus windings shown in Fig. 4.

```
10 REM TORUS WINDINGS
20 SCREEN 12                        'assume VGA graphics mode
30 WINDOW (-2, -1.5)-(2, 1.5)       'define screen coordinates
40 D = 10                           'distance from screen
50 E = 1                            'eye separation
60 WH = 15                          'white color
70 BK = 8                           'black color
80 RD = 12                          'red color
90 CY = 11                          'cyan color
100 PAINT (0, 0), WH                'paint the screen white
110 A = .618034: T = 0              'initial values
120 DT = .005                       'step size
130 WHILE INKEY$ = ""               'test keyboard
140     X = (1 + .5 * COS(T)) * COS(A * T)
150     Y = (1 + .5 * COS(T)) * SIN(A * T)
160     Z = .5 * SIN(T): T = T + DT
170     GOSUB 200
180 WEND                            'loop while no key is pressed
190 END
```

The angles are iteratively advanced in steps given by

$$\Delta\phi = \Delta t(r - 0.5) \tag{10}$$

$$\Delta\Theta = \Delta t(r - 0.5) \,/\, \sin\phi \tag{11}$$

where r is a random deviate uniform over the range 0 to 1 and Δt is the iteration step size, which should be much less than 1 rad but not so small that the random walk is unacceptably slow. The $\sin\phi$ factor in Eq. (11) ensures that the east-west step size is on the average the same as the north-south step size. Code that implements this calculation is given in Box 2. A portion of the output is shown in Fig. 3. However, it is much more effective to see the trajectory evolve in real time than to view a still picture of the result. In either case, the object appears very nearly spherical, although a slight distortion results when the line

of sight is not perpendicular to the plane of the screen, in which case the sphere appears to be slightly ellipsoidal, elongated horizontally.

Torus Windings

A wire or magnetic field line that winds around the surface of a circular torus follows a trajectory given by

$$x = (1 + \epsilon \cos t) \cos(at) \tag{12}$$

$$y = (1 + \epsilon \cos t) \sin(at) \tag{13}$$

$$z = \epsilon \sin t \tag{14}$$

where ϵ is the inverse aspect ratio of the torus and a is the winding number (the number of times the winding goes the long way around the torus for each time around the short way). Rational values of a cause the winding to close on itself after a finite number of transits, and irrational values cause the winding to cover the surface of the torus densely. Code for calculating such a trajectory with $a = (\sqrt{5} - 1) / 2$ (the inverse of the golden mean) is given in Box 3, and a portion of the output is illustrated in Fig. 4.

Box 4. Change in the code of Box 1 plots the Lorenz attractor shown in Fig. 5.

```
10 REM LORENZ ATTRACTOR
20 SCREEN 12                        'assume VGA graphics mode
30 WINDOW (-40, -30)-(40, 30)       'define screen coordinates
40 D = 200                          'distance from screen
50 E = 20                           'eye separation
60 WH = 15                          'white color
70 BK = 8                           'black color
80 RD = 12                          'red color
90 CY = 11                          'cyan color
100 PAINT (0, 0), WH                'paint the screen white
110 X = -2: Y = .3: Z = 24          'initial values
120 DT = .0005                      'step size
130 WHILE INKEY$ = ""               'test keyboard
140    X = X - 10 * (X - Y) * DT
150    Y = Y + (28 * X - Y - X * Z) * DT
160    Z = Z + (X * Y - 8 * Z / 3) * DT
170    GOSUB 200
180 WEND                            'loop while no key is pressed
190 END
```

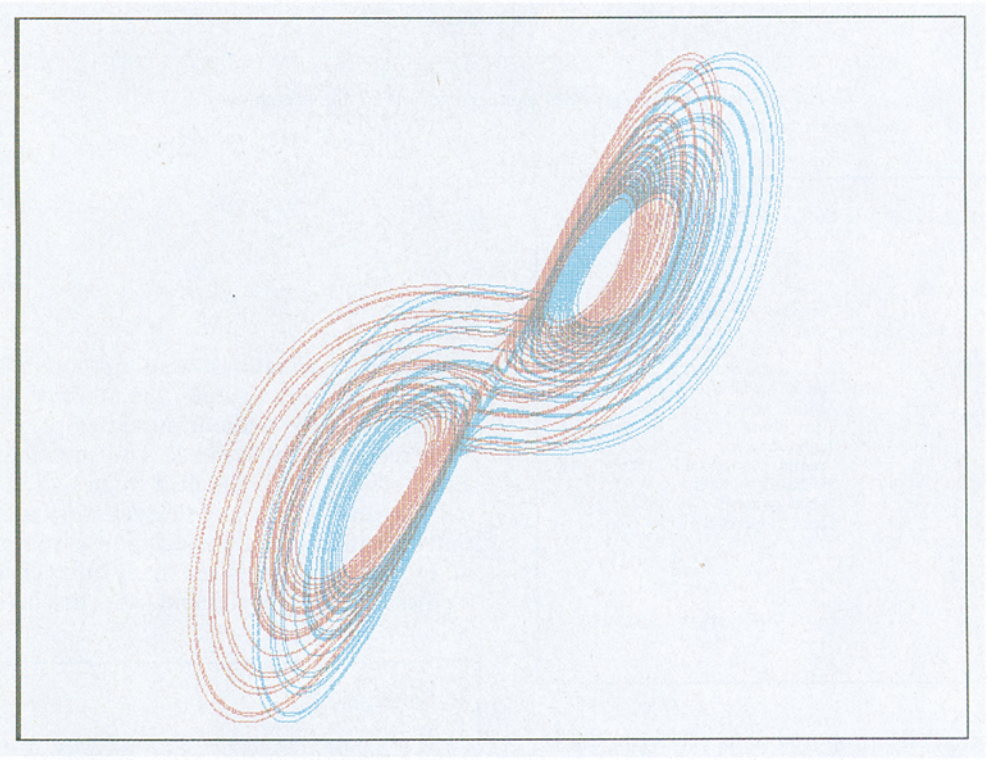

Fig. 5: The Lorenz attractor provides a good example of the use of anaglyphic representation.

Lorenz Attractor

A particularly useful application of the anaglyphic process is to view strange attractors and other chaotic trajectories. Strange attractors arising from differential equations always have dimension greater than two, and thus some enhanced visualization technique is required. One common method is to take a Poincaré section in which some appropriate cross section of the attractor is plotted, reducing the dimension by one, but with significant loss of information. In a periodically driven system, successive Poincaré sections can be taken at different phases of the drive and played back to produce an animation. However, considerable mental gymnastics are required to visualize the attractor, and the method fails when there is no periodic drive function.

A prototypical example of a strange attractor results from a solution of the Lorenz equations[8]

$$x' = \sigma (y - x) \tag{15}$$

$$y' = rx - y - xz \tag{16}$$

$$z' = xy - bz \tag{17}$$

where the canonical values of $\sigma = 10$, $b = 8/3$, and $r = 28$ are taken to ensure chaotic behavior. The Lorenz

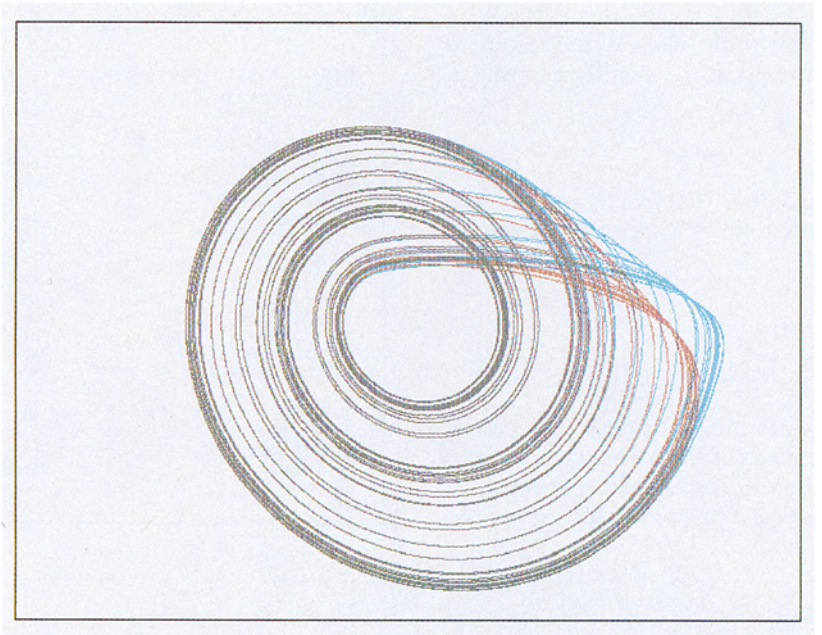

Fig. 6: The Rössler attractor provides another example of the usefulness of anaglyphic representation.

Box 5. Change in the code of Box 1 plots the Rössler attractor shown in Fig. 6.

```
10 REM ROSSLER ATTRACTOR
20 SCREEN 12                        'assume VGA graphics mode
30 WINDOW (-16, -12)-(16, 12)       'define screen coordinates
40 D = 80                           'distance from screen
50 E = 8                            'eye separation
60 WH = 15                          'white color
70 BK = 8                           'black color
80 RD = 12                          'red color
90 CY = 11                          'cyan color
100 PAINT (0, 0), WH                'paint the screen white
110 X = -9: Y = -.6: Z = 0          'initial values
120 DT = .002                       'step size
130 WHILE INKEY$ = ""               'test keyboard
140    X = X - (Y + Z) * DT
150    Y = Y + (X + .2 * Y) * DT
160    Z = Z + (.2 + Z * (X - 5.7)) * DT
170    GOSUB 200
180 WEND                            'loop while no key is pressed
190 END
```

equations were originally proposed as a crude model of atmospheric convection. The Lorenz attractor is a fractal with a Hausdorff dimension[9] of 2.06 ± 0.01 embedded in a three-dimensional phase-space. The code for solving the above equations using a leap-frog scheme is shown in Box 4. The leap-frog method is not particularly accurate for this problem, but it is simple to program, and it preserves the topological character of the solution provided the step size DT is sufficiently small. A plot of a portion of the solution is shown in Fig. 5, but it is more effectively displayed in real time by choosing a value for DT that allows the trajectory to evolve slowly for the particular computer used.

Rössler Attractor

A final example of a strange attractor is the Rössler attractor,[10] which results from the solution of the following equations:

$$x' = -(y+z) \tag{18}$$

$$y' = x + ay \tag{19}$$

$$z' = b + z(x-c)\,. \tag{20}$$

The Rössler equations were proposed for pedagogical purposes in order to study the minimal requirements for chaotic behavior without any attempt to associate them with real physical systems. The equations are solved in Box 5 using the canonical values of $a = b = 0.2$ and $c = 5.7$, and a portion of the trajectory is shown in Fig. 6. Software that includes anaglyphic displays of the Lorenz and Rössler attractors and many other chaotic systems, as well as Poincaré movies and other displays, is available.[11]

References

1. D. Brewster, *The Stereoscope, Its History, Theory and Construction*, 1856, reprinted by Morgan and Morgan, New York, 1971.
2. R. M. Hayes, *3-D Movies, A History and Filmography of Stereoscopic Cinema*, MacFarland & Company, North Carolina, 1989.
3. L. Lipton, *Foundations of the Stereoscopic Cinema, A Study in Depth*, Van Nostrand Reinhold Company, 1982.
4. H. Morgan and D. Symmes, *Amazing 3-D*, Little, Brown and Company, Boston, 1982.
5. D. M. McKinstry, *Computers in Physics* **2**, 44 (1988).
6. Available from Reel 3-D Enterprises, Inc., P.O. Box 2368, Culver City, CA 90231; or The 3-D Zone, P.O. Box 741159, Los Angeles, CA 90004, or most comic-book stores.
7. B. Julesz, *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago, 1971.
8. E. N. Lorenz, *J. Atmos. Sci.* **20**, 130 (1963).
9. D. A. Russel, J. D. Hanson, and E. Ott, *Phys. Rev. Lett.* **45**, 1175 (1980).
10. O. E. Rössler, *Phys. Lett. A* **57**, 397 (1976).
11. J. C. Sprott and G. Rowlands, Chaos Demonstrations, Physics Academic Software, available from The Academic Software Library, Box 8202, North Carolina State University, Raleigh, NC 27695-8202.