Solid Shape Drawing on the Commodore 64

by Richard Rylander

Computer graphics is one of the fastest growing areas of computer software and hardware development. The old saying that "one picture is worth a thousand words" couldn't be truer when you consider the effectiveness of a pie chart or bar graph versus columns of numbers. Even the simplest of home computers is capable of at least this basic form of computer graphics. But the term *computer graphics* usually brings to mind more exciting images than just mundane business applications.

Special visual effects for motion pictures and some television advertisements are now being done with computers to create scenes of "enhanced reality." We see spacecraft making impossible manuevers or shimmering corporate logos "flying by" in ways that would be difficult to simulate with conventional photographic techniques. The enormous amount of data and processing necessary to produce such images has made the use of "super computers" essential. Even with the largest computers working full time on the task, only a few feet of film are produced per day. For home computer owners who want to create their own computer-generated masterpiece, however, the task is not hopeless.

The Apple Macintosh represents a major step in bringing powerful, easy-to-use graphics capabilities to the masses. The "masses," however, may still find the Macintosh pricey, especially if they just want to play around a bit with MacPaint. Low-cost computers, such as the Commodore 64, while lacking the slick graphic interaction of the Mac, are selling for such ridiculously low prices ($150 at the time of writing this article) that they are hard to pass up. They have the potential for some fairly sophisticated graphics, but so far most of the commercially available software has done little more than add the usual "point plotting" and "line drawing" commands.

This article describes a software package that will allow you to create your own computer-generated scenes, such as "Coffee and Donuts" in Figure 1 (page 55) and "Goblets" in Figure 2 (page 55). You can construct images from combinations of elementary shapes (spheres, cylinders, and toroids in various orientations) or by the "polygon mesh" technique typically used to render more complex objects. You can produce drawings with realistic shading effects and in a variety of styles to make some surprisingly detailed pictures with a minimum of effort. The package includes demonstration programs to illustrate how you use each graphic function and style option.

In keeping with the "running light without overbyte" theme of *DDJ*, the entire graphics package fits in 3K of RAM. The program sits in an area of

Pumping pixels requires tight, efficient code. Here are halftone shading, backlighting and other visual effects in a tight little package.

Richard Rylander, 179 N. McKnight Rd. Apt. 203, St. Paul, MN 55119.

RAM that is inaccessible to BASIC (a 4K block following the BASIC ROM) so that you don't lose any useful program space. Commodore supplies a "DOS Wedge" program that occupies the last 1K of this block, and one *of my* objectives in the design of the graphics package was to maintain compatibility with this useful utility.

The compactness of the code required me to leave out some "bells and whistles," but I've made no sacrifices to execution speed or user-interface ease. The main omission is error trapping—the software performs no checking to ensure that you use only "legal" point coordinates and so on. This may make program development a little more difficult, but once you have written and debugged a program properly, error checking becomes extraneous and only slows program execution.

We must address several general tasks in developing a shape-drawing program. The first is, of course, how to calculate the apparent brightness for all points on visible surfaces of objects. Next, how do we display these different brightness levels on a high-resolution display that is basically on or off? Finally, a "general" task, but detailed in nature, is creating the specific software tools that will let us do the required calculations and bit-map manipulations.

The program itself is written in machine language, because even with our simplifying restrictions, the plotting of each pixel still involves considerable computation. The program uses integer arithmetic throughout, and we utilize all the symmetry available to eliminate redundant calculations. The main program actually consists of five separate subprograms to break the process into manageable pieces and to provide a library of separate utilities that you may find useful in other programs. Before getting into the problem of shade calculation, I'll present the first subprogram, *which is* a collection of integer arithmetic utilities that all the later subprograms need.

Integer Arithmetic Utilities

Listing One (page 61) provides the source code for a set of four integer arithmetic routines. Because we will draw our shaded shapes on a high-resolution display of 320 x 200 points, we need only single-precision arguments for most functions and double-precision results for some intermediate values.

The first two routines, multiplying two single-precision numbers to yield a double-precision product and dividing a double-precision dividend by a double-precision divisor to yield a double-precision quotient, are fairly standard routines that should need no description other than the comments in the code. The multiplication routine also contains a special case to treat a single-precision number as a signed integer, which is then squared.

The integer-square-root routine deserves some special comment. Have you ever tried the BASIC square-root function? If you have, you know it takes about 52 milliseconds. At this rate, plotting the 64,000 pixels of a 320 x 200 screen would take about 3,328 seconds, or nearly an hour, if we need a square root per point! Of course, we don't need a square root for each point, but it is an

essential part of many shade calculations. Obviously, anything we can do to speed up this particular function will have a great effect on the overall program speed.

The BASIC square-root routine is slow because BASIC is written to save space, not time. Virtually all small BASICs find a square root by first taking the log of the argument, dividing that result by 2, and then exponentiating. Because LOG and EXP functions are already available, why use any more space than necessary to derive SQR?

One way to speed up this function is to use Newton's method, an iterative procedure that can give full floating-point precision in less time. Because we are interested only in integer results, we can do much better yet, though.

The method we actually use is essentially as fast as doing a single-divide operation. We construct the root, bit by bit, in a manner similar to the way we would take a decimal square root by hand using pencil and paper (does anyone remember how to do that in this age of electronic calculators?). The process is even easier for a binary root because the guesses are, of course, only 1 or 0. The procedure is almost identical to that for division: we guess a partial root bit (as we would guess the next quotient bit) and, in effect, compare the square of the partial quotient to the argument to see if we keep the 1 guessed or change it to a 0 and restore the argument. For more details about this procedure, I recommend section 17.3, "Binary Square Roots—Restoring Method," of the book *The Logic of Computer Arithmetic* by Ivan Flores (Englewood Cliffs, NJ: Prentice-Hall).

The last integer arithmetic routine is a fast procedure for generating pseudo-random bytes. An old and popular method for generating pseudo-random numbers is the so called congruential method, in which you derive each successive random number from the previous random number by multiplying by a suitably chosen constant and taking the product modulo P, where P is a large prime (see, for example, "A Better Random Number Generator," by H. Cem Kaner and John R. Vokey in the June 1984 issue of *MICRO*). Again, though, because we are interested only in generating random bytes, we can use a much faster routine.

The method we use is to store two previous random bytes, exclusive OR them bit by bit, and rotate the result to the right by one bit to generate a new random byte. Because it needs no multiply or divide operations, provides a fairly uniform distribution of random bytes, and has a long period, this process is fast. Note that even though we are only generating random bytes (0-255), the pattern or sequence of successive bytes doesn't repeat until after 35,805 calls to the random-number-generator routine (for the initial parameters used in the program). The long period is necessary to ensure that we won't produce any unwanted secondary patterns in what should be "randomly" shaded images—the human eye is adept at picking out such correlations.

Now that some low-level number crunching is out of the way, we can concentrate on the graphics aspects of the main program.

Graphics Utilities

Listing Two (page 62) provides the source code for the elementary graphics functions of displaying and clearing the bit-map, filling the color map (determining dot/background colors), and plotting (and unplotting) individual points both directly and with the plot or unplot decision weighted by a shade value and a particular shade style function. The PLOT (and UNPLOT), CLEAR, COLOR, GRFON (display graphics screen), and GROFF (return to text screen) routines are specific to the Commodore 64. To adapt this program for other 6502-based computers, you need to rewrite these routines, but the rest of the graphics package (save the final user interface to BASIC) is machine independent.

The SHADE routine is simple and straightforward. We set up an 8 x 8 threshold matrix as a linear array of values 0 to 63. The six bits we need to address an element of this threshold array are determined by masking off the lower three bits of the absolute X and Y screen coordinates (we take X modulo 8 and Y modulo 8) with the Y bits shifted to become the upper three bits. This effectively repeats the threshold pattern over the entire bit-map area. For each point within an object to be drawn, we calculate a shade value normalized to the range 0-63 and then compare it to the threshold value at that screen point to decide whether to plot or unplot.

If we have, for example, a large area where the shade value is constant at 31 (a 50 percent gray), then within each 8 × 8-pixel character cell block, half of the threshold values will be greater than or equal to the shade value, turning on half the pixels. The order in which the values 0 to 63 are arranged in the threshold table determines how these on pixels are distributed. A common pattern for these so called ordered-dither matrices is a recursive arrangement such as that shown in Table 1 (page 53). If that matrix is subdivided into quarters, sixteenths, and so on, each submatrix has the same general pattern, with offsets between submatrices.

The matrix in Table 1 (classic Bayer ordered dither) keeps the on and off pixels spread as far apart as possible for any shade. This keeps the spatial frequencies in the shade "texture" as high as possible for a particular shade. A disadvantage, though, is that texture changes then accompany shade changes, making shade quantization more apparent. Another disadvantage is that many color monitors have a hard time coping with the very high frequency on-off-on sequence of pixels, distorting shade values when isolated pixels get lost.

As an aside, a convenient algorithm to generate the ordered-dither matrix in Table 1 (or such a matrix of any size) is found in Figure 7 (page 59).

The threshold matrix we actually use in the program is shown in Table 2 (page 53). Here we follow a recursive scheme as we start turning on pixels until eight nuclei have been established. As shades increase, new pixels are added around the edges of these nuclei, simulating the dot-growth behavior seen in normal printing halftones. Except in the ex-

treme highlight and shadow regions, the shade texture remains fairly constant. Also, the clustering of on or off pixels is much less demanding on the display bandwidth. But take your pick—try filling the threshold matrix with your own patterns.

The RSHADE routine shades by comparing the shade value to a pseudo-random byte shifted right twice to match the 0-63 range. This scheme also tends to average out the tone errors generated as each pixel is turned only on or off (though we want an intermediate shade of gray) by dithering the threshold value randomly at each pixel. Both shading schemes produce sharp edges because each pixel is plotted independently. Abrupt shade changes are then followed faithfully.

The SCALE routine provides an opportunity to mention a general rule you should follow when you try to write fast programs: If at all possible, avoid division; and if you must divide, try to make it by a power of 2 (so that you can use right shifts). The SCALE routine helps correct some of the geometric distortion that otherwise results from plotting objects on the Commodore 64's rectangular bit map. While the monitor display has an aspect ratio of approximately 4:3, the bit-map aspect ratio is 320:200 or 8:5. A simple way to keep sphere outlines circular (instead of the usual egglike appearance) is to work in pseudoscreen coordinates of 320 × 240, giving the bitmap the same 4:3 aspect ratio as the screen display. The SCALE routine then converts 0-239 YPLOT values to a 0-199 range.

An obvious way to do this is to first multiply by 5 and then divide by 6. Or you might save a multiplication by first dividing a copy of YPLOT by 6 and subtracting that from the original YPLOT. A *much* faster way is to multiply the single-precision YPLOT by 213 and then take just the upper byte of the double-precision product (effectively dividing by 256) as the scaled YPLOT value. This gives us the proper range of absolute screen Y values and "rounds" the scaling as well.

This method of scaling means that you effectively replot every sixth Y line of pixels, but the routine's simplicity more than makes up for plotting 20 percent more points. Scaling is left as an option because some printers (such as the Commodore 1525) have 1:1 dot densities. By not scaling, the screen display is distorted but a hard-copy printout will have the proper geometry. On the other hand, you can set up printers, such as the Epson RX-80, with the appropriate horizontal and vertical dot densities to produce a 4:3 aspect ratio screen dump. Scaling here corrects both the screen and the hard-copy output.

The routine PLTSHD is a higher level routine that simply checks a couple of flags to see what kind of shading we want and whether to scale or not. It then calls the appropriate shading routine. To plot a shade-weighted pixel from BASIC, POKE a shade value (0-63) into VALUE; POKE the absolute X and (optionally scaled) Y values into XPLOT, XPLOT+1, and YPLOT; set up the flags HTORRN (HALFTONE or RANDOM) and NOSCAL; then SYS to PLTSHD. This seems like a lot of work to plot a single point, but we really won't be plotting shade-weighted points by hand—the later shape-drawing routines take care of this.

Line and Facet Drawing

Listing Three (page 64) completes the elementary graphics functions by adding line-drawing and shaded-facet-drawing routines. We draw lines using a modified form of Bresenham's algorithm, a DDA (Digital Differential Analyzer) technique that keeps the actual plotted points within one half-screen unit of the true line. Regardless of the order in which we specify the line's endpoints, the program sorts them so that lines are always drawn from left to right (the X position is incremented or unchanged at each step). We then need only determine which is greater, the change in X or the change in Y, and whether the Y coordinate difference is positive or negative. The program checks

0	32	8	40	2	34	10	42
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

Table 1
Recursive Arrangement of Ordered-Dither Matrix

0	8	53	61	2	10	55	63
16	24	37	45	18	26	39	47
49	57	4	12	51	59	6	14
33	41	20	28	35	43	22	30
3	11	54	62	1	9	52	60
19	27	38	46	17	25	36	44
50	58	7	15	48	56	5	13
34	42	23	31	32	40	21	29

Table 2
Threshold Matrix

the scale flag to see if the endpoints should be adjusted (in their Y coordinates) before plotting commences. This keeps a common coordinate system for drawing shaded shapes (spheres and so on) and lines in the same image. A flag MODE determines whether the line is drawn by setting or clearing pixels ("black" or "white" lines).

The program's last elementary function is to draw shaded triangular facets. A powerful and flexible technique for rendering objects is by means of a polygon mesh. Natural polyhedral objects (for example, cubes and pyramids) are obvious candidates for this method, but you can also approximate curved surfaces by a connected mesh of planar polygonal sections. As in piecewise-linear approximations of curves, the finer the segmentation, the better the approximation—although at the price of greater computational overhead. The coordinates of the polygon vertices then constitute a data base that you can manipulate easily for rotational transformations, perspective transformations, and so on. Triangular sections are the simplest to handle and the most general because you can break any higher-order polygon down into triangular sections.

As in the line-drawing routine, the program sorts the endpoints of the facets into a left-to-right order and checks the scale flag to maintain a consistent coordinate system. At each X position across the facet, a top and bottom Y coordinate pair is determined by a simple proportionality between the X difference and Y difference for the particular triangular sides. Although this involves both multiplication and division operations, it is not the innermost loop here (drawing the shaded line segment between Y pairs is). It also does not give more than one Y value per X position, as would a DDA method (for lines with slope greater than 1 in magnitude).

The maximum X difference is restricted to less than 256 (single precision), though the absolute X coordinates for the triangle vertices can be anywhere on the screen. This is not too severe a constraint because most polygon meshes are made up of small sections, and, if a larger triangle is essential, you can break it into smaller triangles that meet this condition. The occasional inconvenience this might cause is worth the increase in speed and shortened program.

We leave the shade value used in drawing facets as a parameter that the calling program specifies. Although adding "surface normal" calculation and shade value computation based on some illumination model (and requiring the Z coordinates then to be specified for the vertices) is not difficult, the same shade value is used for all points of the facet, making shade determination by a BASIC program practical. Also, leaving shade calculations up to BASIC adds the flexibility that we can use any shading model. The curved surface-drawing routines I describe later do include shade determination, but here each point can potentially have a different shade value, putting it in the innermost loop. For this reason, the curved surfaces must have a fixed shading model.

An interesting use for the BASIC-specified shade values is drawing "white" facets by setting the shade to 64. This doesn't seem particularly useful, but when you draw "wireframe" polyhedra, it makes a simple, hidden-line removal scheme possible. If the facets of an object are sorted so that they are drawn from those furthest to those nearest the observer, with "black" lines added around the edges of each facet, then foreground facets that partially obscure background facets erase the lines in their interior. We can set a flag EDGES so that, after the program has drawn a shaded facet, it can add lines automatically to outline and emphasize the edges. The MODE flag again determines how the lines are drawn (set or cleared).

Now that we have a toolbox of arithmetic and primitive-graphics routines, we can concentrate on the main subprogram for drawing shaded, curved surfaces rapidly.

Determining Shade Values for Curved Surfaces

Many recent advances in computer image synthesis have dealt specifically with how different surfaces reflect, refract, scatter, or absorb light. Increasingly complex models for the visual appearance of various surface textures have led to ever more realistic images, but increasing computational overhead accompanies these models. We'll employ the simplest shading scheme to start with and then suggest how you might modify it to add "realism," while still keeping the computations manageable.

If we restrict ourselves to simple, symmetrical objects, we can greatly simplify the problem of calculating surface brightness. This is not a severe restriction in itself, as you can break most "complex" objects down into combinations of simple elemental shapes. We will have to limit ourselves to "normal" (that is, head-on) views of the objects, because once we allow objects to be rotated, they lose most of their symmetry. Also, calculation time (particularly that for hidden surface removal) increases enormously.

We use the Lambertian (diffuse reflector) model to determine the surface brightness. We assume that light comes from a single, specific direction and then find the cosine of the angle between this reference vector and a surface normal vector. This gives us shade values from 1, where the surface faces the light source, to 0 at the terminator, where light just grazes the surface; and negative values where the surface is turned away from the light. Here is our first option: We can clip the brightness values at 0 so that areas facing away from the light have zero brightness, as an object lit by a single source (in deep space) would appear. Or, we can use the absolute value of the cosine so that the object appears to be lit by two identical sources on opposite sides. Either way is simple, so we allow for both cases by using a flag to determine what kind of illumination scheme we want to use for a whole scene or any individual shape making up the scene.

Although this simple model is valid for diffusely reflecting surfaces, it does not account for any contributions from specular surface reflections, secondary light sources (other

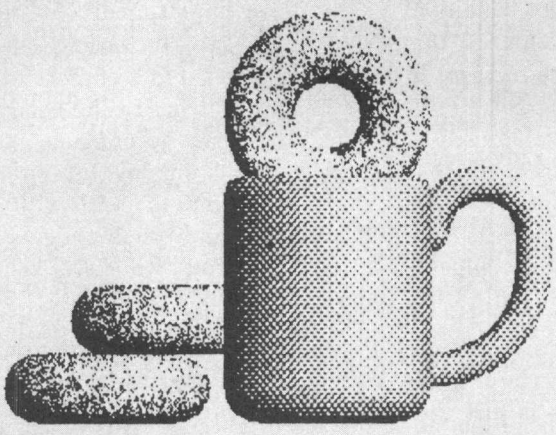

Figure 1

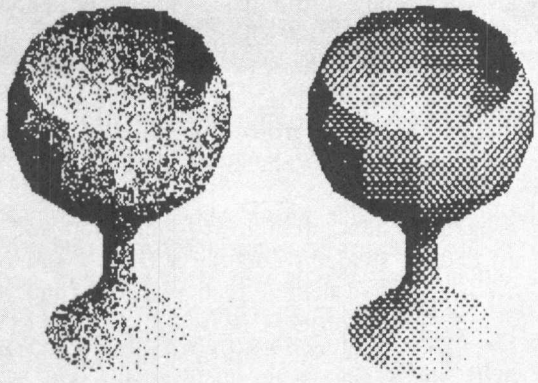

Figure 2

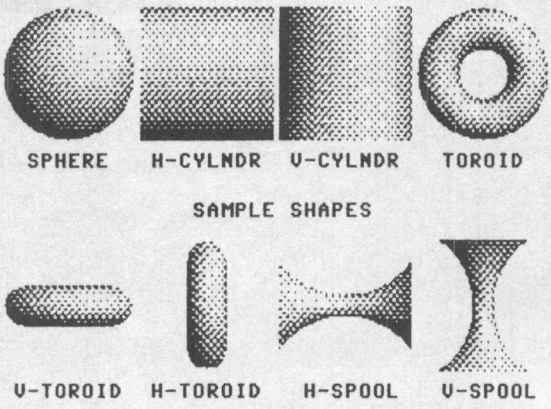

Figure 3

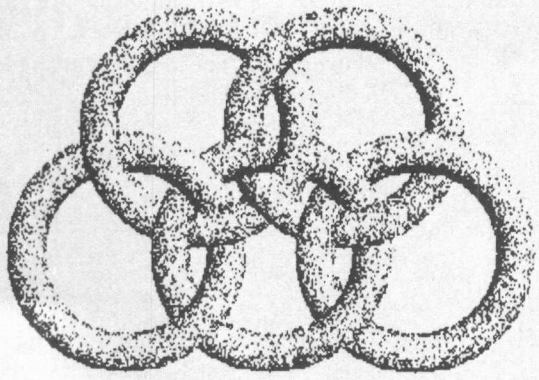

Figure 4

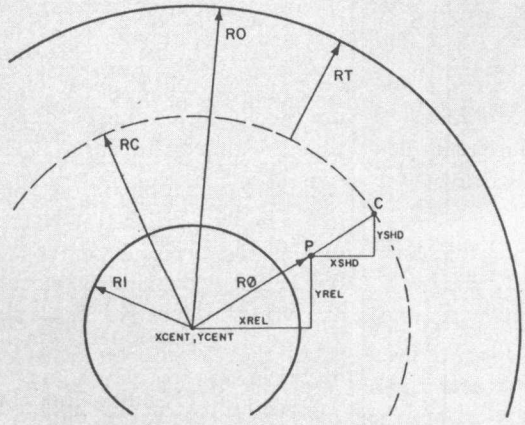

Figure 5

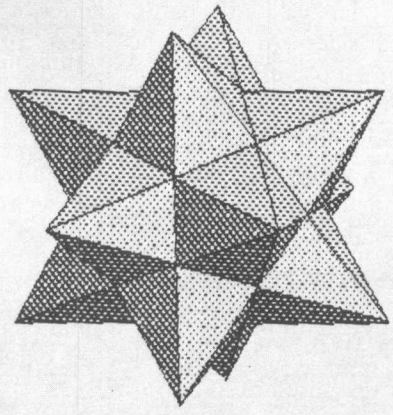

Figure 6

than the "opposite face" light) or ambient (nondirectional) light. These additional subtleties are not difficult to implement, but for the level of detail and dynamic range (number of gray levels) in our halftone display, they really aren't worth the effort.

Object Geometry

Calculating the surface normal vectors for points on an arbitrary surface can be difficult, but this is why we have restricted ourselves to objects that can be made up of combinations of the eight simple shapes shown in Figure 3 (page 55). All the basic shapes have at least fourfold symmetry; that is, once we have found the surface normal for a point (X, Y) in one quadrant of the object, we can easily find the surface normals for the other three quadrants by complementing appropriate components. Also, by restricting ourselves to shapes with relatively simple surfaces, we can determine the required surface normal vectors without resorting to trigonometric functions, using instead some simple geometric considerations.

Another time-saver is to choose an illumination reference vector to best take advantage of symmetries in the objects. Unfortunately, the "best" choice for computational purposes—a light source directly behind the viewer—gives a rather flat-looking image. Much more pleasing shading results from illumination coming from "over the shoulder." The slight assymmetry that results gives a stronger sense of depth to the objects.

The coordinate system describing objects on the screen has its X axis increasing horizontally to the right, the Y axis increasing vertically upward, and the Z axis increasing out of the screen toward the viewer (a conventional right-handed coordinate system). The illumination reference vector used here has X, Y, and Z components of (1,1,2), representing light coming from over the right shoulder. The symmetry in X and Y makes for easier shade calculations and gives a similar looking scene if you rotate a hard-copy output 90 degrees to fit objects that are taller than they are wide into the display. Lighting in the "portrait" (as opposed to "landscape"—unrotated) mode then comes from over the left shoulder.

Drawing a Shaded Sphere

Spheres are particularly easy objects to deal with because a radial vector from the center to a point on the surface is in the same direction as a surface normal from that point. The only real problem we must deal with is one of normalization. We will *always* deal with coordinates relative to the "local center of curvature" of objects. For spheres, this is just their geometric center.

Consider a point on the surface that extends out of the screen from relative coordinates (X,Y). We find the corresponding Z value using the equation for a sphere:

```
Z = SQR(R*R - X*X - Y*Y)
```

This calculated Z then forms the third member of the surface normal vector. To calculate the cosine of the angle between this vector and the illumination reference vector, we must first normalize both to have unit length. The normal we have just found has, of course, a length of R, so we need only divide each component by the radius of the sphere. The illumination vector (1,1,2) has a length of SQR(6), so we normalize this vector to unit length by dividing each component by SQR(6).

Now that we have two vectors of unit length but different directions, we find the cosine of the angle between them by taking their inner product, which is nothing more than summing the products of the X, Y, and Z coordinate pairs—like so:

```
COSINE = (1*X + 1*Y + 2*Z)
  / (R*SQR(6))
```

To use integer arithmetic throughout, we don't really want brightness expressed as a fraction between 0 and 1, but rather scaled to a range of integers from 0 to 63. We choose a maximum shade value of 63 to match the 64 gray levels that can be approximated by an 8×8 threshold matrix (described earlier in the "Graphics Utilities" section). We scale the pseudo-random bytes (used in the RSHADE routine) to the same range simply by shifting them right twice. So, for a surface point (X, Y, Z) on a sphere of radius R, the appropriate SHADE integer (0-63) is:

```
SHADE = 26*(X + Y + 2*Z)/R
```

The factor 26 properly accounts for the SQR (6) in the denominator of the previous equation.

As stated earlier, all the primitive shapes considered here have at least fourfold symmetry. We need compute the Z component only once for each ± X and ± Y quartet of points (relative to the geometric center of the object). The sphere and top-view toroid actually have n-fold symmetry, which the Cartesian grid of screen pixels reduces to eightfold symmetry. That is, in addition to changing signs of X and Y, we can exchange the X and Y coordinates to find another surface point with the same Z value.

Shape-Drawing Routines

Listing Four (page 68) is a set of routines for drawing the eight simple shapes shown in Figure 5. The routines resemble some form of compiled BASIC in that they use no special bit manipulations or unusual addressing methods, but just have appropriately ordered calls to the lower-level routines set up earlier. Comment statements precede each shape-drawing routine, giving an equivalent BASIC routine. This seems to be the easiest way to explain each routine because most readers are familiar with BASIC program methods. I thus give special comment to only a few of the routines here.

GETVAL is a shade-normalization routine that also checks for normal or backlit illumination via the flag BAKLIT. The byte pair at TONE contains the inner (dot) product of the illumination vector with the local surface normal vector. GETVAL then effectively does the division by SQR(6)*R and multiplication by 63 to put VALUE into the proper range, clipping at zero or taking the absolute value, as necessary.

PTPLOT does all the dirty work needed to plot shaded points for four-

fold symmetrical objects. Provision is also made here for clipping the object at independent levels up, down, left, and right of the object's center. The clipping feature is needed to blend various primitive objects into more complex ones with smooth transitions at the seams. Clipping also allows us to create some intricate "weaving" effects by redrawing portions of overlapping shapes as in Figure 4, "Linked Toroids," (page 55).

The flag NOROT determines whether the X and Y coordinates will be exchanged. This is useful both for drawing the eightfold symmetrical objects (top-view toroid and sphere) and for rotating the fourfold symmetrical objects 90 degrees so that a single drawing routine can produce two orientations of an object. The calculations are basically those for the sphere, but we'll see later that we can put other objects, such as toroids, into forms that look like spheres (at least locally).

GETZ calculates the effective Z coordinate for a spherelike surface, given the X and Y coordinates relative to the local center of curvature. Note that this routine needs the SQRT function, and because it is in general called for each quartet of points plotted, our fast SQRT makes a big difference in execution speed.

SPHERE follows the previously outlined algorithm and takes full advantage of the available symmetry. The radius is POKEd into RADIUS, and clipping distances relative to the sphere center are POKEd into CLIPL (left), CLIPR (right), CLIPU (up), and CLIPD (down).

CYLNDR draws a cylinder with sizes POKEd into RADIUS and HLEN (half-length). The flag NOROT determines whether the cylinder will be drawn with a horizontal or a vertical axis. The routine is simple, using PTPLOT for the main shade-calculation tasks. Actually, we could write a much faster routine to take advantage of the fact that we can use the same shade value for all points along lines parallel to the cylinder axis. This would require considerably more space, however.

The peculiar manipulations of relative coordinates for plotting toroids in various orientations are easier to explain with the help of the diagram in Figure 5 (appropriate for a top-view toroid) (page 55).

Consider a point P at coordinates (XREL, YREL) relative to the center of the object. We can take the local center of curvature of the toroid to be at the point C, which lies on the intersection of the line passing through the center of the toroid and a point beneath P (where Z=0) with a circle of radius RC (the "average" radius of the toroid is RC=(RO+RI) / 2). We determine the surface normal by the relative X, Y, and Z displacements from the point C.

The byte R0 is used for temporary storage of the radial distance from the toroid center to the point beneath P (where Z=0):

```
R0 = SQR(XREL*XREL
   + YREL*YREL)
```

We can determine the relative X and Y displacements, XSHD and YSHD, respectively, of P from C easily by using similar triangles:

```
XSHD = XREL*(1 - RC/R0)
YSHD = YREL*(1 - RC/R0)
```

We find the Z coordinate via the Pythagorean theorem, using the radius of the "ring" portion, RT, as the hypotenuse. RT is then also the length of the normal vector and is the value POKEd into RADIUS for normalization in the shade calculations.

The routine TOROID follows this algorithm, taking advantage of the eightfold symmetry in the object. EDGTOR and SPOOL use similar center of curvature coordinates to draw other orientations of a toroid. I have given the BASIC equivalents but leave it to readers to draw the appropriate geometric constructions if they desire more details.

Using the Shape-Drawing Routines—Interface to BASIC

The last module, completing our graphics package, is a convenient interface to BASIC. Listing Five (page 76) is a collection of routines to pass parameters easily to the graphics routines from the BASIC programs that control image generation. These routines are specific to the Commodore 64 in that they use some of the routines in the BASIC ROM to interpret statements in a BASIC program or entered in the direct command mode.

Graphics commands are given in the form:

```
SYS(FNCTN),PARAM1,PARAM2,
   [OPTIONAL PARAMETERS]
```

where FNCTN is the address of the graphic function you desire (or a variable that has been assigned the address value) and PARAM1 and PARAM2 are parameters (such as the center coordinates of an object) that the function requires, followed by any optional parameters.

The parameters can be either literal numerics or expressions that are evaluated for the desired values. We do not have to put the function address in parentheses, but it helps to make the program more readable. Separating parameters by commas is required. If you desire an optional parameter (such as specifying a new radius for sphere drawing), you add it by following the last required parameter by a comma and then the optional value or expression.

The use of optional parameters is more easily explained with the sequence of BASIC statements found in Figure 8 (page 59):

For all the shape-drawing functions, you must specify the X and Y center coordinates, but shape sizes are optional parameters. The last values specified become the new default sizes, allowing you to draw copies of similar objects (such as a bunch of grapes) just by setting the new center coordinates. All shapes but the sphere take two size parameters, and you must specify both when you desire a change (even when only one parameter takes on a new value). All size parameters (radii and cylinder half-lengths) must be less than 256, not a severe limitation considering the screen is only 200 pixels high ("240" with scaling).

The commands to clear the bit map and initialize the color map each take a single optional parameter, but the default values remain unchanged. Unless you specify alternate bytes, the bit

map is filled with 0s (cleared) and the color map is filled with 1s (for black dots on a white background). The most useful alternate byte with which to fill the bit map is 255, setting the entire background. Because the shading process clears, as well as sets, appropriate points, objects still appear normal but against a black background (particularly effective when you use the "back light" style).

The byte used to fill the color map is made up of background and foreground nibbles. To initialize the color map for a different color combination than black on white, use:

```
SYS(52001),16*DC+BC
```

where DC is the dot-color number and BC is the background-color number (*as given in* any reference to programming the Commodore 64).

The addresses for all the graphic functions and the parameters they take, along with locations to be POKEd with clipping values and flags for various drawing styles, are summarized in Table 3 (page 60). The best way to see how these shape-drawing routines are used is to examine the demonstration programs in Listing Six (page 78), "SHAPES DEMO," and Listing Seven (page 80), "STELLATION." All the different style options are exercized there. It is useful to save an abbreviated version of SHAPES DEMO, consisting of just the lines up to 340 and the subroutines following line 1620. This skeleton program can then form a base, providing all the POKE and SYS locations you need and to which you can add your own image-generation control program.

Auxiliary Programs

SHAPES DEMO and STELLATION make use of two auxiliary programs that, while not strictly needed to produce graphic images, provide utilities to both enhance the image itself and speed the drawing process. Listing Eight (page 80) is a routine for sorting an integer array indirectly through a key array (to determine drawing order for facets in a polygon mesh quickly). Listing Nine (page 82) allows you to add text to graphic

```
10   REM ORDERED DITHER ARRAY—RECURSIVE FILL
20   I=0:J=0:K=1:N=0:P=3
30   REM 'P' IS ORDER OF ARRAY, SIZE IS (2 ^ P) x (2 ^ P)
40   P=2 ^ P:DIM A(P - 1,P - 1)
50   GOSUB 100
60   END
100  IF K=P THEN A(I,J)=N:N=N+1:K=K/2:RETURN
110  K=2*K:GOSUB 100
120  I=I+K:J=J+K:K=2*K:GOSUB 100
130  I=I-K:K=2×K:GOSUB 100
140  I=I+K:J=J - K:K=2*K:GOSUB 100
150  I=I - K:K=K/2:RETURN
```

Figure 7

```
...  (SET UP GRAPHICS, STYLES, ETC)
200  SP=52119:REM ADDRESS OF SPHERE FUNCTION
210  SYS(SP),80,75,30:REM DRAW A SPHERE OF RADIUS 30 AT X=80, Y=75
220  SYS(SP),300,50 :REM DRAW ANOTHER SPHERE AT X=300, y=50
230  REM SINCE NO RADIUS IS SPECIFIED, LAST VALUE IS USED AS DEFAULT
240  SYS(SP),200,150,40:REM NEW SPHERE RADIUS (40) BECOMES DEFAULT
```

Figure 8

images in a variety of styles.

The code for these routines is short enough to be tucked in behind Commodore's DOS Wedge program so that, again, you don't lose any BASIC program space but maintain full *compatibility* with the wedge. If you are willing to give up the convenience of the wedge, however, you can easily relocate these routines to the end of the "interface" subprogram to keep the graphics package in one piece.

The KEYSORT routine works with two 1-dimensional integer arrays of the same size. One is filled with some "priority" parameter (such as the "average Z" coordinates for the facets in a polygon mesh), and the other becomes a "key" array whose elements index those in the "priority" array in increasing order.

To use the routine, POKE the maximum element index of the arrays into location 140, the address of the first (0th) element of the key array into locations 251 (low byte) and 252 (high byte), and the base address of the priority array into 253 and 254. You can find these array base addresses easily because the Commodore 64 places the address of the "current BASIC variable" into locations 71 and 72. Setting the 0th ele-

BIT-MAP SCREEN : 40960-48959 ($A000-$BF3F)

COLOR MAP : 33792-34791 ($8400-$87E7)

CLIPPING BOUNDS (relative to object center):

893	($037D)	: LEFT BOUND
894	($037E)	: RIGHT BOUND
895	($037F)	: UPPER BOUND
896	($0380)	: LOWER BOUND
53280	($D020)	: BORDER COLOR [0-15], POKE with 1 to match white screen

STYLE FLAGS:

838	($0346)	: SHADE STYLE [0=random, 1=halftone]
839	($0347)	: SCALE FLAG [0=normal (1:1), 1=scaled (3:4)]
868	($0364)	: EDGES FLAG [0=normal, 1=add lines to facet edges]
871	($0367)	: EDGE/LINE MODE [0=draw, 1=erase]
898	($0382)	: LIGHTING STYLE [0=normal single source, 1=backlit]

FUNCTION LOCATIONS—CALL WITH "SYS(FNCTN), PARAMETERS." OPTIONAL PARAMETERS IN SQUARE BRACKETS, DEFAULT VALUES ARE USED IF NOT SPECIFIED

49378	($C0E2)	: SWITCH TO GRAPHICS MODE
49411	($C103)	: RETURN TO TEXT SCREEN
51979	($CB0B)	: CLEAR[,CLEAR BYTE] FILL BIT MAP WITH CLEAR BYTE [DEFAULT = 0]
52001	($CB21)	: COLOR[,COLOR BYTE] FILL COLOR MAP WITH COLOR BYTE
52023	($CB37)	: PLOT,X,Y SET POINT AT (X,Y)
52026	($CB3A)	: UNPLOT,X,Y CLEAR POINT AT (X,Y)
52049	($CB51)	: LINE,X1,Y1,X2,Y2 DRAW A LINE BETWEEN (X1,Y1) AND (X2,Y2)
52052	($CB54)	: FACET,X1,Y1,X2,Y2,X3,Y3,VA DRAW A SHADED TRIANGULAR FACET BETWEEN COORDINATE PAIRS (X1,Y1), (X2,Y2) AND (X3,Y3) USING "VA" SHADE VALUE (0:BLACK TO 64:WHITE)

CURVED SURFACES:

52119	($CB97)	: SPHERE,X,Y[,R] DRAW A SPHERE CENTERED AT (X,Y) WITH RADIUS R
52141	($CBAD)	: TOROID,X,Y[,RI,RO] DRAW A TOP-VIEW TOROID WITH INNER RADIUS RI AND OUTER RADIUS RO
52150	($CBB6)	: VCYL,X,Y[,R,HL] DRAW A CYLINDER WITH AXIS VERTICAL, RADIUS R AND HALF-LENGTH HL
52153	($CBB9)	: HCYL,X,Y[,R,HL] DRAW A CYLINDER WITH AXIS HORIZONTAL
52186	($CBDA)	: VTOR,X,Y[,RI,RO] DRAW AN EDGE-VIEW TOROID WITH AXIS VERTICAL
52189	($CBDD)	: HTOR,X,Y[,RI,RO] DRAW AN EDGE-VEIW TOROID WITH AXIS HORIZONTAL
52203	($CBEB)	: VSPOOL,X,Y[,RI,RO] DRAW AN INSIDE-VIEW TOROID ("SPOOL") WITH AXIS VERTICAL
52206	($CBEE)	: HSPOOL,X,Y[,RI,RO] DRAW AN INSIDE-VIEW TOROID ("SPOOL") WITH AXIS HORIZONTAL

Table 3
Graphics Addresses

ment of an array equal to itself merely establishes it as the current variable. Then you can transfer the contents of 71 and 72 to the proper pointer at 251/252 or 253/254. Be careful to use literal numerics for "251" and so on, because you don't want to change the current variable. After setting up the pointers, SYS to the start of the KEYSORT routine. Another caution to observe if you use KEYSORT for other applications is *not* to create any new variables between finding the array base addresses and invoking KEYSORT, because arrays will be pushed up in memory to keep them at the end of BASIC's variable storage.

KEYSORT is set up to handle arrays with up to 256 elements each. If needed, you can sort larger arrays by breaking them into smaller arrays, sorting with KEYSORT, and then collating the results. Merging a few already sorted arrays is a fairly simple operation that you can do in a single pass, extending the usefulness of this routine.

STELLATION uses KEYSORT (lines 840–880) to determine the drawing order for the facets of a small stellated dodecahedron (Figure 6, page 55). This is a polyhedron that is not strictly convex (some internal dihedral angles are greater than 180 degrees), causing some foreground facets to partially obscure those in the background. The Painter's algorithm is used to handle hidden surface removal. Each newly drawn facet completely overwrites the background (clearing, as well as setting, points) so that drawing facets from back to front gives a proper rendering of the object.

The stellated dodecahedron of Figure 6 usually has only 30 visible or partially visible facets. Sorting with BASIC would be satisfactory here, but more complex polygon mesh objects can easily contain hundreds of visible and partially visible facets. You can really appreciate KEYSORT in those cases. Although it uses the simple and inefficient bubble-sort algorithm, being implemented in machine code lets KEYSORT run circles around any BASIC sorting alternative.

The final routine, in Listing Nine, WRITE, allows you to add text to the graphic display. The primary reason for implementing this as a machine code routine is that the bit map for the graphic image is located in the shadow RAM beneath the BASIC ROM (again saving useful BASIC program memory space). We can transfer bit images from the character ROM to the bit map just by POKEing values to the bit map, but we gain more flexibility if we first read the bit-map bytes to exclusive OR them, AND them, and so on with character image. Reading the shadow RAM requires that the BASIC ROM be switched out (through the bank switching used in the Commodore 64), something a BASIC program cannot do (and hope to continue execution).

The subroutine following line 1780 in SHAPES DEMO illustrates how to print text in a variety of styles on the bit-map image. Although the program actually uses only one style, the subroutine allows for five possibilities. Normal ("black" on "white") characters and reverse characters can overwrite the background, black characters can be ORed with the background, white characters ANDed with the background, or black characters exclusive ORed with the background. The last possibility gives characters that "change phase" across black/white edges in the image.

Conclusion

The graphics package described here we hope presents some new approaches to computer-generated images on small systems. At the same time, I'm sure I've missed some obvious tricks that would speed up the programs or reduce their size. I will be interested to see responses to this article and shortcuts or enhancements that might be added. Although the price/performance ratio of small graphics system continues to improve (especially with the development of custom LSI graphics chips), there are yet many unexplored software approaches to the problem. **DDJ**

Drawing on the C-64 (Text begins on page 50)

Listing One

```
00001  0000                ; INTEGER ARITHMETIC ROUTINES
00002  0000                ;
00003  0000                ; RICHARD L. RYLANDER    8/12/84
00004  0000                ;
00005  0000                ; REVISED 10/29/84 TO ADD FULL DOUBLE
00006  0000                ; PRECISION ARGUMENTS IN DIVIDE ROUTINE
00007  0000                ;
00008  0000                ;****************************************
00009  0000                ;
00010  0000                ; USE PAGE ZERO LOCATIONS WHERE POSSIBLE FOR
00011  0000                ; ITERATIVE PROCEDURE WORK SPACE
00012  0000                ;
00013  0000                MLPCND =$AC             ; MULTIPLICAND
00014  0000                MLPLIER =$AD            ; MULTIPLIER
00015  0000                PROD   =$AE             ; PRODUCT
00016  0000                ;
00017  0000                DVDND  =$FD             ; DIVIDEND/QUOTIENT
00018  0000                DVSOR  =$FB             ; DIVISOR
00019  0000                RMNDR  =$B4             ; REMAINDER
00020  0000                ;
00021  0000                RADCND =$AC             ; RADICAND
00022  0000                ROOT   =$033C           ; SQUARE ROOT
00023  0000                ;
00024  0000                TEMP   =$FB
00025  0000                ;
00026  0000                ; SET UP SEED VALUES FOR PSEUDO RANDOM NUMBERS
00027  0000                       *=$C000
00028  C000  FF            RNDM   .BYTE $FF,$55
00028  C001  55
00029  C002  00            RTEMP  .BYTE $00,$00
00029  C003  00
00030  C004                ;
00031  C004                ;
00032  C004                ;****************************************
00033  C004                ;
00034  C004                ; MULTIPLY SINGLE PRECISION MULTIPLICAND
00035  C004                ; BY SINGLE PRECISION MULTIPLIER GIVING
00036  C004                ; DOUBLE PRECISION PRODUCT (ENTER AT "MULT")
00037  C004                ;
00038  C004                ; SPECIAL CASE: ENTER AT "SQUARE" TO FIND
00039  C004                ; SQUARE OF SIGNED 8-BIT NUMBER
00040  C004                ;
00041  C004  A5 AC         SQUARE LDA MLPCND       ; ENTRY TO SQUARE
00042  C006  10 07                BPL POSITV      ; USE ABSOLUTE VALUE
00043  C008  38                  SEC             ; NEGATE IF NEEDED
00044  C009  A9 00                LDA #$00
00045  C00B  E5 AC               SBC MLPCND
00046  C00D  85 AC               STA MLPCND
00047  C00F  85 AD         POSITV STA MLPLER
00048  C011  A9 00         MULT   LDA #$00         ; ENTRY TO MULTIPLY
00049  C013  A2 08                LDX #$08
00050  C015  46 AD         MLOOP  LSR MLPLER
00051  C017  90 03                BCC NOADD
00052  C019  18                  CLC
00053  C01A  65 AC                ADC MLPCND
00054  C01C  6A            NOADD  ROR A
00055  C01D  66 AE                ROR PROD
00056  C01F  CA                   DEX
00057  C020  D0 F3                BNE MLOOP
00058  C022  85 AF                STA PROD+1
00059  C024  60                   RTS
00060  C025                ;
00061  C025                ;****************************************
```

(Continued on next page)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing One

```
00062  C025                ;
00063  C025                ; DIVIDE DOUBLE PRECISION DIVIDEND
00064  C025                ; BY DOUBLE PRECISION DIVISOR GIVING
00065  C025                ; DOUBLE PRECISION QUOTIENT
00066  C025                ;
00067  C025                ; DIVIDEND IS REPLACED BY QUOTIENT
00068  C025                ; IN THE PROCESS
00069  C025                ;
00070  C025                ; QUOTIENT IS ROUNDED TO NEAREST INTEGER
00071  C025                ;
00072  C025  A9 00         DIVIDE LDA #$00
00073  C027  85 B4                STA RMNDR
00074  C029  85 B5                STA RMNDR+1
00075  C02B  A2 10                LDX #$10
00076  C02D  26 FD         DLOOP  ROL DVDND
00077  C02F  26 FE                ROL DVDND+1
00078  C031  26 B4                ROL RMNDR
00079  C033  26 B5                ROL RMNDR+1
00080  C035  38                   SEC
00081  C036  A5 B4                LDA RMNDR
00082  C038  E5 FB                SBC DVSOR
00083  C03A  AB                   TAY
00084  C03B  A5 B5                LDA RMNDR+1
00085  C03D  E5 FC                SBC DVSOR+1
00086  C03F  90 04                BCC DECCNT
00087  C041  84 B4                STY RMNDR
00088  C043  85 B5                STA RMNDR+1
00089  C045  CA            DECCNT DEX
00090  C046  D0 E5                BNE DLOOP
00091  C048  26 FD                ROL DVDND          ; CHECK IF REMAINDER
00092  C04A  26 FE                ROL DVDND+1        ; IS >= 1/2 OF DIVIDEND
00093  C04C  06 B4                ASL RMNDR          ; FOR ROUNDING
00094  C04E  26 B5                ROL RMNDR+1
00095  C050  B0 0B                BCS ROUND
00096  C052  38                   SEC
00097  C053  A5 FB                LDA DVSOR
00098  C055  E5 B4                SBC RMNDR
00099  C057  A5 FC                LDA DVSOR+1
00100  C059  E5 B5                SBC RMNDR+1
00101  C05B  B0 06                BCS NOCHNG
00102  C05D  E6 FD         ROUND  INC DVDND
00103  C05F  D0 02                BNE NOCHNG
00104  C061  E6 FE                INC DVDND+1
00105  C063  60            NOCHNG RTS
00106  C064                ;
00107  C064                ;*****************************************
00108  C064                ;
00109  C064                ; TAKE INTEGER SQUARE ROOT OF A
00110  C064                ; DOUBLE PRECISION RADICAND GIVING
00111  C064                ; SINGLE PRECISION ROOT ( <= REAL ROOT )
00112  C064                ;
00113  C064  A2 08         SQRT   LDX #$08
00114  C066  A9 00                LDA #$00
00115  C068  8D 3C 03             STA ROOT
00116  C06B  8D 3D 03             STA ROOT+1
00117  C06E  85 FB                STA TEMP
00118  C070  85 FC                STA TEMP+1
00119  C072  0E 3C 03      SQRT1  ASL ROOT
00120  C075  2E 3D 03             ROL ROOT+1
00121  C078  EE 3C 03             INC ROOT           ; ASSUME CURRENT LSB OF
00122  C07B  D0 03                BNE NEXT1          ; ROOT WILL BE 1
00123  C07D  EE 3D 03             INC ROOT+1
00124  C080  06 AC         NEXT1  ASL RADCND         ; SHIFT RADICAND LEFT
00125  C082  26 AD                ROL RADCND+1       ; TWICE INTO TEMP
00126  C084  26 FB                ROL TEMP
00127  C086  26 FC                ROL TEMP+1
00128  C088  06 AC                ASL RADCND
00129  C08A  26 AD                ROL RADCND+1
00130  C08C  26 FB                ROL TEMP
00131  C08E  26 FC                ROL TEMP+1
00132  C090  38                   SEC                ; SUBTRACT ROOT ESTIMATE
00133  C091  A5 FB                LDA TEMP           ; FROM TEMP
00134  C093  ED 3C 03             SBC ROOT
00135  C096  AB                   TAY
00136  C097  A5 FC                LDA TEMP+1
00137  C099  ED 3D 03             SBC ROOT+1
00138  C09C  90 12                BCC RESTOR
00139  C09E  85 FC                STA TEMP+1         ; SUBTRACTION OK
00140  C0A0  84 FB                STY TEMP
00141  C0A2  EE 3C 03             INC ROOT
00142  C0A5  D0 03                BNE NEXT2
00143  C0A7  EE 3D 03             INC ROOT+1
00144  C0AA  CA            NEXT2  DEX
00145  C0AB  D0 C5                BNE SQRT1
00146  C0AD  4C C1 C0             JMP FINI
00147  C0B0  38            RESTOR SEC                ; IGNORE SUBTRACTION
00148  C0B1  AD 3C 03             LDA ROOT           ; AND RESET LSB OF ROOT
00149  C0B4  E9 01                SBC #$01
00150  C0B6  8D 3C 03             STA ROOT
00151  C0B9  B0 03                BCS NEXT3
00152  C0BB  CE 3D 03             DEC ROOT+1
00153  C0BE  CA            NEXT3  DEX
00154  C0BF  D0 B1                BNE SQRT1
00155  C0C1  6E 3D 03      FINI   ROR ROOT+1         ; FINAL /2 TO NORMALIZE
00156  C0C4  6E 3C 03             ROR ROOT
00157  C0C7  60                   RTS
00158  C0C8                ;
00159  C0C8                ;*****************************************
00160  C0C8                ;
00161  C0C8                ; GENERATE PSEUDO-RANDOM BYTES
00162  C0C8                ; EXIT WITH P-R BYTE IN ACCUM.
00163  C0C8                ;
00164  C0C8  AD 00 C0      RANDOM LDA RNDM
00165  C0CB  8D 02 C0             STA RTEMP
00166  C0CE  4D 01 C0             EOR RNDM+1
00167  C0D1  2E 03 C0             ROL RTEMP+1        ; RTEMP+1 PRESERVES
00168  C0D4  6A                   ROR A              ; CARRY BIT FOR CYCLING
00169  C0D5  6E 03 C0             ROR RTEMP+1        ; RANDOM NUMBERS
00170  C0D8  8D 00 C0             STA RNDM
00171  C0DB  AD 02 C0             LDA RTEMP
00172  C0DE  8D 01 C0             STA RNDM+1
00173  C0E1  60                   RTS
00174  C0E2                .END

ERRORS = 00000

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 DECCNT   C045    DIVIDE   C025    DLOOP   C02D    DVDND   00FD
 DVSOR    00FB    FINI     C0C1    MLOOP   C015    MLPCND  00AC
 MLPLER   00AD    MULT     C011    NEXT1   C080    NEXT2   C0AA
 NEXT3    C0BE    NOADD    C01C    NOCHNG  C063    POSITV  C00F
 PROD     00AE    RADCND   00AC    RANDOM  C0CB    RESTOR  C0B0
 RMNDR    00B4    RNDM     C000    ROOT    033C    ROUND   C05D
 RTEMP    C002    SQRT     C064    SQRT1   C072    SQUARE  C004
 TEMP     00FB

END OF ASSEMBLY
```

End Listing One

Listing Two

```
00001  0000                ; GRAPHICS UTILITIES
00002  0000                ;
00003  0000                ; RICHARD L. RYLANDER   11/4/84
00004  0000                ;
00005  0000                ; LOAD ARITHMETIC UTILITIES FIRST
00006  0000                ;
00007  0000                RAM=$033E
00008  0000                ORIGIN=$C0E2
00009  0000                ;
00010  0000                MLPCND=$AC              ; MULTIPLICAND (S)
00011  0000                MLPLER=$AD              ; MULTIPLIER (S)
00012  0000                PROD=$AE                ; PRODUCT (D)
00013  0000                MULT=$C011              ; CALL FOR MULTIPLY
00014  0000                ;
00015  0000                RNDM=$C000              ; RANDOM NUMBER
00016  0000                RANDOM=$C0CB            ; CALL FOR RANDOM
00017  0000                ; NOTE - A CALL TO 'RANDOM' LEAVES A RANDOM BYTE
00018  0000                ; IN THE ACCUMULATOR
00019  0000                ;
00020  0000                      *=RAM
00021  033E                PLTFLG *=*+1            ; PLOT/UNPLOT FLAG
00022  033F                XPLT   *=*+2            ; ABSOLUTE PLOT COORD
00023  0341                YPLT   *=*+1            ; ABSOLUTE PLOT COORD
00024  0342                VIC1   *=*+1            ; REGISTER STORAGE
00025  0343                VIC2   *=*+1            ; REGISTER STORAGE
00026  0344                VALUE  *=*+2            ; FINAL NORMALIZED SHADE VALUE
00027  0346                HTORRN *=*+1            ; SHADE FLAG, 1=HALFTONE
00028  0347                NOSCAL *=*+1            ; SCALE FLAG, 1=NO SCALE
00029  0348                TEMP   *=*+2            ; TEMPORARY STORAGE
00030  034A                ;
00031  034A                      *=ORIGIN
00032  C0E2                ;
00033  C0E2                ;*****************************************
00034  C0E2                ;
00035  C0E2                ; TURN ON BIT MAP GRAPHICS MODE,
00036  C0E2                ; SAVING REGISTER VALUES FOR
00037  C0E2                ; RETURN TO TEXT MODE LATER.
00038  C0E2                ;
00039  C0E2  AD 11 D0      GRFON  LDA $D011
00040  C0E5  09 20                ORA #$20
00041  C0E7  8D 11 D0             STA $D011
00042  C0EA  AD 00 DD             LDA $DD00
00043  C0ED  8D 42 03             STA VIC1
00044  C0F0  29 FC                AND #$FC
00045  C0F2  09 01                ORA #$01
00046  C0F4  8D 00 DD             STA $DD00
00047  C0F7  AD 18 D0             LDA $D018
00048  C0FA  8D 43 03             STA VIC2
00049  C0FD  A9 19                LDA #$19
00050  C0FF  8D 18 D0             STA $D018
00051  C102  60                   RTS
00052  C103                ;
00053  C103                ;*****************************************
00054  C103                ;
00055  C103                ; RETURN TO TEXT SCREEN
00056  C103                ;
00057  C103  AD 11 D0      GRFOFF LDA $D011
00058  C106  29 DF                AND #$DF
00059  C108  8D 11 D0             STA $D011
00060  C10B  AD 42 03             LDA VIC1
00061  C10E  8D 00 DD             STA $DD00
00062  C111  AD 43 03             LDA VIC2
00063  C114  8D 18 D0             STA $D018
00064  C117  60                   RTS
00065  C118                ;
00066  C118                ;*****************************************
00067  C118                ;
00068  C118                ; FILL COLOR MAP FOR BLACK DOTS ON WHITE
00069  C118                ;
00070  C118  A9 01         COLOR  LDA #$01         ; POKE NEW COLORS HERE
00071  C11A  A2 00                LDX #0
00072  C11C  9D 00 84      COL1   STA $8400,X
00073  C11F  9D 00 85             STA $8500,X
00074  C122  9D 00 86             STA $8600,X
00075  C125  9D 00 87             STA $8700,X
00076  C128  CA                   DEX
00077  C129  D0 F1                BNE COL1
00078  C12B  60                   RTS
00079  C12C                ;
00080  C12C                ;*****************************************
00081  C12C                ;
00082  C12C                ; CLEAR HI-RES GRAPHICS SCREEN
00083  C12C                ;
00084  C12C  A9 A0         CLEAR  LDA #$A0
00085  C12E  85 FC                STA $FC
00086  C130  A0 00                LDY #0
00087  C132  84 FB                STY $FB
00088  C134  A9 00                LDA #0          ; CLEAR BYTE
00089  C136  A2 20                LDX #$20
00090  C138  91 FB         CLRLP  STA ($FB),Y
00091  C13A  C8                   INY
00092  C13B  D0 FB                BNE CLRLP
00093  C13D  E6 FC                INC $FC
00094  C13F  CA                   DEX
00095  C140  D0 F6                BNE CLRLP
00096  C142  60                   RTS
00097  C143                ;
00098  C143                ;*****************************************
00099  C143                ;
00100  C143                ; PLOT AND UNPLOT POINTS ON HI-RES GRAPHICS
00101  C143                ; SCREEN. ABSOLUTE X AND Y SCREEN COORDINATES
00102  C143                ; ARE POKED INTO XPLT, XPLT+1, AND YPLT
00103  C143                ;
00104  C143  A9 00         PLOT   LDA #0
00105  C145  2C                   .BYTE $2C
00106  C146  A9 80         UNPLOT LDA #$80
00107  C148  8D 3E 03             STA PLTFLG
00108  C14B  A5 01                LDA $01         ; BASIC ROM OUT
00109  C14D  29 FE                AND #$FE
00110  C14F  85 01                STA $01
00111  C151  38                   SEC             ; INVERT Y COORDINATE TO
00112  C152  A9 C7                LDA #$C7        ; PUT ORIGIN IN LOWER LEFT
00113  C154  ED 41 03             SBC YPLT        ; CORNER OF SCREEN
00114  C157  AA                   TAX             ; (199.-YPLT)
00115  C158  4A                   LSR A
00116  C159  4A                   LSR A
00117  C15A  4A                   LSR A
00118  C15B  A8                   TAY
00119  C15C  B9 AB C1             LDA TABLE1,Y
00120  C15F  85 FB                STA $FB
00121  C161  B9 C4 C1             LDA TABLE2,Y
00122  C164  85 FC                STA $FC
```

(Continued on page 64)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Two

```
00123  C166  8A              TXA
00124  C167  29 07           AND #$07
00125  C169  18              CLC
00126  C16A  65 FB           ADC $FB
00127  C16C  85 FB           STA $FB
00128  C16E  AD 3F 03        LDA XPLT
00129  C171  29 F8           AND #$F8
00130  C173  65 FB           ADC $FD
00131  C175  85 FB           STA $FD
00132  C177  AD 40 03        LDA XPLT+1
00133  C17A  65 FC           ADC $FC
00134  C17C  85 FC           STA $FC
00135  C17E  A9 A0           LDA #$A0
00136  C180  65 FC           ADC $FC
00137  C182  85 FC           STA $FC
00138  C184  AD 3F 03        LDA XPLT
00139  C187  29 07           AND #$07
00140  C189  49 07           EOR #$07
00141  C18B  AA              TAX
00142  C18C  A9 01           LDA #$01
00143  C18E  CA        PLOTLP DEX
00144  C18F  30 03           BMI PLOT2
00145  C191  0A              ASL A
00146  C192  D0 FA           BNE PLOTLP
00147  C194  A0 00     PLOT2 LDY #0
00148  C196  2C 3E 03        BIT PLTFLG
00149  C199  10 05           BPL NOPLOT
00150  C19B  49 FF           EOR #$FF
00151  C19D  31 FB           AND ($FB),Y
00152  C19F  2C              .BYTE $2C
00153  C1A0  11 FB     NOPLOT ORA ($FB),Y
00154  C1A2  91 FB           STA ($FD),Y
00155  C1A4  A5 01           LDA $01          ; BASIC ROM RESTORED
00156  C1A6  09 01           ORA #$01
00157  C1A8  85 01           STA $01
00158  C1AA  60              RTS
00159  C1AB            ;
00160  C1AB  00        TABLE1 .BYTE $00,$40,$80,$C0
00160  C1AC  40
00160  C1AD  80
00160  C1AE  C0
00161  C1AF  00              .BYTE $00,$40,$80,$C0
00161  C1B0  40
00161  C1B1  80
00161  C1B2  C0
00162  C1B3  00              .BYTE $00,$40,$80,$C0
00162  C1B4  40
00162  C1B5  80
00162  C1B6  C0
00163  C1B7  00              .BYTE $00,$40,$80,$C0
00163  C1B8  40
00163  C1B9  80
00163  C1BA  C0
00164  C1BB  00              .BYTE $00,$40,$80,$C0
00164  C1BC  40
00164  C1BD  80
00164  C1BE  C0
00165  C1BF  00              .BYTE $00,$40,$80,$C0,$00
00165  C1C0  40
00165  C1C1  80
00165  C1C2  C0
00165  C1C3  00
00166  C1C4            ;
00167  C1C4  00        TABLE2 .BYTE $00,$01,$02,$03
00167  C1C5  01
00167  C1C6  02
00167  C1C7  03
00168  C1C8  05              .BYTE $05,$06,$07,$08
00168  C1C9  06
00168  C1CA  07
00168  C1CB  08
00169  C1CC  0A              .BYTE $0A,$0B,$0C,$0D
00169  C1CD  0B
00169  C1CE  0C
00169  C1CF  0D
00170  C1D0  0F              .BYTE $0F,$10,$11,$12
00170  C1D1  10
00170  C1D2  11
00170  C1D3  12
00171  C1D4  14              .BYTE $14,$15,$16,$17
00171  C1D5  15
00171  C1D6  16
00171  C1D7  17
00172  C1D8  19              .BYTE $19,$1A,$1B,$1C,$1E
00172  C1D9  1A
00172  C1DA  1B
00172  C1DB  1C
00172  C1DC  1E
00173  C1DD            ;
00174  C1DD            ;*****************************************
00175  C1DD            ;
00176  C1DD            ; SHADING BY HYBRID DITHER/DOT-GROWTH
00177  C1DD            ;
00178  C1DD  AD 3F 03  SHADE  LDA XPLT         ; USE BITS -----***
00179  C1E0  29 07            AND #$07         ; OF 'X' SCREEN COORD
00180  C1E2  8D 48 03         STA TEMP
00181  C1E5  AD 41 03         LDA YPLT         ; AND BITS -----***
00182  C1E8  29 07            AND #$07         ; OF 'Y' SCREEN COORD
00183  C1EA  0A               ASL A            ; SHIFTED INTO --***---
00184  C1EB  0A               ASL A            ; POSITION TO DETERMINE
00185  C1EC  0A               ASL A            ; 6-BIT OFFSET IN
00186  C1ED  0D 48 03         ORA TEMP         ; THRESHOLD TABLE
00187  C1F0  AA               TAX
00188  C1F1  BD 2F C2         LDA THRESH,X     ; SCREEN-POSITION-WEIGHTED
00189  C1F4  CD 44 03         CMP VALUE        ; THRESHOLD VALUE
00190  C1F7  10 03            BPL GREATR
00191  C1F9  4C 46 C1         JMP UNPLOT
00192  C1FC  4C 43 C1  GREATR JMP PLOT
00193  C1FF            ;
00194  C1FF            ;*****************************************
00195  C1FF            ;
00196  C1FF            ; SHADING BY RANDOM HALFTONE
00197  C1FF            ;
00198  C1FF  20 C8 C0  RSHADE JSR RANDOM
00199  C202  4A               LSR A            ; REDUCE RANDOM BYTE
00200  C203  4A               LSR A            ; TO 6 BITS FOR SHADE
00201  C204  CD 44 03         CMP VALUE        ; VALUE COMPARISON
00202  C207  10 03            BPL MORE
00203  C209  4C 46 C1         JMP UNPLOT
00204  C20C  4C 43 C1  MORE   JMP PLOT
00205  C20F            ;
00206  C20F            ;*****************************************
00207  C20F            ;
00208  C20F            ; PLOT A POINT WEIGHTED BY SHADING SCHEME
00209  C20F            ; AND SHADE VALUE
00210  C20F            ; CHECK 'NOSCAL' FLAG FOR SCALING OF Y COORD
00211  C20F            ; CHECK 'HTORRN' FLAG FOR TYPE OF SHADING
00212  C20F            ;
00213  C20F  AD 47 03  PLTSHD LDA NOSCAL
00214  C212  F0 10            BEQ NORM
00215  C214            ;
00216  C214            ; SCALE Y FROM 0-239 PSEUDO-COORDINATES
00217  C214            ; TO 0-199 TRUE SCREEN COORDINATES BY
00218  C214            ;     Y = (Y+1)*213/256
00219  C214            ;
00220  C214  AC 41 03  SCALE  LDY YPLT
00221  C217  C8               INY
00222  C218  84 AD            STY MLPLER
00223  C21A  A9 D5            LDA #$D5         ; 213.
00224  C21C  85 AC            STA MLPCND
00225  C21E  20 11 C0         JSR MULT         ; RETURN WITH HIGH BYTE
00226  C221  8D 41 03         STA YPLT         ; IN ACCUMULATOR
00227  C224  AD 46 03  NORM   LDA HTORRN
00228  C227  F0 03            BEQ RPLT
00229  C229  4C DD C1         JMP SHADE
00230  C22C  4C FF C1  RPLT   JMP RSHADE
00231  C22F            ;
00232  C22F            ;
00233  C22F            ;
00234  C22F  00        THRESH .BYTE $00,$08,$35,$3D
00234  C230  08
00234  C231  35
00234  C232  3D
00235  C233  02              .BYTE $02,$0A,$37,$3F
00235  C234  0A
00235  C235  37
00235  C236  3F
00236  C237  10              .BYTE $10,$18,$25,$2D
00236  C238  18
00236  C239  25
00236  C23A  2D
00237  C23B  12              .BYTE $12,$1A,$27,$2F
00237  C23C  1A
00237  C23D  27
00237  C23E  2F
00238  C23F  31              .BYTE $31,$39,$04,$0C
00238  C240  39
00238  C241  04
00238  C242  0C
00239  C243  33              .BYTE $33,$3B,$06,$0E
00239  C244  3B
00239  C245  06
00239  C246  0E
00240  C247  21              .BYTE $21,$29,$14,$1C
00240  C248  29
00240  C249  14
00240  C24A  1C
00241  C24B  23              .BYTE $23,$2B,$16,$1E
00241  C24C  2B
00241  C24D  16
00241  C24E  1E
00242  C24F  03              .BYTE $03,$0B,$36,$3E
00242  C250  0B
00242  C251  36
00242  C252  3E
00243  C253  01              .BYTE $01,$09,$34,$3C
00243  C254  09
00243  C255  34
00243  C256  3C
00244  C257  13              .BYTE $13,$1B,$26,$2E
00244  C258  1B
00244  C259  26
00244  C25A  2E
00245  C25B  11              .BYTE $11,$19,$24,$2C
00245  C25C  19
00245  C25D  24
00245  C25E  2C
00246  C25F  32              .BYTE $32,$3A,$07,$0F
00246  C260  3A
00246  C261  07
00246  C262  0F
00247  C263  30              .BYTE $30,$38,$05,$0D
00247  C264  38
00247  C265  05
00247  C266  0D
00248  C267  22              .BYTE $22,$2A,$17,$1F
00248  C268  2A
00248  C269  17
00248  C26A  1F
00249  C26B  20              .BYTE $20,$28,$15,$1D
00249  C26C  28
00249  C26D  15
00249  C26E  1D
00250  C26F              .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 CLEAR   C12C    CLRLP   C138    COL1    C11C    COLOR   C118
 GREATR  C1FC    GRFOFF  C103    GRFON   C0E2    HTORRN  0346
 MLPCND  00AC    MLPLER  00AD    MORE    C20C    MULT    C011
 NOPLOT  C1A0    NORM    C224    NOSCAL  0347    ORIGIN  C0E2
 PLOT    C143    PLOT2   C194    PLOTLP  C1BE    PLTFLG  033E
 PLTSHD  C20F    PROD    00AE    RAM     033E    RANDOM  C0C8
 RNDM    C000    RPLT    C22C    RSHADE  C1FF    SCALE   C214
 SHADE   C1DD    TABLE1  C1AB    TABLE2  C1C4    TEMP    0348
 THRESH  C22F    UNFLOT  C146    VALUE   0344    VIC1    0342
 VIC2    0343    XPLT    033F    YPLT    0341

END OF ASSEMBLY
```

End Listing Two

Listing Three

```
00001  0000            ; FACET - DRAW SHADED TRIANGULAR FACETS
00002  0000            ; AND STRAIGHT LINES.
00003  0000            ;
00004  0000            ; RICHARD L. RYLANDER   11/4/84
00005  0000            ;
00006  0000            ; LOAD "ARITH.HEX" AND "GRAPH.HEX"
00007  0000            ; BEFORE USING
00008  0000            ;
00009  0000  ORIGIN = $C26F
00010  0000  RAM    = $034A
00011  0000            ;
00012  0000  XPLT   = $033F
00013  0000  YPLT   = $0341
00014  0000  NORM   = $C224
00015  0000  NOSCAL = $0347
00016  0000  PLOT   = $C143
00017  0000  UNPLOT = $C146
```

(Continued on page 66)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Three

```
00018  0000             ;
00019  0000             MLPCND = $AC
00020  0000             MLPLER = $AD
00021  0000             PROD   = $AE
00022  0000             MULT   = $C011
00023  0000             ;
00024  0000             DVDND  = $FD
00025  0000             DVSOR  = $FB
00026  0000             QUOT   = $FD
00027  0000             DIVIDE = $C025
00028  0000             ;
00029  0000                    *=RAM
00030  034A             ;
00031  034A             XMIN   *=*+2
00032  034C             YMIN   *=*+1
00033  034D             XMID   *=*+2
00034  034F             YMID   *=*+1
00035  0350             XMAX   *=*+2
00036  0352             YMAX   *=*+1
00037  0353             YTOP   *=*+1
00038  0354             YBOT   *=*+1
00039  0355             YBASE  *=*+1
00040  0356             DLTAX1 *=*+2
00041  0358             DLTAX2 *=*+1
00042  0359             DLTAX3 *=*+1
00043  035A             DELTAX *=*+1
00044  035B             DLTAY1 *=*+1
00045  035C             DLTAY2 *=*+1
00046  035D             DLTAY3 *=*+1
00047  035E             DELTAY *=*+1
00048  035F             XDIFF  *=*+1
00049  0360             FLAG1  *=*+1
00050  0361             FLAG2  *=*+1
00051  0362             FLAG3  *=*+1
00052  0363             FLAG   *=*+1
00053  0364             EDGES  *=*+1
00054  0365             ERROR  *=*+2
00055  0367             MODE   *=*+1
00056  0368             COUNT  *=*+2
00057  036A             ;
00058  036A             ;
00059  036A                    *=ORIGIN
00060  C26F             ;
00061  C26F             ;*****************************************
00062  C26F             ;
00063  C26F             ; SCALE ALL Y COORDINATES FROM 0..239
00064  C26F             ; PSUEDO-COORDINATE RANGE TO 0..199
00065  C26F             ; TRUE SCREEN COORDINATE RANGE
00066  C26F             ;
00067  C26F A0 06       SCALE  LDY #6
00068  C271 A9 D5              LDA #$D5
00069  C273 85 AC              STA MLPCND
00070  C275 B9 4C 03    SCLP   LDA YMIN,Y
00071  C278 85 AD              STA MLPLER
00072  C27A 20 11 C0           JSR MULT
00073  C27D 99 4C 03           STA YMIN,Y
00074  C280 88                 DEY
00075  C281 88                 DEY
00076  C282 88                 DEY
00077  C283 10 F0              BPL SCLP
00078  C285 60                 RTS
00079  C286             ;
00080  C286             ;
00081  C286             ;*****************************************
00082  C286             ;
00083  C286             ; EXCHANGE 'MIN' AND 'MID' COORDINATES
00084  C286             ;
00085  C286 A0 02       SWAP12 LDY #2
00086  C288 B9 4A 03    LOOP1  LDA XMIN,Y
00087  C28B 48                 PHA
00088  C28C B9 4D 03           LDA XMID,Y
00089  C28F 99 4A 03           STA XMIN,Y
00090  C292 68                 PLA
00091  C293 99 4D 03           STA XMID,Y
00092  C296 88                 DEY
00093  C297 10 EF              BPL LOOP1
00094  C299 60                 RTS
00095  C29A             ;
00096  C29A             ;*****************************************
00097  C29A             ;
00098  C29A             ; EXCHANGE 'MID' AND 'MAX' COORDINATES
00099  C29A             ;
00100  C29A A0 02       SWAP23 LDY #2
00101  C29C B9 4D 03    LOOP2  LDA XMID,Y
00102  C29F 48                 PHA
00103  C2A0 B9 50 03           LDA XMAX,Y
00104  C2A3 99 4D 03           STA XMID,Y
00105  C2A6 68                 PLA
00106  C2A7 99 50 03           STA XMAX,Y
00107  C2AA 88                 DEY
00108  C2AB 10 EF              BPL LOOP2
00109  C2AD 60                 RTS
00110  C2AE             ;
00111  C2AE             ;*****************************************
00112  C2AE             ;
00113  C2AE             ; SORT COORDINATES ACCORDING TO X COMPONENTS
00114  C2AE             ;
00115  C2AE A2 02       SORTX  LDX #2
00116  C2B0 38          SORTLP SEC
00117  C2B1 AD 4D 03           LDA XMID
00118  C2B4 ED 4A 03           SBC XMIN
00119  C2B7 AD 4E 03           LDA XMID+1
00120  C2BA ED 4B 03           SBC XMIN+1
00121  C2BD B0 03              BCS NOSWP1
00122  C2BF 20 B6 C2           JSR SWAP12
00123  C2C2 CA          NOSWP1 DEX
00124  C2C3 F0 15              BEQ SORTED
00125  C2C5 38                 SEC
00126  C2C6 AD 50 03           LDA XMAX
00127  C2C9 ED 4D 03           SBC XMID
00128  C2CC AD 51 03           LDA XMAX+1
00129  C2CF ED 4E 03           SBC XMID+1
00130  C2D2 B0 DC              BCS SORTLP
00131  C2D4 20 9A C2           JSR SWAP23
00132  C2D7 4C B0 C2           JMP SORTLP
00133  C2DA 60          SORTED RTS
00134  C2DB             ;
00135  C2DB             ;*****************************************
00136  C2DB             ;
00137  C2DB             ; DRAW A LINE BETWEEN XMIN,YMIN AND XMID,YMID
00138  C2DB             ; USING FAST DDA (DIGITAL DIFFERENTIAL ANALYZER)
00139  C2DB             ; TECHNIQUE
00140  C2DB             ;
00141  C2DB A9 02       LINE   LDA #2          ; ENSURE XMAX IS
00142  C2DD 8D 51 03           STA XMAX+1      ; LARGEST BEFORE
00143  C2E0 20 AE C2           JSR SORTX       ; ORDERING 'MIN' AND 'MID'
00144  C2E3 AD 47 03           LDA NOSCAL
00145  C2E6 F0 03              BEQ OUTLN
00146  C2E8 20 6F C2           JSR SCALE
00147  C2EB 20 6F C4    OUTLN  JSR FINDXY      ; ENTRY POINT TO
00148  C2EE AD 4A 03           LDA XMIN        ; OUTLINE FACETS
00149  C2F1 8D 3F 03           STA XPLT
00150  C2F4 AD 4B 03           LDA XMIN+1
00151  C2F7 8D 40 03           STA XPLT+1
00152  C2FA AD 4C 03           LDA YMIN
00153  C2FD 8D 41 03           STA YPLT
00154  C300 AD 57 03           LDA DLTAX1+1    ; CHECK FOR DX>DY
00155  C303 D0 7D              BNE STEPX
00156  C305 38                 SEC
00157  C306 AD 56 03           LDA DLTAX1
00158  C309 ED 5B 03           SBC DLTAY1
00159  C30C B0 74              BCS STEPX
00160  C30E AD 5B 03    STEPY  LDA DLTAY1
00161  C311 8D 65 03           STA ERROR
00162  C314 8D 68 03           STA COUNT
00163  C317 4E 65 03           LSR ERROR
00164  C31A 38                 SEC
00165  C31B AD 56 03           LDA DLTAX1
00166  C31E ED 65 03           SBC ERROR
00167  C321 8D 65 03           STA ERROR
00168  C324 AD 57 03           LDA DLTAX1+1
00169  C327 E9 00              SBC #0
00170  C329 8D 66 03           STA ERROR+1
00171  C32C EE 68 03           INC COUNT
00172  C32F AD 67 03    LNLP1  LDA MODE        ; 0 = DRAW, 1 = ERASE
00173  C332 D0 06              BNE ERASE1
00174  C334 20 43 C1           JSR PLOT
00175  C337 4C 3D C3           JMP SK1
00176  C33A 20 46 C1    ERASE1 JSR UNPLOT
00177  C33D AD 60 03    SK1    LDA FLAG1       ; 0 = POSITIVE SLOPE
00178  C340 D0 05              BNE NSLOPE
00179  C342 EE 41 03           INC YPLT
00180  C345 D0 03              BNE SK2         ; ALWAYS BRANCH
00181  C347 CE 41 03    NSLOPE DEC YPLT
00182  C34A 2C 66 03    SK2    BIT ERROR+1
00183  C34D 30 1A              BMI SK3
00184  C34F EE 3F 03           INC XPLT
00185  C352 D0 03              BNE NOINC1
00186  C354 EE 40 03           INC XPLT+1
00187  C357 38          NOINC1 SEC
00188  C358 AD 65 03           LDA ERROR
00189  C35B ED 5B 03           SBC DLTAY1
00190  C35E 8D 65 03           STA ERROR
00191  C361 AD 66 03           LDA ERROR+1
00192  C364 E9 00              SBC #0
00193  C366 8D 66 03           STA ERROR+1
00194  C369 18          SK3    CLC
00195  C36A AD 65 03           LDA ERROR
00196  C36D 6D 56 03           ADC DLTAX1
00197  C370 8D 65 03           STA ERROR
00198  C373 AD 66 03           LDA ERROR+1
00199  C376 6D 57 03           ADC DLTAX1+1
00200  C379 8D 66 03           STA ERROR+1
00201  C37C CE 68 03           DEC COUNT
00202  C37F D0 AE              BNE LNLP1
00203  C381 60                 RTS
00204  C382             ;
00205  C382 AD 56 03    STEPX  LDA DLTAX1
00206  C385 8D 65 03           STA ERROR
00207  C388 8D 68 03           STA COUNT
00208  C38B AD 57 03           LDA DLTAX1+1
00209  C38E 8D 66 03           STA ERROR+1
00210  C391 8D 69 03           STA COUNT+1
00211  C394 4E 66 03           LSR ERROR+1
00212  C397 6E 65 03           ROR ERROR
00213  C39A 38                 SEC
00214  C39B AD 5B 03           LDA DLTAY1
00215  C39E ED 65 03           SBC ERROR
00216  C3A1 8D 65 03           STA ERROR
00217  C3A4 A9 00              LDA #0
00218  C3A6 ED 66 03           SBC ERROR+1
00219  C3A9 8D 66 03           STA ERROR+1
00220  C3AC AD 67 03    LNLP2  LDA MODE
00221  C3AF D0 06              BNE ERASE2
00222  C3B1 20 43 C1           JSR PLOT
00223  C3B4 4C BA C3           JMP SKP1
00224  C3B7 20 46 C1    ERASE2 JSR UNPLOT
00225  C3BA EE 3F 03    SKP1   INC XPLT
00226  C3BD D0 03              BNE NOINC2
00227  C3BF EE 40 03           INC XPLT+1
00228  C3C2 2C 66 03    NOINC2 BIT ERROR+1
00229  C3C5 30 20              BMI SKP3
00230  C3C7 AD 60 03           LDA FLAG1
00231  C3CA D0 05              BNE NGSLP
00232  C3CC EE 41 03           INC YPLT
00233  C3CF D0 03              BNE SKP2        ; ALWAYS BRANCH
00234  C3D1 CE 41 03    NGSLP  DEC YPLT
00235  C3D4 38          SKP2   SEC
00236  C3D5 AD 65 03           LDA ERROR
00237  C3D8 ED 56 03           SBC DLTAX1
00238  C3DB 8D 65 03           STA ERROR
00239  C3DE AD 66 03           LDA ERROR+1
00240  C3E1 ED 57 03           SBC DLTAX1+1
00241  C3E4 8D 66 03           STA ERROR+1
00242  C3E7 18          SKP3   CLC
00243  C3E8 AD 65 03           LDA ERROR
00244  C3EB 6D 5B 03           ADC DLTAY1
00245  C3EE 8D 65 03           STA ERROR
00246  C3F1 AD 66 03           LDA ERROR+1
00247  C3F4 69 00              ADC #0
00248  C3F6 8D 66 03           STA ERROR+1
00249  C3F9 38                 SEC
00250  C3FA AD 68 03           LDA COUNT
00251  C3FD E9 01              SBC #1
00252  C3FF 8D 68 03           STA COUNT
00253  C402 B0 03              BCS TEST
00254  C404 CE 69 03           DEC COUNT+1
00255  C407 2C 69 03    TEST   BIT COUNT+1
00256  C40A 10 A0              BPL LNLP2
00257  C40C 60                 RTS
00258  C40D             ;
00259  C40D             ;*****************************************
00260  C40D             ; DRAW A SHADED VERTICAL LINE AT
00261  C40D             ; XPLT FROM YTOP TO YBOT
00262  C40D             ;
00263  C40D 38          VLINE  SEC             ; MAKE SURE YTOP>YBOT
00264  C40E AD 53 03           LDA YTOP
00265  C411 ED 54 03           SBC YBOT
00266  C414 B0 0E              BCS DRAW
00267  C416 AD 53 03           LDA YTOP
00268  C419 48                 PHA
00269  C41A AD 54 03           LDA YBOT
00270  C41D 8D 53 03           STA YTOP
00271  C420 68                 PLA
00272  C421 8D 54 03           STA YBOT
```

(Continued on page 68)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Three

```
00273  C424  AD 53 03   DRAW   LDA YTOP
00274  C427  8D 41 03          STA YPLT
00275  C42A  20 24 C2          JSR NORM        ; PLOT A SHADE-WEIGHTED
00276  C42D  AD 53 03          LDA YTOP        ; PIXEL CHECKING ONLY
00277  C430  CD 54 03          CMP YBOT        ; FOR SHADE STYLE
00278  C433  F0 06             BEQ DONE
00279  C435  CE 53 03          DEC YTOP
00280  C438  4C 24 C4          JMP DRAW
00281  C43B  60                DONE   RTS
00282  C43C                    ;
00283  C43C                    ;****************************************
00284  C43C                    ;
00285  C43C                    ; FIND ENDPOINTS FOR VERTICAL LINES
00286  C43C                    ; BETWEEN FACET EDGES
00287  C43C                    ;
00288  C43C  AD 5F 03          ENDPTS LDA XDIFF
00289  C43F  85 AC                    STA MLPCND
00290  C441  AD 5E 03                 LDA DELTAY
00291  C444  85 AD                    STA MLPLER
00292  C446  20 11 C0                 JSR MULT
00293  C449  85 FE                    STA DVDND+1
00294  C44B  A5 AE                    LDA PROD
00295  C44D  85 FD                    STA DVDND
00296  C44F  A9 00                    LDA #0
00297  C451  85 FC                    STA DVSOR+1
00298  C453  AD 5A 03                 LDA DELTAX
00299  C456  85 FB                    STA DVSOR
00300  C458  20 25 C0                 JSR DIVIDE
00301  C45B  AD 63 03                 LDA FLAG
00302  C45E  D0 08                    BNE NEGSLP
00303  C460  18                       CLC
00304  C461  AD 55 03                 LDA YBASE
00305  C464  65 FD                    ADC QUOT
00306  C466  90 06                    BCC SKIP2
00307  C468  38                NEGSLP SEC
00308  C469  AD 55 03                 LDA YBASE
00309  C46C  E5 FD                    SBC QUOT
00310  C46E  60                SKIP2  RTS
00311  C46F                    ;
00312  C46F                    ;****************************************
00313  C46F                    ;
00314  C46F                    ; FIND COORDINATE DIFFERENCES
00315  C46F                    ;
00316  C46F                    ; ALL "DELTA X" VALUES POSITIVE,
00317  C46F                    ; SINGLE PRECISION (JUST LOWER BYTE)
00318  C46F                    ;
00319  C46F  38                FINDXY SEC
00320  C470  AD 4D 03                 LDA XMID
00321  C473  ED 4A 03                 SBC XMIN
00322  C476  8D 56 03                 STA DLTAX1
00323  C479  AD 4E 03                 LDA XMID+1
00324  C47C  ED 4B 03                 SBC XMIN+1
00325  C47F  8D 57 03                 STA DLTAX1+1
00326  C482  38                       SEC
00327  C483  AD 50 03                 LDA XMAX
00328  C486  ED 4D 03                 SBC XMID
00329  C489  8D 58 03                 STA DLTAX2
00330  C48C  38                       SEC
00331  C48D  AD 50 03                 LDA XMAX
00332  C490  ED 4A 03                 SBC XMIN
00333  C493  8D 59 03                 STA DLTAX3
00334  C496                    ;
00335  C496                    ; USE ABS(DELTA Y) VALUES,
00336  C496                    ; FLAGS INDICATE SLOPE OF LIMIT LINES
00337  C496                    ;
00338  C496  A9 00                    LDA #$00
00339  C498  8D 60 03                 STA FLAG1
00340  C49B  8D 61 03                 STA FLAG2
00341  C49E  8D 62 03                 STA FLAG3
00342  C4A1  38                       SEC
00343  C4A2  AD 4F 03                 LDA YMID
00344  C4A5  ED 4C 03                 SBC YMIN
00345  C4A8  B0 09                    BCS STORE1
00346  C4AA  EE 60 03                 INC FLAG1
00347  C4AD  AD 4C 03                 LDA YMIN
00348  C4B0  ED 4F 03                 SBC YMID
00349  C4B3  8D 5B 03          STORE1 STA DLTAY1
00350  C4B6  38                       SEC
00351  C4B7  AD 52 03                 LDA YMAX
00352  C4BA  ED 4F 03                 SBC YMID
00353  C4BD  B0 09                    BCS STORE2
00354  C4BF  EE 61 03                 INC FLAG2
00355  C4C2  AD 4F 03                 LDA YMID
00356  C4C5  ED 52 03                 SBC YMAX
00357  C4C8  8D 5C 03          STORE2 STA DLTAY2
00358  C4CB  38                       SEC
00359  C4CC  AD 52 03                 LDA YMAX
00360  C4CF  ED 4C 03                 SBC YMIN
00361  C4D2  B0 09                    BCS STORE3
00362  C4D4  EE 62 03                 INC FLAG3
00363  C4D7  AD 4C 03                 LDA YMIN
00364  C4DA  ED 52 03                 SBC YMAX
00365  C4DD  8D 5D 03          STORE3 STA DLTAY3
00366  C4E0  60                       RTS
00367  C4E1                    ;
00368  C4E1                    ;****************************************
00369  C4E1                    ;
00370  C4E1                    ; DRAW A SHADED TRIANGULAR FACET
00371  C4E1                    ;
00372  C4E1  20 AE C2          FACET  JSR SORTX
00373  C4E4  AD 47 03                 LDA NOSCAL
00374  C4E7  F0 03                    BEQ YSOK
00375  C4E9  20 6F C2                 JSR SCALE
00376  C4EC  20 6F C4          YSOK   JSR FINDXY
00377  C4EF  AD 4A 03                 LDA XMIN
00378  C4F2  8D 3F 03                 STA XPLT
00379  C4F5  AD 4B 03                 LDA XMIN+1
00380  C4F8  8D 40 03                 STA XPLT+1
00381  C4FB  38                FCETLP SEC
00382  C4FC  AD 3F 03                 LDA XPLT
00383  C4FF  ED 4A 03                 SBC XMIN
00384  C502  8D 5F 03                 STA XDIFF
00385  C505  AD 56 03                 LDA DLTAX1
00386  C508  F0 53                    BEQ CONT
00387  C50A  8D 5A 03                 STA DELTAX
00388  C50D  AD 5B 03                 LDA DLTAY1
00389  C510  8D 5E 03                 STA DELTAY
00390  C513  AD 60 03                 LDA FLAG1
00391  C516  8D 63 03                 STA FLAG
00392  C519  AD 4C 03                 LDA YMIN
00393  C51C  8D 55 03                 STA YBASE
00394  C51F  20 3C C4                 JSR ENDPTS
00395  C522  8D 53 03                 STA YTOP
00396  C525  AD 59 03                 LDA DLTAX3
00397  C528  F0 33                    BEQ CONT
00398  C52A  8D 5A 03                 STA DELTAX
00399  C52D  AD 5D 03                 LDA DLTAY3
00400  C530  8D 5E 03                 STA DELTAY
00401  C533  AD 62 03                 LDA FLAG3
00402  C536  8D 63 03                 STA FLAG
00403  C539  20 3C C4                 JSR ENDPTS
00404  C53C  8D 54 03                 STA YBOT
00405  C53F  20 0D C4                 JSR VLINE
00406  C542  AD 40 03                 LDA XPLT+1
00407  C545  CD 4E 03                 CMP XMID+1
00408  C548  D0 08                    BNE NEXTX1
00409  C54A  AD 3F 03                 LDA XPLT
00410  C54D  CD 4D 03                 CMP XMID
00411  C550  F0 0B                    BEQ CONT
00412  C552  EE 3F 03          NEXTX1 INC XPLT
00413  C555  D0 03                    BNE SKIP3
00414  C557  EE 40 03                 INC XPLT+1
00415  C55A  4C FB C4          SKIP3  JMP FCETLP
00416  C55D  38                CONT   SEC
00417  C55E  AD 3F 03                 LDA XPLT
00418  C561  ED 4A 03                 SBC XMIN
00419  C564  8D 5F 03                 STA XDIFF
00420  C567  AD 59 03                 LDA DLTAX3
00421  C56A  F0 63                    BEQ FINI
00422  C56C  8D 5A 03                 STA DELTAX
00423  C56F  AD 5D 03                 LDA DLTAY3
00424  C572  8D 5E 03                 STA DELTAY
00425  C575  AD 62 03                 LDA FLAG3
00426  C578  8D 63 03                 STA FLAG
00427  C57B  AD 4C 03                 LDA YMIN
00428  C57E  8D 55 03                 STA YBASE
00429  C581  20 3C C4                 JSR ENDPTS
00430  C584  8D 54 03                 STA YBOT
00431  C587  38                       SEC
00432  C588  AD 3F 03                 LDA XPLT
00433  C58B  ED 4D 03                 SBC XMID
00434  C58E  8D 5F 03                 STA XDIFF
00435  C591  AD 58 03                 LDA DLTAX2
00436  C594  F0 39                    BEQ FINI
00437  C596  8D 5A 03                 STA DELTAX
00438  C599  AD 5C 03                 LDA DLTAY2
00439  C59C  8D 5E 03                 STA DELTAY
00440  C59F  AD 61 03                 LDA FLAG2
00441  C5A2  8D 63 03                 STA FLAG
00442  C5A5  AD 4F 03                 LDA YMID
00443  C5A8  8D 55 03                 STA YBASE
00444  C5AB  20 3C C4                 JSR ENDPTS
00445  C5AE  8D 53 03                 STA YTOP
00446  C5B1  20 0D C4                 JSR VLINE
00447  C5B4  AD 40 03                 LDA XPLT+1
00448  C5B7  CD 51 03                 CMP XMAX+1
00449  C5BA  D0 08                    BNE NEXTX2
00450  C5BC  AD 3F 03                 LDA XPLT
00451  C5BF  CD 50 03                 CMP XMAX
00452  C5C2  F0 0B                    BEQ FINI
00453  C5C4  EE 3F 03          NEXTX2 INC XPLT
00454  C5C7  D0 03                    BNE SKIP4
00455  C5C9  EE 40 03                 INC XPLT+1
00456  C5CC  4C 5D C5          SKIP4  JMP CONT
00457  C5CF  AD 64 03          FINI   LDA EDGES
00458  C5D2  F0 15                    BEQ FINISH
00459  C5D4  20 EB C2                 JSR OUTLN
00460  C5D7  20 9A C2                 JSR SWAP23
00461  C5DA  20 EB C2                 JSR OUTLN
00462  C5DD  20 B6 C2                 JSR SWAP12
00463  C5E0  20 9A C2                 JSR SWAP23
00464  C5E3  20 B6 C2                 JSR SWAP12
00465  C5E6  20 EB C2                 JSR OUTLN
00466  C5E9  60                FINISH RTS
00467  C5EA                    .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 CONT    C55D   COUNT   0368   DELTAX  035A   DELTAY  035E
 DIVIDE  C025   DLTAX1  0356   DLTAX2  0358   DLTAX3  0359
 DLTAY1  035B   DLTAY2  035C   DLTAY3  035D   DONE    C43B
 DRAW    C424   DVDND   00FD   DVSOR   00FB   EDGES   0364
 ENDPTS  C43C   ERASE1  C33A   ERASE2  C3B7   ERROR   0365
 FACET   C4E1   FCETLP  C4FB   FINDXY  C46F   FINI    C5CF
 FINISH  C5E9   FLAG    0363   FLAG1   0360   FLAG2   0361
 FLAG3   0362   LINE    C2DB   LNLP1   C32F   LNLP2   C3AC
 LOOP1   C288   LOOP2   C29C   MLPCND  00AC   MLPLER  00AD
 MODE    0367   MULT    C011   NEGSLP  C468   NEXTX1  C552
 NEXTX2  C5C4   NGSLP   C3D1   NOINC1  C357   NOINC2  C3C2
 NORM    C224   NOSCAL  0347   NOSWP1  C2C2   NSLOPE  C347
 ORIGIN  C26F   OUTLN   C2EB   PLOT    C143   PROD    00AE
 QUOT    00FD   RAM     034A   SCALE   C26F   SCLP    C275
 SK1     C33D   SK2     C34A   SK3     C369   SKIP2   C46E
 SKIP3   C55A   SKIP4   C5CC   SKP1    C38A   SKP2    C3D4
 SKP3    C3E7   SORTED  C2DA   SORTLP  C2B0   SORTX   C2AE
 STEPX   C382   STEPY   C30E   STORE1  C4B3   STORE2  C4C8
 STORE3  C4DD   SWAP12  C286   SWAP23  C29A   TEST    C407

 SYMBOL TABLE

 SYMBOL VALUE
  UNPLOT   C146   VLINE   C40D   XDIFF   035F   XMAX   0350
  XMID     034D   XMIN    034A   XPLT    033F   YBASE  0355
  YBOT     0354   YMAX    0352   YMID    034F   YMIN   034C
  YPLT     0341   YSOK    C4EC   YTOP    0353

 END OF ASSEMBLY
```

End Listing Three

Listing Four

```
00001  0000                    ; PRIMITIVE SOLID SHAPE DRAWING
00002  0000                    ;
00003  0000                    ; RICHARD L. RYLANDER   11/7/84
00004  0000                    ;
00005  0000                    ; LOAD ARITHMETIC AND GRAPHIC UTILITIES FIRST
00006  0000                    ;
00007  0000                    ;****************************************
00008  0000                    RAM=$036A
00009  0000                    ORIGIN=$C5EA
00010  0000                    ;
00011  0000                    MLPCND=$AC              ; MULTIPLICAND (S)
00012  0000                    MLPLER=$AD              ; MULTIPLIER (S)
00013  0000                    PROD=$AE                ; PRODUCT (D)
00014  0000                    MULT=$C011              ; CALL FOR MULTIPLY
00015  0000                    ;
00016  0000                    DVDND=$FD               ; DIVIDEND (D)
```

(Continued on page 70)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Four

```
00017  0000                    DVSOR=$FB               ; DIVISOR (D)
00018  0000                    DUOT=$FD                ; QUOTIENT (D)
00019  0000                    DIVIDE=$C025            ; CALL FOR DIVIDE
00020  0000                    ;
00021  0000                    ARG=$AC                 ; ARGUMENT (S)
00022  0000                    SQR=$AE                 ; SQUARE OF ARG (D)
00023  0000                    SQUARE=$C004            ; CALL FOR SQUARE
00024  0000                    ;
00025  0000                    RADCND=$AC              ; RADICAND (D)
00026  0000                    ROOT=$033C              ; SQUARE ROOT (S)
00027  0000                    SQRT=$C064              ; CALL FOR SQRT
00028  0000                    ;
00029  0000                    RNDM=$C000              ; RANDOM NUMBER
00030  0000                    RANDOM=$C0C8            ; CALL FOR RANDOM
00031  0000                    ; NOTE - A CALL TO 'RANDOM' LEAVES A RANDOM BYTE
00032  0000                    ; IN THE ACCUMULATOR
00033  0000                    ;
00034  0000                    XPLT=$033F
00035  0000                    YPLT=$0341
00036  0000                    NORM=$C224
00037  0000                    PLTSHD=$C20F
00038  0000                    VALUE=$0344             ; FINAL NORMALIZED SHADE VALUE
00039  0000                    HTORRN=$0346            ; SHADE FLAG, 1=HALFTONE
00040  0000                    NOSCAL=$0347            ; SCALE FLAG, 1=NO SCALE
00041  0000                    ;
00042  0000                            *=RAM
00043  036A                    XCENT   *=*+2           ; CENTER COORD
00044  036C                    XREL    *=*+1           ; RELATIVE (TO CENTER)
00045  036D                    XSHD    *=*+2           ; USED IN SHADE CALC
00046  036F                    YCENT   *=*+1           ; CENTER COORD
00047  0370                    YREL    *=*+1           ; RELATIVE (TO CENTER)
00048  0371                    YSHD    *=*+2           ; USED IN SHADE CALC
00049  0373                    ZREL    *=*+2           ; RELATIVE (TO CENTER)
00050  0375                    ZWX     *=*+2           ; Z WITH X (+ OR -)
00051  0377                    ;
00052  0377                    RADIUS  *=*+2           ; LOCAL RADIUS OF SURFACE
00053  0379                    TONE    *=*+2           ; USED IN SHADE CALC
00054  037B                    TNTMP   *=*+2           ; USED IN SHADE CALC
00055  037D                    ;
00056  037D                    CLIPL   *=*+1           ; LEFT CLIPPING BOUND
00057  037E                    CLIPR   *=*+1           ; RIGHT CLIPPING BOUND
00058  037F                    CLIPU   *=*+1           ; UP CLIPPING BOUND
00059  0380                    CLIPD   *=*+1           ; DOWN CLIPPING BOUND
00060  0381                    ;
00061  0381                    HEMI    *=*+1           ; PLOTTING HEMISPHERE
00062  0382                    ;
00063  0382                    BAKLIT  *=*+1           ; BACKLIT FLAG
00064  0383                    HVFLAG  *=*+1           ; HORIZONTAL/VERTICAL FLAG
00065  0384                    TEMP    *=*+2           ; TEMPORARY STORAGE
00066  0386                    CNTX    *=*+1           ; LOOP COUNTER
00067  0387                    CNTY    *=*+1           ; LOOP COUNTER
00068  0388                    MAX     *=*+1           ; LOOP LIMIT
00069  0389                    ;
00070  0389                    HLEN    *=*+1           ; HALF LENGTH OF CYLINDERS
00071  038A                    RS      *=*+2           ; SQUARE OF TOROID RADIUS
00072  038C                    RT      *=*+1           ; TOROID (RING) RADIUS
00073  038D                    RC      *=*+1           ; CENTER RADIUS OF TOROID
00074  038E                    RO      *=*+1           ; OUTER RADIUS OF TOROID
00075  038F                    RI      *=*+1           ; INNER RADIUS OF TOROID
00076  0390                    XSQR    *=*+2
00077  0392                    XMAX    *=*+1
00078  0393                    ;
00079  0393                    R0=HLEN
00080  0393                    ;
00081  0393                            *=ORIGIN
00082  C5EA                    ;***************************************
00083  C5EA                    ;
00084  C5EA                    ; DIVIDE WITH SINGLE PRECISION DIVISOR
00085  C5EA                    ; (USED OFTEN IN SHAPE ROUTINES)
00086  C5EA                    ;
00087  C5EA  A9 00             SDIV    LDA #0
00088  C5EC  85 FC                     STA DVSOR+1
00089  C5EE  4C 25 C0                  JMP DIVIDE
00090  C5F1                    ;
00091  C5F1                    ;***************************************
00092  C5F1                    ;
00093  C5F1                    ; CALCULATE SHADE VALUE (0-63) BY
00094  C5F1                    ; MULTIPLYING 'TONE' BY 26 THEN
00095  C5F1                    ; DIVIDE RESULT BY RADIUS OF SURFACE
00096  C5F1                    ;
00097  C5F1  2C 7A 03          GETVAL  BIT TONE+1
00098  C5F4  10 12                     BPL CNTNU       ; IF 'TONE'<0, THEN
00099  C5F6  AD 82 03                  LDA BAKLIT      ; MAKE VALUE 0 OR ABS(TONE)
00100  C5F9  D0 04                     BNE NEGATE      ; DEPENDING ON BAKLIT FLAG
00101  C5FB  8D 44 03                  STA VALUE
00102  C5FE  60                       RTS
00103  C5FF  38                NEGATE SEC
00104  C600  A9 00                     LDA #0
00105  C602  ED 79 03                  SBC TONE
00106  C605  8D 79 03                  STA TONE
00107  C608  AD 79 03          CNTNU   LDA TONE
00108  C60B  85 AC                     STA MLPCND
00109  C60D  A9 1A                     LDA #$1A
00110  C60F  85 AD                     STA MLPLER
00111  C611  20 11 C0                  JSR MULT
00112  C614  85 FE                     STA DVDND+1
00113  C616  A5 AE                     LDA PROD
00114  C618  85 FD                     STA DVDND
00115  C61A  AD 77 03                  LDA RADIUS
00116  C61D  85 FB                     STA DVSOR
00117  C61F  20 EA C5                  JSR SDIV
00118  C622  A5 FD                     LDA DUOT
00119  C624  8D 44 03                  STA VALUE
00120  C627  60                       RTS
00121  C628                    ;
00122  C628                    ;***************************************
00123  C628                    ;
00124  C628                    ; POINT PLOTTING BY QUADRANTS USING
00125  C628                    ; THE FOUR-FOLD SYMMETRY OF SIMPLE OBJECTS
00126  C628                    ;
00127  C628                    ; DEPENDING ON STATUS OF 'HVFLAG', EXCHANGE
00128  C628                    ; X AND Y COORDINATES TO ROTATE OBJECTS 90 DEG
00129  C628                    ; SINGLE SHAPE ROUTINE CAN THEN BE USED TO
00130  C628                    ; DRAW 'HORIZONTAL' OR 'VERTICAL' VERSIONS
00131  C628                    ; OF AN OBJECT
00132  C628                    ;
00133  C628                    ; THE FOLOWING IS A 'BASIC SUBROUTINE'
00134  C628                    ; EQUIVALENT TO EXPLAIN ITS OPERATION
00135  C628                    ;
00136  C628                    ; NOTE THAT LABELS ARE USED IN PLACE OF
00137  C628                    ; LINE NUMBERS
00138  C628                    ;
00139  C628                    ; 'PTPLOT' IF HVFLAG<0 THEN GOTO 'NOROT'
00140  C628                    ;          (STACK)=XREL:XREL=YREL:YREL=(STACK)
00141  C628                    ;          (STACK)=XSHD:XSHD=YSHD:YSHD=(STACK)
00142  C628                    ; 'NOROT'  GOSUB 'GETZ'
00143  C628                    ;          REM CALCULATE 2*Z FROM X,Y AND RADIUS
00144  C628                    ;          HEMI = 1
00145  C628                    ;          IF XREL>CLIPL THEN GOTO 'RHEMI'
00146  C628                    ;          ZWX=2*Z-XSHD
00147  C628                    ;  '       XPLT=XCENT-XREL:REM LEFT HEMISPHERE
00148  C628                    ; 'CHCLUP' IF YREL>CLIPU THEN GOTO 'DHEMI'
00149  C628                    ;          TONE=ZWX+YSHD
00150  C628                    ;          GOSUB 'GETVAL':REM NORMALIZE SHADE VAL
00151  C628                    ;          YPLT=YCENT+YREL
00152  C628                    ;          GOSUB 'PLTSHD':REM PLOT OR UNPLOT
00153  C628                    ;          REM   POINTS WEIGHTED BY SHADE VALUE
00154  C628                    ; 'DHEMI'  IF YREL>CLIPD THEN GOTO 'RHEMI'
00155  C628                    ;          TONE=ZWX-YSHD
00156  C628                    ;          GOSUB 'GETVAL'
00157  C628                    ;          YPLT=YCENT-YREL
00158  C628                    ;          GOSUB 'PLTSHD'
00159  C628                    ; 'RHEMI'  IF HEMI=0 THEN RETURN
00160  C628                    ;          HEMI=0
00161  C628                    ;          IF XREL>CLIPR THEN RETURN
00162  C628                    ;          ZWX=2*Z+XSHD
00163  C628                    ;          XPLT=XCENT+XREL
00164  C628                    ;          GOSUB 'CHCLUP'
00165  C628                    ;          RETURN
00166  C628                    ;
00167  C628  2C 83 03          PTPLOT BIT HVFLAG
00168  C62B  10 2D                     BPL NOROT
00169  C62D  AD 6C 03                  LDA XREL
00170  C630  48                       PHA
00171  C631  48                       PHA
00172  C632  AD 70 03                  LDA YREL
00173  C635  8D 6C 03                  STA XREL
00174  C638  68                       PLA
00175  C639  8D 70 03                  STA YREL
00176  C63C  AD 6D 03                  LDA XSHD
00177  C63F  48                       PHA
00178  C640  48                       PHA
00179  C641  AD 71 03                  LDA YSHD
00180  C644  8D 6D 03                  STA XSHD
00181  C647  68                       PLA
00182  C648  8D 71 03                  STA YSHD
00183  C64B  AD 6E 03                  LDA XSHD+1
00184  C64E  48                       PHA
00185  C64F  48                       PHA
00186  C650  AD 72 03                  LDA YSHD+1
00187  C653  8D 6E 03                  STA XSHD+1
00188  C656  68                       PLA
00189  C657  8D 72 03                  STA YSHD+1
00190  C65A  20 45 C7          NOROT   JSR GETZ
00191  C65D  A9 01             PTPLT2  LDA #$01
00192  C65F  8D 81 03                  STA HEMI
00193  C662  38                       SEC
00194  C663  AD 7D 03                  LDA CLIPL       ; CHECK LEFT HEMISPHERE
00195  C666  CD 6C 03                  CMP XREL
00196  C669  90 7D                     BCC RHEMI
00197  C66B  38                       SEC
00198  C66C  AD 3C 03                  LDA ROOT
00199  C66F  ED 6D 03                  SBC XSHD
00200  C672  8D 75 03                  STA ZWX
00201  C675  AD 3D 03                  LDA ROOT+1
00202  C678  ED 6E 03                  SBC XSHD+1
00203  C67B  8D 76 03                  STA ZWX+1
00204  C67E  38                       SEC
00205  C67F  AD 6A 03                  LDA XCENT
00206  C682  ED 6C 03                  SBC XREL
00207  C685  8D 3F 03                  STA XPLT
00208  C688  AD 6B 03                  LDA XCENT+1
00209  C68B  E9 00                     SBC #$00
00210  C68D  8D 40 03                  STA XPLT+1
00211  C690                    ;
00212  C690  38                CHCLUP  SEC
00213  C691  AD 7F 03                  LDA CLIPU       ; CHECK FOR UP CLIPPING
00214  C694  CD 70 03                  CMP YREL
00215  C697  90 23                     BCC DHEMI
00216  C699  18                       CLC
00217  C69A  AD 75 03                  LDA ZWX
00218  C69D  6D 71 03                  ADC YSHD
00219  C6A0  8D 79 03                  STA TONE
00220  C6A3  AD 76 03                  LDA ZWX+1
00221  C6A6  6D 72 03                  ADC YSHD+1
00222  C6A9  8D 7A 03                  STA TONE+1
00223  C6AC  20 F1 C5                  JSR GETVAL
00224  C6AF  18                       CLC
00225  C6B0  AD 6F 03                  LDA YCENT
00226  C6B3  6D 70 03                  ADC YREL
00227  C6B6  8D 41 03                  STA YPLT
00228  C6B9  20 0F C2                  JSR PLTSHD
00229  C6BC                    ;
00230  C6BC  38                DHEMI   SEC
00231  C6BD  AD 80 03                  LDA CLIPD       ; CHECK FOR DOWN CLIPPING
00232  C6C0  CD 70 03                  CMP YREL
00233  C6C3  90 23                     BCC RHEMI
00234  C6C5  38                       SEC
00235  C6C6  AD 75 03                  LDA ZWX
00236  C6C9  ED 71 03                  SBC YSHD
00237  C6CC  8D 79 03                  STA TONE
00238  C6CF  AD 76 03                  LDA ZWX+1
00239  C6D2  ED 72 03                  SBC YSHD+1
00240  C6D5  8D 7A 03                  STA TONE+1
00241  C6D8  20 F1 C5                  JSR GETVAL
00242  C6DB  38                       SEC
00243  C6DC  AD 6F 03                  LDA YCENT
00244  C6DF  ED 70 03                  SBC YREL
00245  C6E2  8D 41 03                  STA YPLT
00246  C6E5  20 0F C2                  JSR PLTSHD
00247  C6E8                    ;
00248  C6E8  AD 81 03          RHEMI   LDA HEMI
00249  C6EB  F0 34                     BEQ PLDONE
00250  C6ED  CE 81 03                  DEC HEMI
00251  C6F0  38                       SEC
00252  C6F1  AD 7E 03                  LDA CLIPR       ; CHECK FOR RIGHT CLIPPING
00253  C6F4  CD 6C 03                  CMP XREL
00254  C6F7  90 28                     BCC PLDONE
00255  C6F9  18                       CLC
00256  C6FA  AD 3C 03                  LDA ROOT
00257  C6FD  6D 6D 03                  ADC XSHD
00258  C700  8D 75 03                  STA ZWX
00259  C703  AD 3D 03                  LDA ROOT+1
00260  C706  6D 6E 03                  ADC XSHD+1
00261  C709  8D 76 03                  STA ZWX+1
00262  C70C  18                       CLC
00263  C70D  AD 6A 03                  LDA XCENT
00264  C710  6D 6C 03                  ADC XREL
00265  C713  8D 3F 03                  STA XPLT
00266  C716  AD 6B 03                  LDA XCENT+1
00267  C719  69 00                     ADC #$00
00268  C71B  8D 40 03                  STA XPLT+1
00269  C71E  4C 90 C6                  JMP CHCLUP
00270  C721  2C 83 03          PLDONE  BIT HVFLAG
00271  C724  10 1E                     BPL NORSTR
```

(Continued on page 72)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Four

```
00272  C726  AD 6E 03          LDA XSHD+1      ; RESTORE COORDS
00273  C729  8D 72 03          STA YSHD+1
00274  C72C  68                PLA
00275  C72D  8D 6E 03          STA XSHD+1
00276  C730  AD 6D 03          LDA XSHD
00277  C733  8D 71 03          STA YSHD
00278  C736  68                PLA
00279  C737  8D 6D 03          STA XSHD
00280  C73A  AD 6C 03          LDA XREL
00281  C73D  8D 70 03          STA YREL
00282  C740  68                PLA
00283  C741  8D 6C 03          STA XREL
00284  C744  60          NORSTR RTS
00285  C745                    ;
00286  C745                    ;****************************************
00287  C745                    ;
00288  C745                    ; CALCULATE Z FROM LOCAL X,Y BY
00289  C745                    ; PYTHAGOREAN SUM
00290  C745                    ;
00291  C745  AD 77 03    GETZ  LDA RADIUS
00292  C748  85 AC             STA ARG
00293  C74A  20 04 C0          JSR SQUARE
00294  C74D  8D 7C 03          STA TNTMP+1
00295  C750  A5 AE             LDA SQR
00296  C752  8D 7B 03          STA TNTMP
00297  C755  AD 6D 03          LDA XSHD
00298  C758  85 AC             STA ARG
00299  C75A  20 04 C0          JSR SQUARE
00300  C75D  38                SEC
00301  C75E  AD 7B 03          LDA TNTMP
00302  C761  E5 AE             SBC SQR
00303  C763  8D 7B 03          STA TNTMP
00304  C766  AD 7C 03          LDA TNTMP+1
00305  C769  E5 AF             SBC SQR+1
00306  C76B  8D 7C 03          STA TNTMP+1
00307  C76E  AD 71 03          LDA YSHD
00308  C771  85 AC             STA ARG
00309  C773  20 04 C0          JSR SQUARE
00310  C776  38                SEC
00311  C777  AD 7B 03          LDA TNTMP
00312  C77A  E5 AE             SBC SQR
00313  C77C  85 AC             STA RADCND
00314  C77E  AD 7C 03          LDA TNTMP+1
00315  C781  E5 AF             SBC SQR+1
00316  C783  85 AD             STA RADCND+1
00317  C785  30 0A             BMI ZEROOT
00318  C787  20 64 C0          JSR SQRT
00319  C78A  0E 3C 03          ASL ROOT
00320  C78D  2E 3D 03          ROL ROOT+1
00321  C790  60                RTS
00322  C791  A9 00       ZEROOT LDA #$00
00323  C793  8D 3C 03          STA ROOT
00324  C796  8D 3D 03          STA ROOT+1
00325  C799  60                RTS
00326  C79A                    ;
00327  C79A                    ;****************************************
00328  C79A                    ;
00329  C79A                    ; SET UP PARAMETERS FOR TOROIDS
00330  C79A                    ;
00331  C79A                    ; RT=(RO-RI)/2   RS=RT*RT   RC=RT+RI
00332  C79A                    ;
00333  C79A  AD BE 03    TPARM LDA RO
00334  C79D  38                SEC
00335  C79E  ED BF 03          SBC RI
00336  C7A1  4A                LSR A
00337  C7A2  8D BC 03          STA RT
00338  C7A5  8D 77 03          STA RADIUS
00339  C7A8  18                CLC
00340  C7A9  6D BF 03          ADC RI
00341  C7AC  8D BD 03          STA RC
00342  C7AF  AD BC 03          LDA RT
00343  C7B2  85 AC             STA ARG
00344  C7B4  20 04 C0          JSR SQUARE
00345  C7B7  A5 AE             LDA SQR
00346  C7B9  8D BA 03          STA RS
00347  C7BC  A5 AF             LDA SQR+1
00348  C7BE  8D BB 03          STA RS+1
00349  C7C1  A9 00             LDA #0
00350  C7C3  8D B6 03          STA CNTX
00351  C7C6  60                RTS
00352  C7C7                    ;
00353  C7C7                    ;****************************************
00354  C7C7                    ;
00355  C7C7                    ; DRAW A SHADED SPHERE
00356  C7C7                    ;
00357  C7C7                    ; 'BASIC SUBROUTINE' EQUIVALENT
00358  C7C7                    ;
00359  C7C7                    ; 'SPHERE' FOR CNTX=0 TO RADIUS/SQR(2)
00360  C7C7                    ;          XREL=CNTX:XSHD=CNTX
00361  C7C7                    ;          FOR CNTY=CNTX TO SQR(RAD*RAD-CNTX*CNTX)
00362  C7C7                    ;          YREL=CNTY:YSHD=CNTY
00363  C7C7                    ;          HVFLAG=0
00364  C7C7                    ;          GOSUB 'PTPLOT'
00365  C7C7                    ;          REM EXCHANGE X & Y TO USE 8-FOLD SYM
00366  C7C7                    ;          HVFLAG=-128
00367  C7C7                    ;
00368  C7C7                    ;          GOSUB 'PTPLOT'
00369  C7C7                    ;          NEXT CNTY
00370  C7C7                    ;          NEXT CNTX
00371  C7C7                    ;          RETURN
00372  C7C7                    ;
00373  C7C7                    ;
00374  C7C7  AD 77 03    SPHERE LDA RADIUS
00375  C7CA  85 AC             STA ARG
00376  C7CC  20 04 C0          JSR SQUARE
00377  C7CF  06 AE             ASL SQR
00378  C7D1  26 AF             ROL SQR+1
00379  C7D3  A5 AE             LDA SQR
00380  C7D5  85 AC             STA RADCND
00381  C7D7  A5 AF             LDA SQR+1
00382  C7D9  85 AD             STA RADCND+1
00383  C7DB  20 64 C0          JSR SQRT
00384  C7DE  4E 3D 03          LSR ROOT+1
00385  C7E1  6E 3C 03          ROR ROOT
00386  C7E4  AD 3C 03          LDA ROOT
00387  C7E7  8D 92 03          STA XMAX
00388  C7EA  A9 00             LDA #$00
00389  C7EC  8D B6 03          STA CNTX
00390  C7EF  8D 6E 03          STA XSHD+1
00391  C7F2  8D 72 03          STA YSHD+1
00392  C7F5  AD 77 03          LDA RADIUS
00393  C7F8  85 AC             STA ARG
00394  C7FA  20 04 C0          JSR SQUARE
00395  C7FD  8D B5 03          STA TEMP+1
00396  C800  A5 AE             LDA SQR
00397  C802  8D B4 03          STA TEMP
00398  C805  AD B6 03    LOOPX LDA CNTX
00399  C808  8D B7 03          STA CNTY
00400  C80B  85 AC             STA ARG
00401  C80D  8D 6C 03          STA XREL
00402  C810  8D 6D 03          STA XSHD
00403  C813  20 04 C0          JSR SQUARE
00404  C816  38                SEC
00405  C817  AD B4 03          LDA TEMP
00406  C81A  E5 AE             SBC SQR
00407  C81C  85 AC             STA RADCND
00408  C81E  AD B5 03          LDA TEMP+1
00409  C821  E5 AF             SBC SQR+1
00410  C823  85 AD             STA RADCND+1
00411  C825  20 64 C0          JSR SQRT
00412  C828  AD 3C 03          LDA ROOT
00413  C82B  8D B8 03          STA MAX
00414  C82E  AD B7 03    LOOPY LDA CNTY
00415  C831  8D 70 03          STA YREL
00416  C834  8D 71 03          STA YSHD
00417  C837  A9 00             LDA #0
00418  C839  8D B3 03          STA HVFLAG
00419  C83C  20 2B C6          JSR PTPLOT
00420  C83F  A9 80             LDA #$80
00421  C841  8D B3 03          STA HVFLAG
00422  C844  20 2B C6          JSR PTPLOT
00423  C847  AD B7 03          LDA CNTY
00424  C84A  CD B8 03          CMP MAX
00425  C84D  F0 06             BEQ DONEY
00426  C84F  EE B7 03          INC CNTY
00427  C852  4C 2E C8          JMP LOOPY
00428  C855  AD B6 03    DONEY LDA CNTX
00429  C858  CD 92 03          CMP XMAX
00430  C85B  F0 06             BEQ DONE
00431  C85D  EE B6 03          INC CNTX
00432  C860  4C 05 C8          JMP LOOPX
00433  C863  60          DONE  RTS
00434  C864                    ;
00435  C864                    ;****************************************
00436  C864                    ;
00437  C864                    ; DRAW SHADED CYLINDERS
00438  C864                    ;
00439  C864                    ; 'BASIC SUBROUTINE' EQUIVALENT
00440  C864                    ;
00441  C864                    ; 'CYLNDR' XSHD=0
00442  C864                    ;          FOR YREL=RADIUS TO 0
00443  C864                    ;          YSHD=YREL
00444  C864                    ;          FOR XREL=HLEN TO 0
00445  C864                    ;          GOSUB 'PTPLOT'
00446  C864                    ;          NEXT XREL
00447  C864                    ;          NEXT YREL
00448  C864                    ;          RETURN
00449  C864                    ;
00450  C864  A9 00       CYLNDR LDA #0
00451  C866  8D 6D 03          STA XSHD
00452  C869  8D 6E 03          STA XSHD+1
00453  C86C  8D 72 03          STA YSHD+1
00454  C86F  AD 77 03          LDA RADIUS
00455  C872  8D 70 03          STA YREL
00456  C875  AD B9 03    CYLOOP LDA HLEN
00457  C878  8D 6C 03          STA XREL
00458  C87B  AD 70 03          LDA YREL
00459  C87E  8D 71 03          STA YSHD
00460  C881  20 2B C6    CXLOOP JSR PTPLOT
00461  C884  CE 6C 03          DEC XREL
00462  C887  10 F8             BPL CXLOOP
00463  C889  CE 70 03          DEC YREL
00464  C88C  10 E7             BPL CYLOOP
00465  C88E  60                RTS
00466  C88F                    ;
00467  C88F                    ;****************************************
00468  C88F                    ;
00469  C88F                    ; DRAW EDGE-VIEW TOROIDS
00470  C88F                    ;
00471  C88F                    ; 'BASIC SUBROUTINE' EQUIVALENT
00472  C88F                    ;
00473  C88F                    ; 'EDGTOR' GOSUB 'TPARM':REM SET UP RADII
00474  C88F                    ;          FOR CNTX=0 TO RT
00475  C88F                    ;          XREL=CNTX:XSHD=CNTX
00476  C88F                    ;          R0=SQR(RT*RT-CNTX*CNTX)
00477  C88F                    ;          FOR CNTY=0 TO R0+RC
00478  C88F                    ;          YREL=CNTY
00479  C88F                    ;          YSHD=(R0*CNTY)/(R0+RC)
00480  C88F                    ;          GOSUB 'PTPLOT'
00481  C88F                    ;          NEXT CNTY
00482  C88F                    ;          NEXT CNTX
00483  C88F                    ;          RETURN
00484  C88F                    ;
00485  C88F  20 9A C7    EDGTOR JSR TPARM
00486  C892  A9 00             LDA #$00
00487  C894  8D 6E 03          STA XSHD+1
00488  C897  8D 72 03          STA YSHD+1
00489  C89A  AD B6 03    LOOPX4 LDA CNTX
00490  C89D  8D 6C 03          STA XREL
00491  C8A0  8D 6D 03          STA XSHD
00492  C8A3  85 AC             STA ARG
00493  C8A5  20 04 C0          JSR SQUARE
00494  C8A8  38                SEC
00495  C8A9  AD BA 03          LDA RS
00496  C8AC  E5 AE             SBC SQR
00497  C8AE  85 AC             STA RADCND
00498  C8B0  AD BB 03          LDA RS+1
00499  C8B3  E5 AF             SBC SQR+1
00500  C8B5  85 AD             STA RADCND+1
00501  C8B7  20 64 C0          JSR SQRT
00502  C8BA  AD 3C 03          LDA ROOT
00503  C8BD  8D B9 03          STA R0
00504  C8C0  18                CLC
00505  C8C1  6D BD 03          ADC RC
00506  C8C4  8D B8 03          STA MAX
00507  C8C7  A9 00             LDA #$00
00508  C8C9  8D B7 03          STA CNTY
00509  C8CC  AD B7 03    LOOPY4 LDA CNTY
00510  C8CF  8D 70 03          STA YREL
00511  C8D2  85 AD             STA MLPLER
00512  C8D4  AD B9 03          LDA R0
00513  C8D7  85 AC             STA MLPCND
00514  C8D9  20 11 C0          JSR MULT
00515  C8DC  85 FE             STA DVDND+1
00516  C8DE  A5 AE             LDA PROD
00517  C8E0  85 FD             STA DVDND
00518  C8E2  AD B8 03          LDA MAX
00519  C8E5  85 FB             STA DVSOR
00520  C8E7  20 EA C5          JSR SDIV
00521  C8EA  A5 FD             LDA QUOT
00522  C8EC  8D 71 03          STA YSHD
00523  C8EF  20 2B C6          JSR PTPLOT
00524  C8F2  AD B7 03          LDA CNTY
00525  C8F5  CD B8 03          CMP MAX
00526  C8F8  F0 06             BEQ DONE4
00527  C8FA  EE B7 03          INC CNTY
```

(Continued on page 74)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Four

```
00528  C8FD  4C CC C8          JMP LOOPY4
00529  C900  AD 86 03  DONE4   LDA CNTX
00530  C903  CD 8C 03          CMP RT
00531  C906  F0 06             BEQ DONEHT
00532  C908  EE 86 03          INC CNTX
00533  C90B  4C 9A C8          JMP LOOPX4
00534  C90E  60          DONEHT RTS
00535  C90F                    ;
00536  C90F                    ;****************************************
00537  C90F                    ;
00538  C90F                    ; DRAW A SHADED, TOP-VIEW TOROID
00539  C90F                    ;
00540  C90F                    ; 'BASIC SUBROUTINE' EQUIVALENT
00541  C90F                    ;
00542  C90F                    ; 'TOROID' GOSUB 'TPARM'
00543  C90F                    ;          FOR CNTX=0 TO RO/SQR(2)
00544  C90F                    ;          REM 8-FOLD SYMMETRY USED
00545  C90F                    ;          XREL=CNTX
00546  C90F                    ;          MAX=SQR(RO*RO-CNTX*CNTX)
00547  C90F                    ;          IF CNTX>RI THEN GOTO 'GRTR'
00548  C90F                    ;          CNTY=SQR(RI*RI-CNTY*CNTY)
00549  C90F                    ;          GOTO 'LLPY1'
00550  C90F                    ; 'GRTR'   CNTY=CNTX
00551  C90F                    ; 'LLPY1'  YREL=CNTY
00552  C90F                    ;          R0=SQR(CNTY*CNTY+CNTX*CNTX)
00553  C90F                    ;          XSHD=CNTX-(CNTX*RC)/R0
00554  C90F                    ;          YSHD=CNTY-(CNTY*RC)/R0
00555  C90F                    ;          HVFLAG=0:GOSUB 'PTPLOT'
00556  C90F                    ;          HVFLAG=-128:GOSUB 'PTPLOT'
00557  C90F                    ;          IF CNTY=MAX THEN GOTO 'DDNY1'
00558  C90F                    ;          CNTY=CNTY+1
00559  C90F                    ;          GOTO 'LLPY1'
00560  C90F                    ; 'DDNY1'  NEXT CNTX
00561  C90F                    ;          RETURN
00562  C90F                    ;
00563  C90F  20 9A C7  TOROID JSR TPARM
00564  C912  AD 8E 03          LDA RO
00565  C915  85 AC             STA ARG
00566  C917  20 04 C0          JSR SQUARE
00567  C91A  06 AE             ASL SQR
00568  C91C  26 AF             ROL SQR+1
00569  C91E  A5 AE             LDA SQR
00570  C920  85 AC             STA RADCND
00571  C922  A5 AF             LDA SQR+1
00572  C924  85 AD             STA RADCND+1
00573  C926  20 64 C0          JSR SQRT
00574  C929  4E 3D 03          LSR ROOT+1
00575  C92C  6E 3C 03          ROR ROOT
00576  C92F  AD 3C 03          LDA ROOT
00577  C932  8D 92 03          STA XMAX
00578  C935  AD 86 03  LLPX1   LDA CNTX
00579  C938  8D 6C 03          STA XREL
00580  C93B  85 AC             STA ARG
00581  C93D  20 04 C0          JSR SQUARE
00582  C940  8D 91 03          STA XSQR+1
00583  C943  A5 AE             LDA SQR
00584  C945  8D 90 03          STA XSQR
00585  C948  AD 8E 03          LDA RO
00586  C94B  85 AC             STA ARG
00587  C94D  20 04 C0          JSR SQUARE
00588  C950  38                SEC
00589  C951  A5 AE             LDA SQR
00590  C953  ED 90 03          SBC XSQR
00591  C956  85 AC             STA RADCND
00592  C958  A5 AF             LDA SQR+1
00593  C95A  ED 91 03          SBC XSQR+1
00594  C95D  85 AD             STA RADCND+1
00595  C95F  20 64 C0          JSR SQRT
00596  C962  AD 3C 03          LDA ROOT
00597  C965  8D 8B 03          STA MAX
00598  C968  38                SEC
00599  C969  AD 8F 03          LDA RI
00600  C96C  ED 86 03          SBC CNTX
00601  C96F  90 23             BCC GRTR
00602  C971  AD 8F 03          LDA RI
00603  C974  85 AC             STA ARG
00604  C976  20 04 C0          JSR SQUARE
00605  C979  38                SEC
00606  C97A  A5 AE             LDA SQR
00607  C97C  ED 90 03          SBC XSQR
00608  C97F  85 AC             STA RADCND
00609  C981  A5 AF             LDA SQR+1
00610  C983  ED 91 03          SBC XSQR+1
00611  C986  85 AD             STA RADCND+1
00612  C988  20 64 C0          JSR SQRT
00613  C98B  AD 3C 03          LDA ROOT
00614  C98E  8D 87 03          STA CNTY
00615  C991  4C 9A C9          JMP LLPY1
00616  C994  AD 86 03  GRTR    LDA CNTX
00617  C997  8D 87 03          STA CNTY
00618  C99A  AD 87 03  LLPY1   LDA CNTY
00619  C99D  8D 70 03          STA YREL
00620  C9A0  85 AC             STA ARG
00621  C9A2  20 04 C0          JSR SQUARE
00622  C9A5  18                CLC
00623  C9A6  A5 AE             LDA SQR
00624  C9A8  6D 90 03          ADC XSQR
00625  C9AB  85 AC             STA RADCND
00626  C9AD  A5 AF             LDA SQR+1
00627  C9AF  6D 91 03          ADC XSQR+1
00628  C9B2  85 AD             STA RADCND+1
00629  C9B4  20 64 C0          JSR SQRT
00630  C9B7  AD 3C 03          LDA ROOT
00631  C9BA  8D B9 03          STA R0
00632  C9BD  85 FB             STA DVSOR
00633  C9BF  AD 86 03          LDA CNTX
00634  C9C2  85 AD             STA MLPLER
00635  C9C4  AD 8D 03          LDA RC
00636  C9C7  85 AC             STA MLPCND
00637  C9C9  20 11 C0          JSR MULT
00638  C9CC  85 FE             STA DVDND+1
00639  C9CE  A5 AE             LDA PROD
00640  C9D0  85 FD             STA DVDND
00641  C9D2  20 EA C5          JSR SDIV
00642  C9D5  38                SEC
00643  C9D6  AD 86 03          LDA CNTX
00644  C9D9  E5 FD             SBC QUOT
00645  C9DB  8D 6D 03          STA XSHD
00646  C9DE  A9 00             LDA #$00
00647  C9E0  E5 FE             SBC QUOT+1
00648  C9E2  8D 6E 03          STA XSHD+1
00649  C9E5  AD 87 03          LDA CNTY
00650  C9E8  85 AD             STA MLPLER
00651  C9EA  AD 8D 03          LDA RC
00652  C9ED  85 AC             STA MLPCND
00653  C9EF  20 11 C0          JSR MULT
00654  C9F2  85 FE             STA DVDND+1
00655  C9F4  A5 AE             LDA PROD
00656  C9F6  85 FD             STA DVDND
00657  C9F8  AD 89 03          LDA R0
00658  C9FB  85 FB             STA DVSOR
00659  C9FD  20 EA C5          JSR SDIV
00660  CA00  38                SEC
00661  CA01  AD 87 03          LDA CNTY
00662  CA04  E5 FD             SBC QUOT
00663  CA06  8D 71 03          STA YSHD
00664  CA09  A9 00             LDA #$00
00665  CA0B  8D 83 03          STA HVFLAG
00666  CA0E  E5 FE             SBC QUOT+1
00667  CA10  8D 72 03          STA YSHD+1
00668  CA13  20 28 C6          JSR PTPLOT
00669  CA16  A9 80             LDA #$80
00670  CA18  8D 83 03          STA HVFLAG
00671  CA1B  20 28 C6          JSR PTPLOT
00672  CA1E  AD 87 03          LDA CNTY
00673  CA21  CD 8B 03          CMP MAX
00674  CA24  F0 06             BEQ DDNY1
00675  CA26  EE 87 03          INC CNTY
00676  CA29  4C 9A C9          JMP LLPY1
00677  CA2C  AD 86 03  DDNY1   LDA CNTX
00678  CA2F  CD 92 03          CMP XMAX
00679  CA32  F0 06             BEQ DUNTOR
00680  CA34  EE 86 03          INC CNTX
00681  CA37  4C 35 C9          JMP LLPX1
00682  CA3A  60          DUNTOR RTS
00683  CA3B                    ;
00684  CA3B                    ;****************************************
00685  CA3B                    ;
00686  CA3B                    ; DRAW "INSIDE VIEW" TOROIDS
00687  CA3B                    ;
00688  CA3B                    ; 'BASIC SUBROUTINE' EQUIVALENT
00689  CA3B                    ;
00690  CA3B                    ; 'SPOOL'  GOSUB 'TPARM'
00691  CA3B                    ;          FOR CNTX=0 TO RT
00692  CA3B                    ;          XREL=CNTX:XSHD=CNTX
00693  CA3B                    ;          MAX=RC-SQR(RS-CNTX*CNTX)
00694  CA3B                    ;          FOR CNTY=0 TO MAX
00695  CA3B                    ;          YREL=CNTY
00696  CA3B                    ;          YSHD=(RC*CNTY/MAX)-CNTY
00697  CA3B                    ;          GOSUB 'PTPLOT'
00698  CA3B                    ;          NEXT CNTY
00699  CA3B                    ;          NEXT CNTX
00700  CA3B                    ;          RETURN
00701  CA3B                    ;
00702  CA3B  20 9A C7  SPOOL   JSR TPARM
00703  CA3E  AD 86 03  LLPX2   LDA CNTX
00704  CA41  8D 6C 03          STA XREL
00705  CA44  85 AC             STA ARG
00706  CA46  38                SEC
00707  CA47  A9 00             LDA #$00
00708  CA49  ED 86 03          SBC CNTX
00709  CA4C  8D 6D 03          STA XSHD
00710  CA4F  A9 00             LDA #$00
00711  CA51  E9 00             SBC #$00
00712  CA53  8D 6E 03          STA XSHD+1
00713  CA56  20 04 C0          JSR SQUARE
00714  CA59  38                SEC
00715  CA5A  AD BA 03          LDA RS
00716  CA5D  E5 AE             SBC SQR
00717  CA5F  85 AC             STA RADCND
00718  CA61  AD BB 03          LDA RS+1
00719  CA64  E5 AF             SBC SQR+1
00720  CA66  85 AD             STA RADCND+1
00721  CA68  20 64 C0          JSR SQRT
00722  CA6B  38                SEC
00723  CA6C  AD 8D 03          LDA RC
00724  CA6F  ED 3C 03          SBC ROOT
00725  CA72  8D 8B 03          STA MAX
00726  CA75  A9 00             LDA #$00
00727  CA77  8D 87 03          STA CNTY
00728  CA7A  AD 87 03  LLPY2   LDA CNTY
00729  CA7D  8D 70 03          STA YREL
00730  CA80  85 AD             STA MLPLER
00731  CA82  AD 8D 03          LDA RC
00732  CA85  85 AC             STA MLPCND
00733  CA87  20 11 C0          JSR MULT
00734  CA8A  85 FE             STA DVDND+1
00735  CA8C  A5 AE             LDA PROD
00736  CA8E  85 FD             STA DVDND
00737  CA90  AD 8B 03          LDA MAX
00738  CA93  85 FB             STA DVSOR
00739  CA95  20 EA C5          JSR SDIV
00740  CA98  A5 FD             LDA QUOT
00741  CA9A  38                SEC
00742  CA9B  ED 87 03          SBC CNTY
00743  CA9E  8D 71 03          STA YSHD
00744  CAA1  A5 FE             LDA QUOT+1
00745  CAA3  E9 00             SBC #$00
00746  CAA5  8D 72 03          STA YSHD+1
00747  CAA8  20 28 C6          JSR PTPLOT
00748  CAAB  AD 87 03          LDA CNTY
00749  CAAE  CD 8B 03          CMP MAX
00750  CAB1  F0 06             BEQ DDNY2
00751  CAB3  EE 87 03          INC CNTY
00752  CAB6  4C 7A CA          JMP LLPY2
00753  CAB9  AD 86 03  DDNY2   LDA CNTX
00754  CABC  CD 8C 03          CMP RT
00755  CABF  F0 06             BEQ DUNHSP
00756  CAC1  EE 86 03          INC CNTX
00757  CAC4  4C 3E CA          JMP LLPX2
00758  CAC7  60          DUNHSP RTS
00759  CAC8                    .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 ARG      00AC   BAKLIT   0382   CHCLUP   C690   CLIPD    0380
 CLIPL    037D   CLIPR    037E   CLIPU    037F   CNTNU    C600

SYMBOL TABLE

SYMBOL VALUE
 CNTX     0386   CNTY     0387   CXLOOP   C881   CYLNDR   C864
 CYLOOP   C875   DDNY1    CA2C   DDNY2    CAB9   DHEMI    C6BC
 DIVIDE   C025   DONE     C863   DONE4    C900   DONEHT   C90E
 DONEY    C855   DUNHSP   CAC7   DUNTOR   CA3A   DVDND    00FD
 DVSOR    00FB   EDGTOR   C88F   GETVAL   C5F1   GETZ     C745
 GRTR     C994   HEMI     0381   HLEN     0389   HTORRN   0346
 HVFLAG   0383   LLPX1    C935   LLPX2    CA3E   LLPY1    C99A
 LLPY2    CA7A   LOOPX    C805   LOOPX4   C89A   LOOPY    C82E
 LOOPY4   C8CC   MAX      0388   MLPCND   00AC   MLPLER   00AD
 MULT     C811   NEGATE   C5FF   NORM     C224   NOROT    C65A
```

End Listing Four

(Listing Five begins on page 76)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Four

```
NORSTR  C744    NOSCAL  0347    ORIGIN  C5EA    PLDONE  C721
PLTSHD  C20F    PROD    00AE    PTPLOT  C628    PTPLT2  C65D
QUOT    00FD    R0      0389    RADCND  00AC    RADIUS  0377
RAM     036A    RANDOM  C0C8    RC      038D    RHEMI   C6E8
RI      038F    RNDM    C000    RO      038E    ROOT    033C
RS      038A    RT      038C    SDIV    C5EA    SPHERE  C7C7
SPOOL   CA3B    SQR     00AE    SQRT    C064    SQUARE  C004
TEMP    0384    TNTMP   037B    TONE    0379    TOROID  C90F
TPARM   C79A    VALUE   0344    XCENT   036A    XMAX    0392
XPLT    033F    XREL    036C    XSHD    036D    XSQR    0390
YCENT   036F    YPLT    0341    YREL    0370    YSHD    0371
ZEROOT  C791    ZREL    0373    ZWX     0375

END OF ASSEMBLY
```

Listing Five

```
0000  0000              ; INTERFACE - EASY PARAMETER SETTING FOR SHAPE
0000  0000              ; DRAWING ROUTINES FROM BASIC.
0000  0000              ;
0000  0000              ; RICHARD L. RYLANDER  11/23/84
0000  0000              ;
0000  0000              ;****************************************
0000  0000              ORIGIN=$CAC8
0000  0000              RAM    =$0393
0000  0000              ;
0000  0000              ; PARAMETER LOCATIONS FOR VARIOUS SHAPES
0000  0000              ;
0000  0000              XCENT  =$036A
0000  0000              YCENT  =$036F
0000  0000              XPLOT  =$033F
0000  0000              YPLOT  =$0341
0000  0000              XMIN   =$034A
0000  0000              YMIN   =$034C
0000  0000              XMID   =$034D
0000  0000              YMID   =$034F
0000  0000              XMAX   =$0350
0000  0000              YMAX   =$0352
0000  0000              RADIUS =$0377
0000  0000              HLEN   =$0389
0000  0000              RI     =$038F
0000  0000              RO     =$038E
0000  0000              ;
0000  0000              HVFLAG =$0383
0000  0000              VALUE  =$0344
0000  0000              PLTFLG =$033E
0000  0000              ;
0000  0000              DEFLAG =$FB
0000  0000              ;
0000  0000              ;****************************************
0000  0000              ;
0000  0000              ; FUNCTION LOCATIONS
0000  0000              ;
0000  0000              GRFON  =$C0E2           ; SWITCH TO GRAPHICS MODE
0000  0000              GRFOFF =$C103           ; RETURN TO TEXT DISPLAY
0000  0000              ;
0000  0000              CLEARR=$C12C            ; CLEAR BITMAP
0000  0000              CLRBYT=$C135            ; CLEAR (FILL) BYTE
0000  0000              COLORR=$C118            ; LOAD COLOR MAP
0000  0000              COLBYT=$C119            ; COLOR BYTE
0000  0000              ;
0000  0000              PLOTR  =$C14B           ; POINT PLOT ROUTINE
0000  0000              LINER  =$C2DB           ; DRAW A LINE
0000  0000              FACETR =$C4E1           ; DRAW A SHADED FACET
0000  0000              ;
0000  0000              ;****************************************
0000  0000              ;
0000  0000              ; SHADED SHAPE DRAWING ROUTINES
0000  0000              ;
0000  0000              SPHERR=$C7C7            ; SPHERE
0000  0000              CYLNDR=$CB64            ; CYLINDER
0000  0000              TORUSR=$C90F            ; TOP-VIEW TOROID
0000  0000              EDGTOR=$CB8F            ; EDGE-VIEW TOROID
0000  0000              SPOOLR=$CA3B            ; INSIDE-VIEW TOROID
0000  0000              ;
0000  0000              ;****************************************
0000  0000              ;
0000  0000              ; BASIC ROM ROUTINES
0000  0000              ;
0000  0000              CHKCOM=$AEFD            ; CHECK FOR COMMA
0000  0000              EVAEXP=$AD9E            ; EVALUATE EXPRESSION
0000  0000              FLTFIX=$B1AA            ; CONVERT TO FIXED
0000  0000              ;
0000  0000              **RAM
0000  0393              LINFAC **+1             ; LINE OR FACET FLAG
0000  0394              ;
0000  0394              **ORIGIN
0000  CAC8              ;
0000  CAC8              ;****************************************
0000  CAC8              ;
0000  CAC8              ; GET PARAMETERS FROM BASIC CALLING STATEMENT
0000  CAC8              ; OF THE FORM:
0000  CAC8              ; SYS(FNCTN),PARAM1,PARAM2,PARAM3[OPT]
0000  CAC8              ; WHERE THE THIRD PARAMETER (FOR EXAMPLE)
0000  CAC8              ; MAY BE OPTIONAL (A DEFAULT VALUE IS USED
0000  CAC8              ; IF THE PARAMETER IS NOT SPECIFIED)
0000  CAC8              ;
0081  CAC8  20 FD AE    GETNUM JSR CHKCOM       ; LOOK FOR COMMA
0082  CAC8  20 9E AD          JSR EVAEXP        ; EVALUATE EXPRESSION
0083  CACE  20 AA B1          JSR FLTFIX        ; CHANGE TO INTEGER WITH
0084  CAD1                ; HIGH BYTE IN "A" AND LOW BYTE IN "Y"
0085  CAD1  60                RTS
0086  CAD2              ;
0087  CAD2              ; CHECK FOR ADDITIONAL (OPTIONAL) PARAMETERS
0088  CAD2              ;
0089  CAD2  A9 2C       PCHECK LDA #$2C         ; "," COMMA
0090  CAD4  A0 00            LDY #0
0091  CAD6  84 FB            STY DEFLAG
0092  CAD8  D1 7A            CMP ($7A),Y
0093  CADA  D0 03            BNE NOMORE        ; NO COMMA - USE DEFAULT
0094  CADC  4C 73 00          JMP $0073
0095  CADF  A0 80       NOMORE LDY #$80
0096  CAE1  84 FB            STY DEFLAG
0097  CAE3  60                RTS
0098  CAE4              ;
0099  CAE4              ; GET TWO ADDITIONAL PARAMETERS FOR TOROIDS
0100  CAE4              ;
0101  CAE4  20 D2 CA    GETTWO JSR PCHECK
0102  CAE7  24 FB            BIT DEFLAG
0103  CAE9  30 0F            BMI DFAULT
0104  CAEB  20 9E AD          JSR EVAEXP
0105  CAEE  20 AA B1          JSR FLTFIX
0106  CAF1  8C 8F 03          STY RI
0107  CAF4  20 C8 CA          JSR GETNUM
0108  CAF7  8C 8E 03          STY RO
0109  CAFA  60          DFAULT RTS
0110  CAFB              ;
0111  CAFB              ;****************************************
0112  CAFB              ;
0113  CAFB              ; SET CENTER COORDINATES
0114  CAFB              ;
0115  CAFB  20 C8 CA    CENTER JSR GETNUM
0116  CAFE  8C 6A 03          STY XCENT
0117  CB01  8D 6B 03          STA XCENT+1
0118  CB04  20 C8 CA          JSR GETNUM
0119  CB07  8C 6F 03          STY YCENT
0120  CB0A  60                RTS
0121  CB0B              ;
0122  CB0B              ;****************************************
0123  CB0B              ;
0124  CB0B              ; CLEAR THE BITMAP, FILLING WITH (OPTIONAL)
0125  CB0B              ; FILL VALUE SPECIFIED OR WITH (DEFAULT) "0"
0126  CB0B              ;
0127  CB0B  20 D2 CA    CLEAR2 JSR PCHECK
0128  CB0E  24 FB            BIT DEFLAG
0129  CB10  30 07            BMI DEFCLR
0130  CB12  20 9E AD          JSR EVAEXP
0131  CB15  20 AA B1          JSR FLTFIX
0132  CB18  2C                .BYTE #2C
0133  CB19  A0 00       DEFCLR LDY #0
0134  CB1B  8C 35 C1          STY CLRBYT
0135  CB1E  4C 2C C1          JMP CLEARR
0136  CB21              ;
0137  CB21              ;****************************************
0138  CB21              ;
0139  CB21              ; FILL COLOR MAP WITH (OPTIONAL) COLOR BYTE
0140  CB21              ; SPECIFIED OR WITH (DEFAULT) "$01"
0141  CB21              ; (BLACK DOTS ON WHITE BACKGROUND)
0142  CB21              ;
0143  CB21  20 D2 CA    COLOR2 JSR PCHECK
0144  CB24  24 FB            BIT DEFLAG
0145  CB26  30 07            BMI DEFCOL
0146  CB28  20 9E AD          JSR EVAEXP
0147  CB2B  20 AA B1          JSR FLTFIX
0148  CB2E  2C                .BYTE #2C
0149  CB2F  A0 01       DEFCOL LDY #$01
0150  CB31  8C 19 C1          STY COLBYT
0151  CB34  4C 18 C1          JMP COLORR
0152  CB37              ;
0153  CB37              ;****************************************
0154  CB37              ;
0155  CB37              ; PLOT OR UNPLOT POINTS
0156  CB37              ;
0157  CB37  A9 00       PLOT2  LDA #0
0158  CB39  2C                .BYTE #2C
0159  CB3A  A9 80       UNPLT2 LDA #$80
0160  CB3C  8D 3E 03          STA PLTFLG
0161  CB3F  20 C8 CA          JSR GETNUM
0162  CB42  8C 3F 03          STY XPLOT
0163  CB45  8D 40 03          STA XPLOT+1
0164  CB48  20 C8 CA          JSR GETNUM
0165  CB4B  8C 41 03          STY YPLOT
0166  CB4E  4C 4B C1          JMP PLOTR
0167  CB51              ;
0168  CB51              ;****************************************
0169  CB51              ;
0170  CB51              ; DRAW LINES BETWEEN (X1,Y1) AND (X2,Y2)
0171  CB51              ; OR SHADED FACETS BETWEEN THREE POINTS
0172  CB51              ; (X1,Y1), (X2,Y2) AND (X3,Y3)
0173  CB51              ;
0174  CB51  A9 00       LINE2  LDA #0
0175  CB53  2C                .BYTE #2C
0176  CB54  A9 80       FACET2 LDA #$80
0177  CB56  8D 93 03          STA LINFAC
0178  CB59  20 C8 CA          JSR GETNUM
0179  CB5C  8C 4A 03          STY XMIN
0180  CB5F  8D 4B 03          STA XMIN+1
0181  CB62  20 C8 CA          JSR GETNUM
0182  CB65  8C 4C 03          STY YMIN
0183  CB68  20 C8 CA          JSR GETNUM
0184  CB6B  8C 4D 03          STY XMID
0185  CB6E  8D 4E 03          STA XMID+1
0186  CB71  20 C8 CA          JSR GETNUM
0187  CB74  8C 4F 03          STY YMID
0188  CB77  2C 93 03          BIT LINFAC
0189  CB7A  10 18             BPL LDRAW
0190  CB7C  20 C8 CA          JSR GETNUM
0191  CB7F  8C 50 03          STY XMAX
0192  CB82  8D 51 03          STA XMAX+1
0193  CB85  20 C8 CA          JSR GETNUM
0194  CB88  8C 52 03          STY YMAX
0195  CB8B  20 C8 CA          JSR GETNUM
0196  CB8E  8C 44 03          STY VALUE
0197  CB91  4C E1 C4          JMP FACETR
0198  CB94  4C DB C2    LDRAW  JMP LINER
0199  CB97              ;
0200  CB97              ;****************************************
0201  CB97              ;
0202  CB97              ; DRAW A SPHERE CENTERED AT (XCENT,YCENT)
0203  CB97              ; DEFAULT RADIUS IS LAST VALUE USED
0204  CB97              ;
0205  CB97  20 FB CA    SPHER2 JSR CENTER
0206  CB9A  20 D2 CA          JSR PCHECK
0207  CB9D  24 FB             BIT DEFLAG
0208  CB9F  30 09             BMI SKIP1
0209  CBA1  20 9E AD          JSR EVAEXP
0210  CBA4  20 AA B1          JSR FLTFIX
0211  CBA7  8C 77 03          STY RADIUS
0212  CBAA  4C C7 C7    SKIP1  JMP SPHERR
0213  CBAD              ;
0214  CBAD              ;****************************************
0215  CBAD              ;
0216  CBAD              ; DRAW A TOP-VIEW TOROID AT (XCENT,YCENT)
0217  CBAD              ; DEFAULT INNER AND OUTER RADII ARE LAST USED
0218  CBAD              ;
0219  CBAD  20 FB CA    TORUS2 JSR CENTER
0220  CBB0  20 E4 CA          JSR GETTWO
0221  CBB3  4C 0F C9          JMP TORUSR
0222  CBB6              ;
0223  CBB6              ;****************************************
0224  CBB6              ;
0225  CBB6              ; DRAW CYLINDERS WITH AXES HORIZONTAL OR
0226  CBB6              ; VERTICAL.  DEFAULT RADIUS AND "HALF-LENGTH"
0227  CBB6              ; ARE LAST VALUES USED.
0228  CBB6              ;
0229  CBB6  A9 80       VCYL2  LDA #$80
0230  CBB8  2C                .BYTE #2C
0231  CBB9  A9 00       HCYL2  LDA #0
0232  CBBB  8D 83 03          STA HVFLAG
0233  CBBE  20 FB CA          JSR CENTER
0234  CBC1  20 D2 CA          JSR PCHECK
0235  CBC4  24 FB             BIT DEFLAG
0236  CBC6  30 0F             BMI SKIP2
0237  CBC8  20 9E AD          JSR EVAEXP
```

(Continued on page 78)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Five

```
00238  CBCB  20 AA B1          JSR FLTFIX
00239  CBCE  8C 77 03          STY RADIUS
00240  CBD1  20 C8 CA          JSR GETNUM
00241  CBD4  8C 89 03          STY HLEN
00242  CBD7  4C 64 CB    SKIP2 JMP CYLNDR
00243  CBDA                    ;
00244  CBDA                    ;**************************************
00245  CBDA                    ;
00246  CBDA                    ; DRAW EDGE-VIEW TOROIDS WITH AXES HORIZONTAL
00247  CBDA                    ; OR VERTICAL
00248  CBDA                    ; INNNER AND OUTER RADII ARE OPTIONAL
00249  CBDA                    ;
00250  CBDA  A9 80       VTOR2 LDA ##80
00251  CBDC  2C                .BYTE #2C
00252  CBDD  A9 00       HTOR2 LDA #0
00253  CBDF  8D 83 03          STA HVFLAG
00254  CBE2  20 FB CA          JSR CENTER
00255  CBE5  20 E4 CA          JSR GETTWO
00256  CBE8  4C 8F CB          JMP EDGTOR
00257  CBEB                    ;
00258  CBEB                    ;**************************************
00259  CBEB                    ;
00260  CBEB                    ; DRAW INSIDE-VIEW TOROIDS, "SPOOLS",
00261  CBEB                    ; WITH AXES HORIZONTAL OR VERTICAL
00262  CBEB                    ; INNER AND OUTER RADII ARE OPTIONAL
00263  CBEB                    ;
00264  CBEB  A9 80       VSPL2 LDA ##80
00265  CBED  2C                .BYTE #2C
00266  CBEE  A9 00       HSPL2 LDA #0
00267  CBF0  8D 83 03          STA HVFLAG
00268  CBF3  20 FB CA          JSR CENTER
00269  CBF6  20 E4 CA          JSR GETTWO
00270  CBF9  4C 3B CA          JMP SPOOLR
00271  CBFC                .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 CENTER   CAFB     CHKCOM   AEFD     CLEAR2   CB0B     CLEARR   C12C
 CLRBYT   C135     COLBYT   C119     COLOR2   CB21     COLORR   C118
 CYLNDR   CB64     DEFCLR   CB19     DEFCOL   CB2F     DEFLAG   00FB
 DFAULT   CAFA     EDGTOR   CB8F     EVAEXP   AD9E     FACET2   CB54
 FACETR   C4E1     FLTFIX   B1AA     GETNUM   CAC8     GETTWO   CAE4
 GRFOFF   C103     GRFON    C0E2     HCYL2    CBB9     HLEN     0389
 HSPL2    CBEE     HTOR2    CBDD     HVFLAG   0383     LDRAW    CB94
 LINE2    CB51     LINER    C2DB     LINFAC   0393     NOMORE   CADF
 ORIGIN   CAC8     PCHECK   CAD2     PLOT2    CB37     PLOTR    C14B
 PLTFLG   033E     RADIUS   0377     RAM      0393     RI       038F
 RO       038E     SKIP1    CBAA     SKIP2    CBD7     SPHER2   CB97
 SPHERR   C7C7     SPOOLR   CA3B     TORUS2   CBAD     TORUSR   C90F
 UNFLT2   CB3A     VALUE    0344     VCYL2    CBB6     VSPL2    CBEB
 VTOR2    CBDA     XCENT    036A     XMAX     0350     XMID     034D
 XMIN     034A     XPLOT    033F     YCENT    036F     YMAX     0352
 YMID     034F     YMIN     034C     YPLOT    0341

END OF ASSEMBLY
```

End Listing Five

Listing Six

```
10 REM        SHAPES DEMO
20 :
30 REM      RICHARD L. RYLANDER     11/23/84 (REVISED 1/20/85 TO ADD LABELING)
40 :
50 GR=49378    :REM GRAPHICS MODE
60 TX=49411    :REM TEXT MODE
70 :
80 LB=893      :REM LEFT BOUND
90 RB=894      :REM RIGHT BOUND
100 UB=895     :REM UP BOUND
110 DB=896     :REM DOWN BOUND
120 :
130 REM   FLAGS FOR VARIOUS DRAWING MODES
140 :
150 SH=838     :REM SHADE STYLE - 0=RANDOM, 1=HALFTONE
160 SC=839     :REM SCALING - 0=NORMAL (1:1), 1=SCALED (3:4) FOR SCREEN DISPLAY
170 LT=898     :REM LIGHTING - 0=NORMAL SINGLE-SOURCE, 1=BACKLIT ILLUMINATION
180 :
190 BO=53280   :REM  BORDER COLOR
200 :
210 REM   FUNCTION LOCATIONS
220 :
230 CL=51979   :REM CLEAR BITMAP AREA
240 CO=52001   :REM FILL COLOR MAP
250 :
260 SP=52119   :REM  SPHERE
270 TR=52141   :REM  TOP-VIEW TOROID
280 VC=52150   :REM  CYLINDER (AXIS VERTICAL)
290 HC=52153   :REM  CYLINDER (AXIS HORIZONTAL)
300 VT=52186   :REM  EDGE-VIEW TOROID (AXIS VERTICAL)
310 HT=52189   :REM  EDGE-VIEW TOROID (AXIS HORIZONTAL)
320 VS=52203   :REM  INSIDE-VIEW TOROID ("SPOOL") (AXIS VERTICAL)
330 HS=52206   :REM  INSIDE-VIEW TOROID ("SPOOL") (AXIS HORIZONTAL)
340 :
350 REM  DRAW SAMPLES OF PRIMITIVE SHAPES WITH LABEL BENEATH EACH
360 :
370 POKE SH,1 :REM HALFTONE SHADING
380 POKE SC,1 :REM USE SCREEN SCALING (FOR UNDISTORTED SPHERES, ETC.)
390 POKE LT,0 :REM USE NORMAL (NO BACKLIGHT) ILLUMINATION
400 SYS(CL)   :REM CLEAR SCREEN (DEFAULT - FILL BITMAP WITH 0'S)
410 SYS(CO),16*11+1  :REM COLOR COMBINATION - DARK GRAY (11) DOTS ON WHITE (1)
412 REM    LOOKS BETTER THAN DEFAULT COLOR COMBINATION OF BLACK DOTS ON WHITE
414 REM    ON MOST COLOR MONITORS
420 POKE BO,1 :REM WHITE BORDER TO MATCH BACKGROUND
430 SYS(GR)   :REM ENTER BITMAPPED GRAPHICS MODE
432 RW=12:CM=14:MD=1:A#="SAMPLE SHAPES":GOSUB 1900:REM  PUT TITLE ON IMAGE
434 X1=110:Y1=120:X2=210:Y2=120:BC=1:DC=7:GOSUB 1700:REM  COLOR TITLE TEXT
440 POKE LB,38:POKE RB,38:POKE UB,38:POKE DB,38       :REM INITIALIZE CLIPPING
450 SYS(SP),40,199,38
455 RW=9:CM=2:MD=1:A#="SPHERE":GOSUB1900
460 SYS(HC),120,199,38,38
465 CM=11:A#="H-CYLNDR":GOSUB1900
470 SYS(VC),200,199 :REM NO SIZE PARAMETERS GIVEN, DEFAULT (PREVIOUS) ARE USED
475 CM=21:A#="V-CYLNDR":GOSUB1900
480 SYS(TR),280,199,15,38
485 CM=32:A#="TOROID":GOSUB1900
490 SYS(VT),40,64
495 RW=23:CM=1:A#="V-TOROID":GOSUB1900
500 SYS(HT),120,64
505 CM=11:A#="H-TOROID":GOSUB1900
510 SYS(HS),200,64,5,100
515 CM=22:A#="H-SPOOL":GOSUB1900
520 SYS(VS),280,64
525 CM=32:A#="V-SPOOL":GOSUB1900
530 POKE 198,0:WAIT 198,1:POKE 198,0
540 REM  WAIT FOR A KEY TO BE PRESSED
550 :
560 REM  DRAW TWO "GOBLETS" ONE WITH HALFTONE, THE OTHER RANDOM SHADING.
570 :
580 SYS(CL):SYS(CO),16*11+1
582 RW=14:CM=15:A#="COMPARISON":GOSUB 1900:RW=15:CM=19:A#="OF":GOSUB 1900
584 RW=16:CM=14:A#="SHADE STYLES":GOSUB 1900:RW=18:CM=14:A#="<-- HALFTONE"
586 GOSUB 1900:RW=20:CM=15:A#="RANDOM -->":GOSUB 1900
590 POKE LB,255:POKE RB,255:POKE UB,49:POKE DB,255: REM CLIP AT SCREEN TOP
600 SYS(SP),80,190,80
610 POKE UB,51:POKE DB,51: REM CLIP AT JUNCTION WITH SPHERE FOR SMOOTH SEAM
620 SYS(VS),80,69,10,130
630 POKE DB,9:POKE UB,8
640 SYS(VT),80,9,25,45
650 POKE SH,0  :REM SWITCH TO RANDOM SHADING
660 POKE LB,255:POKE RB,255:POKE UB,49:POKE DB,255
670 SYS(SP),240,190,80
680 POKE UB,51:POKE DB,51
690 SYS(VS),240,69,10,130
700 POKE DB,9:POKE UB,8
710 SYS(VT),240,9,25,45
720 POKE 198,0:WAIT 198,1:POKE 198,0
740 :
750 REM  DRAW "WINE" SCENE
760 POKE LT,1 :REM BACKLIT ILLUMINATION
770 POKE SH,1 :REM HALFTONE SHADING FOR MOST (RANDOM "LABEL" ON BOTTLE)
780 SYS(CO):SYS(CL),255: REM FILL BITMAP WITH 1'S ("SET" BACKGROUND)
790 POKE BO,0 : REM BLACK BORDER TO MATCH BACKGROUND
792 RW=0:CM=0:MD=2:A#="DRAWING  WITH":GOSUB 1900:RW=1:CM=2:A#="BACKLIGHT"
794 GOSUB 1900:RW=2:CM=0:A#="AGAINST A SET":GOSUB 1900
796 RW=3:CM=2:A#="BACKGROUND":GOSUB 1900
798 RW=1:CM=26:A#="COLORS  ADDED":GOSUB 1900
800 RW=2:CM=25:A#="TO SELECT AREAS":GOSUB 1900
810 REM  DRAW BOTTLE
820 POKE UB,0:POKE DB,255:POKE LB,255:POKE RB,255
830 SYS(VT),150,10,30,50
840 POKE UB,255:SYS(VC),150,70,50,60
850 POKE DB,0:SYS(VT),150,130,6,50
860 POKE DB,55:POKE UB,0:SYS(VS),150,204,15,181
870 POKE UB,255:SYS(VC),150,221,16,17
880 :
890 REM  DRAW WINE GLASS
900 POKE UB,20:SYS(SP),80,120,60
910 POKE UB,35:POKE DB,34:SYS(VS),80,34,10,110
920 :
930 REM  DRAW SOME GRAPES
940 SYS(SP),8,8,8: REM OTHER "GRAPES" WILL BE SAME RADIUS - JUST GIVE POSITIONS
950 SYS(SP),20,8:SYS(SP),40,8:SYS(SP),12,20:SYS(SP),30,20:SYS(SP),25,16
960 :
970 REM  DRAW AN APPLE BY A PAIR OF EDGE-VIEW TOROIDS AND A SPHERE SECTION
980 POKE UB,255:POKE DB,255:POKE LB,255:POKE RB,59
990 SYS(VT),260,29,0,50:SYS(VT),260,79
1000 POKE UB,43:POKE DB,43:SYS(SP),260,54,60
1010 REM   PUT STEM ON APPLE
1020 POKE RB,0:POKE DB,0:SYS(TR),272,104,10,15
1030 REM   ADD A LEAF BY A SPHERE SECTION
1040 POKE DB,255:POKE RB,0:SYS(SP),256,119,15
1050 REM  ADD A RANDOM SHADED "LABEL" TO THE BOTTLE
1060 POKE UB,255:POKE RB,255:POKE LB,6
1070 POKE,SH,0:SYS(VC),150,72,50,48
1080 :
1090 REM  ADD COLORS TO "WINE" SCENE
1100 SYS(CO),12:REM REPLACE WHITE "HOLES" (BACKGROUND) WITH MEDIUM GRAY
1102 X1=0:Y1=200:X2=100:Y2=239:DC=0:BC=0:GOSUB 1700:REM HIDE TEXT IN CORNERS
1104 X1=180:Y1=200:X2=319:Y2=239:DC=0:BC=0:GOSUB 1700
1110 X1=200:Y1=1:X2=315:Y2=100:DC=0:BC=2:GOSUB 1700
1120 X1=240:Y1=110:X2=255:Y2=150:BC=5:GOSUB 1700
1130 X1=260:Y1=110:X2=270:Y2=135:BC=9:GOSUB 1700
1140 X1=1:Y1=1:X2=48:Y2=30:BC=4:GOSUB 1700
1150 X1=140:Y1=205:X2=180:Y2=235:BC=7:GOSUB 1700
1160 X1=145:Y1=25:X2=195:Y2=115:BC=6:GOSUB 1700
1170 POKE 198,0:WAIT 198,1:POKE 198,0
1190 :
1200 REM   "COFFEE AND DONUTS"
1210 POKE SH,0 :REM RANDOM SHADING ON DONUTS
1220 SYS(CO),16*11+1:SYS(CL):POKE BO,1:REM WHITE BORDER TO MATCH BACKGROUND
1230 POKE LB,255:POKE RB,255:POKE UB,255:POKE DB,255 :REM NO INITIAL CLIPPING
1240 SYS(VT),60,20,20,60
1250 POKE RB,29:SYS(VT),99,60:POKE RB,255
1260 SYS(TR),188,180
1270 REM ADD COFFEE CUP (HALFTONE SHADING)
1280 POKE SH,1:POKE UB,0:SYS(VT),188,20:POKE UB,255
1290 POKE DB,0:SYS(TR),278,110,20,40
1300 POKE DB,255:POKE UB,0:POKE LB,0
1310 SYS(TR),248,90,50,70:POKE LB,255:SYS(SP),248,110,10:POKE UB,255
1320 SYS(VC),308,100,10,10
1330 SYS(VC),188,77,60,57
1340 POKE DB,0:SYS(VT),188,134,40,60
1350 REM  ADD COLOR - FIRST MAKE SCREEN BROWN DOTS ON WHITE THEN MAKE CUP GREEN
1360 SYS(CO),1+16*9 :REM 1=WHITE BACKGROUND, 9=BROWN DOTS
1370 X1=130:Y1=1:X2=319:Y2=136:BC=1:DC=5:GOSUB 1700
1380 X1=250:Y1=144:X2=319:Y2=144:GOSUB 1700
1390 POKE 198,0:WAIT 198,1:POKE 198,0
1410 :
1420 REM  DRAW "LINKED" TOROIDS BY REDRAWING OVERLAP AREAS USING CLIPPING
1430 POKE LT,0 :REM BLUE DOTS ON WHITE, NO BACKLIGHT
1440 SYS(CL):SYS(CO),1+16*6: REM 1=WHITE BACKGROUND, 6=BLUE DOTS
1450 POKE UB,255:POKE DB,255:POKE LB,255:POKE RB,255: REM NO INITIAL CLIPPING
1460 POKE SH,0   :REM RANDOM SHADING
1470 SYS(TR),244,84,48,70
1480 SYS(TR),160,84:SYS(TR),76,84
1490 SYS(TR),118,156:SYS(TR),202,156
1500 REM  ADD PROPER OVERLAP SECTIONS
1510 POKE RB,0:POKE DB,0:SYS(TR),160,84:POKE RB,255:POKE LB,0
1520 SYS(TR),76,84:POKE DB,255:POKE UB,0:SYS(TR),118,156
1530 POKE LB,255:POKE RB,0:SYS(TR),202,156:POKE LB,27
1540 POKE DB,0:POKE UB,255:SYS(TR),160,84
1550 POKE DB,0:POKE UB,255:SYS(TR),160,84
1560 POKE LB,255:POKE UB,27:SYS(TR),244,84
1570 POKE LB,0:POKE RB,27:POKE UB,255:SYS(TR),244,84
1580 POKE 198,0
1590 GET A#:IF A#= "" THEN 1590
1600 SYS(TX):POKE BO,14:REM RETURN TO TEXT MODE
1610 END
1620 :
1630 REM SUBROUTINE FOR ADDING COLOR TO DIFFERENT SCREEN AREAS
1640 REM REMEMBER THAT COLOR BOUNDARIES MUST CORRESPOND TO CHARACTER BOUNDARIES
1650 REM DEFINE A RECTANGULAR AREA BY LOWER-LEFT AND UPPER-RIGHT COORDINATES
1660 REM (X1,Y1)=LOWER-LEFT POINT, (X2,Y2)=UPPER-RIGHT POINT
1670 REM THE CORNER POINTS CAN BE ARBITRARY BUT COLOR CHANGE WILL BE IN THE
1680 REM SMALLEST CHARACTER-CELL BOUNDED RECTANGLE THAT INCLUDES THOSE POINTS
1690 REM  Y-COORDINATES MUST BE "UNSCALED" IF SCALE FLAG IS SET:
1700 IF PEEK(SC)THEN Y1=(Y1+1)*213/256:Y2=(Y2+1)*213/256
1710 REM COLOR TO BE POKED IN IS GIVEN BY VARIABLE CC="COMPOSITE COLOR"
1720 REM WHERE CC=16*DC + BC [DC=DOT COLOR, BC=BACKGROUND COLOR NUMBERS]
1730 CC=16*DC+BC
1740 FOR IX=INT(X1/8) TO INT(X2/8)
1750 FOR IY=INT(Y1/8) TO INT(Y2/8)
```

(Continued on page 80)

Drawing on the C-64 (Listing Continued, text begins on page 50)

Listing Six

```
1760 POKE 34752+IX-40*IY,CC
1770 NEXT:NEXT:RETURN
1780 :
1790 REM SUBROUTINE TO ADD TEXT TO GRAPHIC SCREEN.
1800 REM "RW" AND "CM" ARE THE ROW (0-24) AND COLUMN (0-39) COORDINATES FOR THE
1810 REM FIRST LETTER OF THE TEXT STRING TO BE PRINTED.
1820 REM THE TEXT STRING ITSELF IS ASSIGNED TO "A$".
1830 REM "MD" IS THE MODE FOR THE PRINTING. FIVE MODES ARE ALLOWED:
1840 REM  1 - NORMAL ("BLACK" LETTERS ON "WHITE" BACKGROUND)
1850 REM  2 - REVERSED ("WHITE" LETTERS ON "BLACK" BACKGROUND)
1860 REM  3 - SET ("BLACK") LETTERS "OR'ED" WITH BACKGROUND
1870 REM  4 - UNSET ("WHITE") LETTERS "AND'ED" WITH BACKGROUND
1880 REM  5 - SET LETTERS "XOR'ED" WITH BACKGROUND
1890 :
1900 SB=40952:TB=54272:IF(MD AND 1)THEN TB=53248:REM SCREEN AND TEXT BASE ADDR'S
1910 OS=320*RW+8*CM:REM  OFFSET FROM CHARACTER SCREEN BASE
1920 POKE 56334,PEEK(56334)AND 254:REM DISABLE IRQ TIMER
1930 POKE 1,PEEK(1)AND 251:REM SWITCH CHARACTER ROM IN
1940 L=LEN(A$):FOR N=1 TO L:N8=N8+N*8+OS+SB
1950 X=ASC(MID$(A$,N,1)):IF X>63 THEN X=X-64
1960 TC=TB+8*X
1970 ON MD GOTO 1980,1980,1990,2000,2010
1980 POKE 53231,36:GOTO 2020
1990 POKE 53231,17:GOTO 2020
2000 POKE 53231,49:GOTO 2020
2010 POKE 53231,81
2020 POKE252,N8/256:POKE251,N8-256*INT(N8/256)
2030 POKE254,TC/256:POKE253,TC-256*INT(TC/256)
2040 SYS(53221):NEXT
2050 POKE 1,PEEK(1)OR 4:POKE 56334,PEEK(56334)OR 1:REM RESTORE TO NORMAL
2060 RETURN
```

READY.

End Listing Six

Listing Seven

```
10 REM "STELLATION" DRAW A SMALL STELLATED DODECAHEDRON IN VARIOUS ORIENTATIONS
20 REM  AND STYLES    RICHARD L. RYLANDER      12/5/84
30 :
40 GR=49378     :REM GRAPHICS MODE
50 TX=49411     :REM TEXT MODE
60 BO=53280     :REM BORDER COLOR
70 :
80 REM  INITIALIZE A FEW STYLE PARAMETERS
90 POKE 839,1   :REM SCALE (3:4) FOR UNDISTORTED SCREEN DISPLAY
100 POKE 871,0 :REM FACET EDGE/LINE MODE (0=DRAW, 1=ERASE)
110 SH=838      :REM SHADE STYLE (0=RANDOM, 1=HALFTONE)
120 EG=868      :REM EDGES FLAG (0=NORMAL, 1=ADD LINES TO FACET EDGES)
130 :
140 REM   FUNCTION LOCATIONS
150 CL=51979    :REM CLEAR BITMAP
160 CO=52001    :REM FILL COLOR MAP
170 FC=52052    :REM DRAW A SHADED FACET
180 KS=53081    :REM DO KEYED SORT
190 :
200 XC=160:YC=120  :REM (SCALED) SCREEN CENTER COORDINATES
210 :
220 PRINT"{SC}     ********************************"
230 PRINT"     * SMALL STELLATED DODECAHEDRON *"
240 PRINT"     ********************************"
250 PRINT"{CD}{CD} SELECT SHADING STYLE:"
260 PRINT" R=RANDOM, H=HALFTONE"
270 INPUT"{CD} YOUR CHOICE    H{CL}{CL}{CL}";A$
280 POKE SH,0:IF A$="H" THEN POKE SH,1
290 PRINT"{CD}{CD} SELECT STYLE FOR DRAWING:"
300 PRINT"{CD} N - NORMAL":PRINT"{CD} E - EDGES EMPHASIZED"
310 PRINT"{CD} W - WIRE FRAME (HIDDEN LINES REMOVED)"
320 INPUT"{CD} YOUR CHOICE    N{CL}{CL}{CL}";A$
330 POKE EG,0:WI=0:IF A$="N" THEN 360
340 POKE EG,1:IF A$="W" THEN WI=-1
350 :
360 PRINT"{CD} READING VERTEX DATA"
370 VN=32:DIM F%(VN-1,2):REM  VN = NUMBER OF VERTICES
380 FOR N=0 TO VN-1:READ P%(N,0),P%(N,1),P%(N,2):NEXT
390 :
400 PRINT"{CD} ENTER X, Y, AND Z ANGLES FOR ROTATION"
410 PRINT"        (ANGLES IN DEGREES)"
420 INPUT X,Y,Z
430 J=3.14159265/180:X=X*J:Y=Y*J:Z=Z*J
440 X0=COS(Y)*COS(Z)-SIN(X)*SIN(Y)*SIN(Z):X1=COS(Y)*SIN(Z)+SIN(X)*SIN(Y)*COS(Z)
450 X2=-COS(X)*SIN(Y):Y0=-COS(X)*SIN(Z):Y1=COS(X)*COS(A):Y2=SIN(X)
460 Z0=SIN(Y)*COS(Z)+SIN(X)*COS(Y)*SIN(Z)
470 Z1=SIN(Y)*SIN(Z)-SIN(X)*COS(Y)*COS(Z):Z2=COS(X)*COS(Y)
480 PRINT"{CD} PERFORMING ROTATION"
490 FOR N=0 TO VN-1
500 X=P%(N,0):Y=P%(N,1):Z=P%(N,2)
510 P%(N,0)=X0*X+X1*Y+X2*Z:P%(N,1)=Y0*X+Y1*Y+Y2*Z:P%(N,2)=Z0*X+Z1*Y+Z2*Z:NEXT
520 :
530 FA=60: REM  TOTAL NUMBER OF FACETS
540 DIM F%(FA/2,2),SH(FA/2),Z%(FA/2),K%(FA/2)
550 PRINT" READING CONNECTION DATA          "
560 VF=-1: REM  VF = # VISIBLE FACETS
570 FOR N=1 TO FA
580 VF=VF+1
590 FOR I=0 TO 2 :READ F%(VF,I):NEXT
600 REM  CALCULATE COMPONENTS OF NORMAL VECTOR TO DETERMINE VISIBILITY
610 Z=(P%(F%(VF,2),0)-P%(F%(VF,1),0))*(P%(F%(VF,0),1)-P%(F%(VF,1),1))
620 Z=Z-(P%(F%(VF,0),0)-P%(F%(VF,1),0))*(P%(F%(VF,2),1)-P%(F%(VF,1),1))
630 IF Z<=0 THEN 720:REM  FACET NOT VISIBLE, GO ON
640 X=(P%(F%(VF,2),1)-P%(F%(VF,1),1))*(P%(F%(VF,0),2)-P%(F%(VF,1),2))
650 X=X-(P%(F%(VF,0),1)-P%(F%(VF,1),1))*(P%(F%(VF,2),2)-P%(F%(VF,1),2))
660 Y=(P%(F%(VF,2),2)-P%(F%(VF,1),2))*(P%(F%(VF,0),0)-P%(F%(VF,1),0))
670 Y=Y-(P%(F%(VF,0),2)-P%(F%(VF,1),2))*(P%(F%(VF,2),0)-P%(F%(VF,1),0))
680 NC=SQR(X*X+Y*Y+Z*Z):REM  LENGTH OF NORMAL VECTOR
690 SH(VF)=26*(2*Z+X+Y)/NC
700 SH(VF)=(SH(VF)+64)*(SH(VF)+64)/256:REM RAISED COSINE SHADING
710 GOTO 730
720 VF=VF-1
730 NEXT
740 :
750 PRINT" SCALING TO DISPLAY SIZE"
760 Y=0:FOR N=0 TO VN-1:IF ABS(P%(N,1))>Y THEN Y=ABS(P%(N,1))
770 NEXT:S=119/Y
780 FOR N=0 TO VN-1:P%(N,1)=S*P%(N,1)+YC:P%(N,0)=S*P%(N,0)+XC:NEXT
790 :
800 REM FIND AVERAGE Z FOR EACH FACET
810 FOR N=0 TO VF
820 Z%(N)=(P%(F%(N,0),2)+P%(F%(N,1),2)+P%(F%(N,2),2))/3:NEXT
830 :
840 PRINT" SORTING FACETS ACCORDING TO AVG 'Z' "
850 POKE 140,VF
860 K%(0)=K%(0):POKE 251,PEEK(71):POKE 252,PEEK(72)
870 Z%(0)=Z%(0):POKE 253,PEEK(71):POKE 254,PEEK(72)
880 SYS(KS)
890 :
900 REM DRAW FACETS (ACCORDING TO Z DEPTH SINCE NOT CONVEX)
910 SYS(GR):SYS(CO):SYS(CL):POKE BO,1
920 FOR N=0 TO VF:FA=K%(N)
930 IF WI THEN SH(FA)=64
940 X0=P%(F%(FA,0),0):Y0=P%(F%(FA,0),1):X1=P%(F%(FA,1),0):Y1=P%(F%(FA,1),1)
950 X2=P%(F%(FA,2),0):Y2=P%(F%(FA,2),1)
960 SYS(FC),X0,Y0,X1,Y1,X2,Y2,SH(FA)
970 NEXT
980 POKE198,0
990 GET A$:IF A$="" THEN 990
1000 SYS(TX):POKE BO,14:END
1010 :
1020 REM      VERTEX DATA
1030 DATA 1000,618,0, 1000,-618,0, -1000,618,0, -1000,-618,0
1040 DATA 0,1000,618, 0,1000,-618, 0,-1000,618, 0,-1000,-618
1050 DATA 618,0,1000, -618,0,1000, 618,0,-1000, -618,0,-1000
1060 DATA 618,0,236, 618,0,-236, -618,0,236, -618,0,-236
1070 DATA 236,618,0, -236,618,0, 236,-618,0, -236,-618,0
1080 DATA 0,236,618, 0,-236,618, 0,236,-618, 0,-236,-618
1090 DATA 382,382,382, 382,382,-382, 382,-382,382, 382,-382,-382
1100 DATA -382,382,382, -382,382,-382, -382,-382,382, -382,-382,-382
1110 :
1120 REM    CONNECTION DATA
1130 DATA 0,12,13, 0,13,25, 0,25,16, 0,16,24, 0,24,12
1140 DATA 1,12,26, 1,26,18, 1,18,27, 1,27,13, 1,13,12
1150 DATA 2,15,14, 2,14,28, 2,28,17, 2,17,29, 2,29,15
1160 DATA 3,14,15, 3,15,31, 3,31,19, 3,19,30, 3,30,14
1170 DATA 4,16,17, 4,17,20, 4,20,20, 4,20,24, 4,24,16
1180 DATA 5,17,16, 5,16,25, 5,25,22, 5,22,29, 5,29,17
1190 DATA 6,19,18, 6,18,26, 6,26,21, 6,21,30, 6,30,19
1200 DATA 7,18,19, 7,19,31, 7,31,23, 7,23,27, 7,27,18
1210 DATA 8,20,21, 8,21,26, 8,26,12, 8,12,24, 8,24,20
1220 DATA 9,21,20, 9,20,28, 9,28,14, 9,14,30, 9,30,21
1230 DATA 10,23,22, 10,22,25, 10,25,13, 10,13,27, 10,27,23
1240 DATA 11,22,23, 11,23,31, 11,31,15, 11,15,29, 11,29,22
```

READY.

End Listing Seven

Listing Eight

```
00001  0000                ; KEYSORT - RELOCATABLE BUBBLE SORT USING KEY ARRAY
00002  0000                ; POINTING TO INTEGER ARRAY
00003  0000                ;
00004  0000                ;  RICHARD L. RYLANDER    1/12/85
00005  0000                ;
00006  0000                ORIGIN=$CF59            ; 53081. (FOLLOWING DOS 5.1)
00007  0000                ;
00008  0000                KB      = $FB           ; 251. POINTER TO KEY ARRAY
00009  0000                ZB      = $FD           ; 253. POINTER TO DATA ARRAY
00010  0000                MAX     = $8C           ; 140. POKE WITH MAX ARRAY INDEX
00011  0000                TOP     = $AC
00012  0000                TOPDIS  = $AD
00013  0000                FLAG    = $AE
00014  0000                NXTFLG  = $61
00015  0000                CRRNT   = $62
00016  0000                REPEAT  = $64
00017  0000                ;
00018  0000                        *=ORIGIN
00019  CF59                ;
00020  CF59  A0 FF         INIT    LDY #$FF        ; INITIALIZE KEY ARRAY
00021  CF5B  C8            INLOOP  INY
00022  CF5C  98                    TYA
00023  CF5D  91 FB                 STA (KB),Y
00024  CF5F  C5 8C                 CMP MAX
00025  CF61  D0 F8                 BNE INLOOP
00026  CF63                ;
00027  CF63  85 AD         SORT    STA TOPDIS      ; 'A' HOLDS 'MAX'
00028  CF65  A5 AD         LOOP1   LDA TOPDIS
00029  CF67  85 AC                 STA TOP
00030  CF69  A2 00                 LDX #0
00031  CF6B  86 61                 STX NXTFLG
00032  CF6D  86 AE                 STX FLAG
00033  CF6F  86 64         LOOP2   STX REPEAT
00034  CF71                ;
00035  CF71                ; GET BOTH BYTES OF INTEGER POINTED TO BY
00036  CF71                ; 'KEY' ELEMENT.  RETURN WITH MSB ON STACK
00037  CF71                ; AND LSB IN THE ACCUMULATOR
00038  CF71                ;
00039  CF71  8A            GETINT  TXA
00040  CF72  A8                    TAY
00041  CF73  B1 FB                 LDA (KB),Y
00042  CF75  0A                    ASL A
00043  CF76  90 04                 BCC LOAD
00044  CF78  C6 61                 DEC NXTFLG
00045  CF7A  E6 FE                 INC ZB+1
00046  CF7C  A8            LOAD    TAY
00047  CF7D  B1 FD                 LDA (ZB),Y
00048  CF7F  48                    PHA
00049  CF80  C8                    INY
00050  CF81  B1 FD                 LDA (ZB),Y
00051  CF83  24 61                 BIT NXTFLG
00052  CF85  10 04                 BPL NODEC
00053  CF87  E6 61                 INC NXTFLG
00054  CF89  C6 FE                 DEC ZB+1
00055  CF8B  E4 64         NODEC   CPX REPEAT
00056  CF8D  D0 08                 BNE ORDER
00057  CF8F  85 62                 STA CRRNT
00058  CF91  68                    PLA
00059  CF92  85 63                 STA CRRNT+1
00060  CF94  E8                    INX
00061  CF95  D0 DA                 BNE GETINT
00062  CF97                ;
00063  CF97                ; COMPARE INTEGERS OBTAINED THROUGH KEY ARRAY
00064  CF97                ; IF 'CURRENT' >= 'NEXT' THEN SWAP KEY
00065  CF97                ; ELEMENTS, ELSE CONTINUE
00066  CF97                ;
00067  CF97  C5 62         ORDER   CMP CRRNT
00068  CF99  68                    PLA
00069  CF9A  E5 63                 SBC CRRNT+1
00070  CF9C  50 02                 BVC TEST
00071  CF9E  49 80                 EOR #$80
00072  CFA0  10 13         TEST    BPL NOSWAP
00073  CFA2  8A            SWAP    TXA
00074  CFA3  A8                    TAY
00075  CFA4  86 AD                 STX TOPDIS
00076  CFA6  B1 FB                 LDA (KB),Y
00077  CFA8  48                    PHA
00078  CFA9  88                    DEY
00079  CFAA  B1 FB                 LDA (KB),Y
00080  CFAC  C8                    INY
00081  CFAD  91 FB                 STA (KB),Y
00082  CFAF  68                    PLA
00083  CFB0  88                    DEY
00084  CFB1  91 FB                 STA (KB),Y
```

(Continued on page 82)

Drawing on the C-64

(Listing Continued, text begins on page 50)

Listing Eight

```
00085  CFB3  E6 AE           INC FLAG
00086  CFB5  E4 AC     NOSWAP CPX TOP
00087  CFB7  D0 B6           BNE LOOP2
00088  CFB9  A5 AE           LDA FLAG
00089  CFBB  D0 A8           BNE LOOP1
00090  CFBD                 ;
00091  CFBD                 ; UNPACK THE BYTE ELEMENTS OF THE 'KEY' ARRAY
00092  CFBD                 ; INTO BASIC'S NORMAL 2-BYTE INTEGER FORMAT
00093  CFBD                 ;
00094  CFBD  A6 BC     UNPACK LDX MAX
00095  CFBF  E8               INX
00096  CFC0  CA        PKLOOP DEX
00097  CFC1  8A               TXA
00098  CFC2  A8               TAY
00099  CFC3  B1 FB            LDA (KB),Y
00100  CFC5  48               PHA
00101  CFC6  8A               TXA
00102  CFC7  0A               ASL A            ; MOVE TO 2*I+1
00103  CFC8  09 01            ORA #1
00104  CFCA  90 04            BCC STORE
00105  CFCC  E6 61            INC NXTFLG
00106  CFCE  E6 FC            INC KB+1
00107  CFD0  A8        STORE  TAY
00108  CFD1  68               PLA
00109  CFD2  91 FB            STA (KB),Y
00110  CFD4  A9 00            LDA #0
00111  CFD6  88               DEY
00112  CFD7  91 FB            STA (KB),Y
00113  CFD9  A5 61            LDA NXTFLG
00114  CFDB  F0 04            BEQ OK
00115  CFDD  C6 61            DEC NXTFLG
00116  CFDF  C6 FC            DEC KB+1
00117  CFE1  8A        OK     TXA
00118  CFE2  D0 DC            BNE PKLOOP
00119  CFE4  60        DONE   RTS
00120  CFE5                 .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 CRRNT    0062   DONE     CFE4   FLAG     00AE   GETINT   CF71
 INIT     CF59   INLOOP   CF5B   KB       00FB   LOAD     CF7C
 LOOP1    CF65   LOOP2    CF6F   MAX      00BC   NODEC    CF8B
 NOSWAP   CFB5   NXTFLG   0061   OK       CFE1   ORDER    CF97
 ORIGIN   CF59   PKLOOP   CFC0   REPEAT   0064   SORT     CF63
 STORE    CFD0   SWAP     CFA2   TEST     CFA0   TOP      00AC
 TOPDIS   00AD   UNPACK   CFBD   ZB       00FD

END OF ASSEMBLY
```

End Listing Eight

Listing Nine

```
00001  0000                 ;   "WRITE"   RICHARD L. RYLANDER
00002  0000                 ;             12/30/84
00003  0000                 ; REVISED  1/19/85  - ORIGIN MOVED TO $CFE5 (53221.)
00004  0000                 ;
00005  0000                 ;  PUT TEXT CHARACTERS ON GRAPHIC SCREEN
00006  0000                 ;  (UNDER BASIC ROM) IN VARIOUS STYLES
00007  0000                 ;
00008  0000                 *=$CFE5                ; PUT CODE AFTER DOS 5.1
00009  CFE5  A5 01          WRITE  LDA $01         ; SWITCH OUT BASIC ROM
00010  CFE7  29 FE                 AND #$FE
00011  CFE9  85 01                 STA $01
00012  CFEB  A0 07                 LDY #7
00013  CFED  B1 FD          LOOP   LDA ($FD),Y     ; READ CHARACTER BYTE
00014  CFEF  31 FB                 AND ($FB),Y     ; MODIFY W/SCREEN BYTE
00015  CFF1  91 FB                 STA ($FB),Y     ; STORE IN SCREEN
00016  CFF3                 ;
00017  CFF3                 ; POKE NEW LOGICAL OPERATOR TO REPLACE
00018  CFF3                 ; 'AND' (53231.) FOR DIFFERENT STYLES
00019  CFF3                 ; ORA=17.  BIT (NOP)=36.  AND=49.  EOR=81.
00020  CFF3                 ;
00021  CFF3  88                    DEY
00022  CFF4  10 F7                 BPL LOOP
00023  CFF6  A5 01                 LDA $01         ; RESTORE BASIC ROM
00024  CFF8  09 01                 ORA #1
00025  CFFA  85 01                 STA $01
00026  CFFC  60                    RTS
00027  CFFD                 .END

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE
 LOOP     CFED    WRITE    CFE5

END OF ASSEMBLY
```

End Listings