BY DAVID MEREDITH

DISPLAYING MOVING GRAPHICS

Are those flat, static computer graphics boring you? Would you like to display moving pictures of three-dimensional objects, pictures that move around the screen, grow bigger and smaller, and rotate in space? My program, Display, does all this.

Display is a hybrid Basic/machine-language program that allows a 32K CoCo to show moving high-resolution images of three-dimensional objects. The objects displayed are *skeletons* (points joined by line segments), and the possible motions include sliding left, right, up, and down (called *panning*); growing larger and smaller (called *scaling*); and rotating six different ways.

To use Display, first enter each point you want to display. Then specify which pairs of points are to be connected and type DISPLAY. Your picture will appear on the screen within one second.

Then, by pressing various command keys, you can move your picture around the screen, make it grow larger or smaller, or make it appear to rotate in space. You can stop the display at any time to change points or line segments or to create an entirely new picture, and then return to the display. Finally, if you like your picture, you can save it on tape.

System Requirements

32K RAM
Extended Color Basic
Editor/Assembler

You can create 3-dimensional objects and manipulate them in space using this amazing program.

The Microworld of Display

Throughout this article, I will use "object" to refer to the three-dimensional skeleton to be displayed and "picture" to mean the resulting graphics display.

Display creates a microworld, as explained by Seymour Pappert in *Mindstorms* (New York: Basic Books, 1980). That is, when you use Display, you are operating in a limited but complete and coherent logical environment. This is a three-dimensional world populated by points and line segments. The points have names: A, B, C, . . . Z, so there can be 26 of them at most.

When defining a point, you specify its name and its x-,y-,z-coordinates. You define line segments by naming their endpoints. When the screen displays your picture, the origin of the coordinate system is at the center of the TV screen, the x-axis goes left to right, the y-axis goes bottom to top, and the z-axis points right at you. Each screen pixel corresponds to one unit on a coordinate axis.

When you define points, you must specify their coordinates as integers between −80 and 80, but when the program runs, the coordinates can take on real values between −4,096 and 4,096. Large coordinate values can be generated when you allow a picture to slide far in one direction or to grow very large. Because coordinates are kept as real values (accurate to 1/256), you can shrink pictures to a single dot on the screen and then expand them again without losing detail.

If your object grows large, the TV screen will show only part of it. Only points with x-coordinate between −128 and 127 and y-coordinate between −96 and 95 will be visible. If any part of a line segment falls in this range, that part of the line segment will be shown, even though the ends of the line may not be visible. Thus, you can create a complex object, allow it to grow larger than the screen, and then display different magnified parts of it while retaining the entire object in your computer's memory.

How to Use Display

First load the Basic component of the program, then type and enter PCLEAR8:RUN. (The reason for PCLEAR8 is explained in Note 1 at the end of this article.) The program first executes CLOADM"DISPLAY" to load its machine-language component (which must be ready for loading when you type RUN) and then displays the message, "Enter HELP anytime for guidance." At this point, or at any other time when the text screen is visible, you can create, modify, erase, verify, save, load, and display objects.

To define (or redefine) a point, type

the name of the point, then =, and then the coordinates separated by commas, as in the following:

```
A= 0,0,0,
F= -50,50,-50
```

The coordinates must be integers between −80 and 80.

To specify a line segment, just enter the endpoints like this: AF. To erase a line segment, enter the endpoints followed by #: AF#. You can define or erase more than one line segment on one line. For example, ACCBDE#HF would define line segments AC, CB, and HF and would erase DE. Up to 60 line segments can be included in one object.

To print the coordinates of all the points you have defined, enter ?. You can display a subrange of defined points with commands such as the following:

```
?C-G
?H-
?-G
```

To list all of the line segments you have defined, type ??.

To save the current object on tape, prepare your tape recorder for writing a new file and type SAVE"name". Your object will be saved under whatever file name you designate. As with all tape files, the name can include up to eight alphabetic characters, or you can omit the file name.

To read an object from tape, enter LOAD"name".

The program saves objects and loads them as binary files with the CLOADM and CSAVEM commands (see Note 2). Actually, all that is saved on tape are the buffers holding the points and line segments—355 bytes in all. That is all it takes to define an object.

Now for the rest of Display's commands: To erase an object entirely, just type NEW. To get a screenful of helpful information about using Display, type HELP. Finally, to see a picture of your object, type DISPLAY.

When you type DISPLAY the Basic program does four operations. It reads the list of line segments and marks all unused points as undefined. This prevents the machine-language routines from wasting time moving points that are not displayed. It then POKEs 0 into memory cell $11A (282) to deactivate the uppercase keyboard lock, executes the machine-language component of Display, and it POKEs 255 into memory cell $11A to restore the uppercase keyboard lock.

While executing the machine-language program, you can move the picture of your object 13 ways, as described in Table 1.

Machine-Language Routines

The clearest way of describing the machine-language component of Display is with a pseudocode outline. I will use as pseudocode subroutine names the same mnemonics that are used in the Assembly listing.

The program begins and ends with some housekeeping:

Disable regular interrupts
Establish a new S-stack
Execute MAIN
Restore the S-stack (for Basic)
Return.

The routine MAIN is the traffic cop. It does the following:

Sets the VDG register and the control register for high-resolution graphics
Executes DISPLA (Display a picture of the object)
Repeats

↙See List of Advertisers on page 130

Waits for the command key
If the command key is in [↑, ↓, →, ←, b, s, X, Y, Z, z, y, z], then it repeats
Modifies object according to command key
Executes DISPLA until key is no longer depressed
Endif
Until key = @
Returns

Basic Data Structures

To understand how Display pans, scales, rotates, and displays objects, you must first see how the program stores points and line segments. There are separate buffers for points and lines. The 234-byte buffer POINTS starting at $7000, allots 9 bytes to each of the 26 points, A–Z. If a point is undefined, $80 is stored in its first byte. (Since all coordinates must lie between -2^{12} and 2^{12}, you will see that $80 can never be the first byte of a defined point.)

Each point has three coordinates stored in order x, y, z, and each coordinate is stored as a 3-byte signed integer that is 256 times the coordinate value. Alternatively, think of each 3-byte coordinate as a signed hexadecimal value accurate to two hexadecimal digits (8 bits) to the right of the hexadecimal point.

Line segments are stored as 2-byte records in the 121-byte buffer LINES starting at $70EA. The program packs the records in the low end of the buffer followed by a single termination byte containing $FF. The 2 bytes defining a line segment contain the address of the endpoints of the line relative to POINTS. That is, point A corresponds to 0, B to 9, C to $12 = 18, and so forth. Thus, the 2 bytes defining the line segment AC contain 0,$12; the 2 bytes defining the line segment DK contain $1B,$5A (decimal 27,90).

The line-segment buffer is a table of pointers. If ADDR is the address of a line-segment record, and if (ADDR) means the contents of ADDR, then the address of the first endpoint of the line segment is POINTS + (ADDR), and the address of the second endpoint is POINTS + (ADDR + 1).

I've divided the rest of this article into two parts. First, I will explain the routines that pan, scale, and rotate your object. These change the coordinates of the points without affecting the line segments at all. I move the object by moving the points and then reconstructing the line segments between the newly positioned points.

Second, I will explain how the program creates the picture of your object. When you press a command key, the program alternately modifies the object a little and then displays a new picture of the repositioned object. In this way the object seems to slide around your TV screen, grow larger or smaller, or rotate continuously in space.

Moving the Object

Whether you are panning, scaling, or rotating, you move the object in small increments between successive displays.

Panning is the easiest motion to achieve. Each time the program calls one of the routines, PANL, PANR, PANU, or PAND, it adds or subtracts two from the x- or y-coordinate of each defined point. For example, to slide the object to the right, call PANR. This subroutine adds two to the signed number in the first 2 bytes of the x-coordinate of each point. (Remember that these 2 bytes represent the integer part of the coordinate.)

Scaling is almost as easy. Each time the program executes the subroutine BIGGER, each coordinate of each defined point grows by 1/32. SMALLR shrinks coordinates by 1/32. In each case, the scaling is achieved by placing each coordinate on the S-stack, shifting it 5 bits to the right (dividing by 32), and adding or subtracting the result from the original coordinate value.

Now I come to the rotations. There are six of them, as you can rotate two ways around each of three axes. Each rotation subroutine (ROTX, ROTY, ROTZ, ROTMX, ROTMY, and ROTMZ) rotates all the points 7.18 degrees one way around one of the axes. I selected this amount of rotation because sin(7.18°) is almost equal to 1/8 and cos(7.18°) is almost equal to 127/128.

To rotate a point around an axis, you must modify two of the coordinates of the point. Each of the six rotation subroutines works the same way. Each goes through the list of points and, for each defined point, moves the addresses of the two coordinates to be modified to the X-register and Y-register. Then it calls a subroutine ROTATE to effect the rotation.

The six different ways in which the subroutines can point the X-register and Y-register at two out of three coordinates correspond to the six possible rotations.

Let's look in detail at rotating a single point counter-clockwise around the x-axis. Call the original coordinates of the point x_{old}, y_{old}, and z_{old}. The coordinates of the point after rotation will be as follows:

$$x_{new} = x_{old}$$

$$y_{new} = \frac{127}{128} y_{old} - \frac{1}{8} z_{old}$$

$$z_{new} = \frac{1}{8} y_{old} + \frac{127}{128} z_{old}$$

Aficionados of linear algebra will recognize these equations:

$$x_{new} = x_{old}$$
$$y_{new} = y_{old}\cos(\theta) - z_{old}\sin(\theta)$$
$$z_{new} = z_{old}\sin(\theta) + z_{old}\cos(\theta)$$

θ is the angle of rotation. I chose the

Command key	Action
↑, ↓, ←, →	move your object around the screen (panning)
B, S	make your object bigger or smaller (scaling)
X, Y, Z	rotate your object counter-clockwise around the X-, Y-, or Z-axis
(Shift) X, Y, Z	rotate your object clockwise around one of the axes
@	return to the Basic program

These commands continue working as long as you depress the command key.

Table 1. Commands for Object Movement

```
Clear the unseen graphics screen.
For each line segment PQ in the object
   X1 = INT(x-coordinate of P) (INT takes the integer part)
   Y1 = INT(y-coordinate of P)
   X2 = INT(x-coordinate of Q)
   Y2 = INT(y-coordinate of Q)
Execute Line (draw a line on the unseen screen from (X1,Y1) to (X2,Y2))
ENDFOR.
Display this graphics screen.
```

Table 2. A Pseudocode Version of DISPLA

angle $\theta = 7.18°$ so that multiplication by $\cos(\theta) = 127/128$ and $\sin(\theta) = 1/8$ would require only a few arithmetic shifts, followed by a single addition or subtraction. (The values given for sine and cosine are accurate to one part in 25,000.)

The program uses an algorithm slightly different from the one above. No distinction is made between old and new coordinates. Rather, the program executes the following equivalent sequence:

$$y = (127/128)y - (1/8)z$$
$$z = (1/8)y + (129/128)z$$

If you compare this last algorithm with the one preceding it, you will see that they do not have exactly the same result. However, for the accuracy required by the program, they are sufficiently close.

The routines for panning, scaling, and rotating all contain checks within themselves to see that points do not grow too large. Coordinates must remain between $-4{,}096$ and 4,096. If one of the movement routines pushes a point out of bounds, the program restores the offending point and all points preceding it in the points buffer to their original state. That is, the current, partially completed incremental change is undone.

In addition, objects are not permitted to grow too small. After the subroutine SMALLR reduces an object, the program checks each line segment. If any segment has endpoints in which the coordinates all have equal integer parts (the 2 high-order bytes of each coordinate), then the program expands the object to its previous size. Note that if the coordinates of two points all have equal integer parts, then the program will plot two points at the same place on the screen no matter how you rotate the object in space.

Drawing the Picture

While you press a command key, the program alternately moves the object very slightly and then displays the picture of the object in its new position.

To draw the picture of the object, the program goes to the subroutine DISPLA, which clears the screen and then draws each line segment in turn. Actually, you never see DISPLA clearing the screen or drawing the line segments. Clearing and drawing are done on an "invisible" graphics screen before the program displays the picture.

The CoCo can display a high-resolution picture based on information contained in $1800 bytes beginning at any multiple of $200. I use what the Basic manual calls the first four graphics pages ($600–$1DFF) to hold one graphics screen, and graphics pages 5–8 ($1E00–$35FF) for the second graphics screen. One of these screens is being displayed on your TV whenever Display is running.

When the program calls DISPLA, it clears (sets to 0) the part of memory devoted to the other graphics screen and draws the picture of the object there. Then it displays the new graphics screen and uses the other one for the next picture of the object.

The variable TL ($721F–$7220) keeps track of which graphics screen is being displayed. *Getting Started with Color Basic*, pp. 259–260, explains switching graphics screens.

```
      IF X1≥LEFT AND Y1≥Y THEN (The Left endpoint is already on the visible screen)
        GOTO LN3
      XTEMP=X2
      YTEMP=Y2
LN4   X=(X1+XTEMP) DIV 2 (DIV 2 means divide by 2 and take the integer part)
      Y=(Y1+YTEMP) DIV 2
      IF (X≤LEFT AND Y≥TOP) OR (X≥RIGHT AND Y≤BOTTOM) THEN
      RETURN (No part of the line is visible)
      IF X<LEFT OR Y<BOTTOM THEN
        X1=X
        Y1=Y
        GOTO LN4
      ELSE IF X>LEFT AND Y>BOTTOM THEN
        XTEMP=X
        YTEMP=Y
        GOTO LN4
      ELSE ((X,Y) is on the left or bottom boundary of the visible screen)
        X1=X (These are the corrected values for X1 and Y1)
        Y1=Y
LN 3  .....
```

Table 3. Pseudocode Version of Clip Routine

Table 2 gives a pseudocode version of DISPLA. You might note two points about this routine. First of all, I ignore the z-coordinate of each point. That is because the value of the z-coordinate does not affect the location of a point in the picture. Second, I use only the integer parts of the x- and y-coordinates. Truncating is faster than rounding and no less precise. Points separated by more than one unit are displayed separately on the screen.

The last routine I want to discuss is the line-drawing routine, LINE. Four parameters, X1, Y1, X2, and Y2 are passed to LINE, and it draws a line on the screen from (X1,Y1) to (X2,Y2). Remember that (0,0) is the center of the screen, the x-coordinate goes left to right, and the y-coordinate goes bottom to top.

These conventions are different from Radio Shack's, which place (0,0) at the top left corner and measure the Y-coordinate top to bottom. Radio Shack's coordinates are perfect for describing the location of characters on a printed page; mine are those used universally by scientists, engineers, and mathematicians for two-dimensional graphics.

LINE begins by comparing X1 and X2, and swapping (X1,Y1) with (X2,Y2) if necessary, to assure that $X1 \leq X2$. Then LINE divides into two branches. I will discuss the case $Y1 \leq Y2$; the other case, handled by SELine (south east line), is similar.

To recapitulate, I want to draw a line from (X1,Y1) to (X2,Y2), and I have arranged matters so that $X1 \leq X2$ and $Y1 \leq Y2$. The coordinates X1, Y1, X2, and Y2 are between $-4{,}096$ and 4,096, but you can only see a point (X,Y) if $-128 \leq X \leq 127$ and $-96 \leq Y \leq 95$.

The first thing to do is to check to see if any part of the line will fall into the visible part of the screen. I define some constants corresponding to the borders of the visible screen:

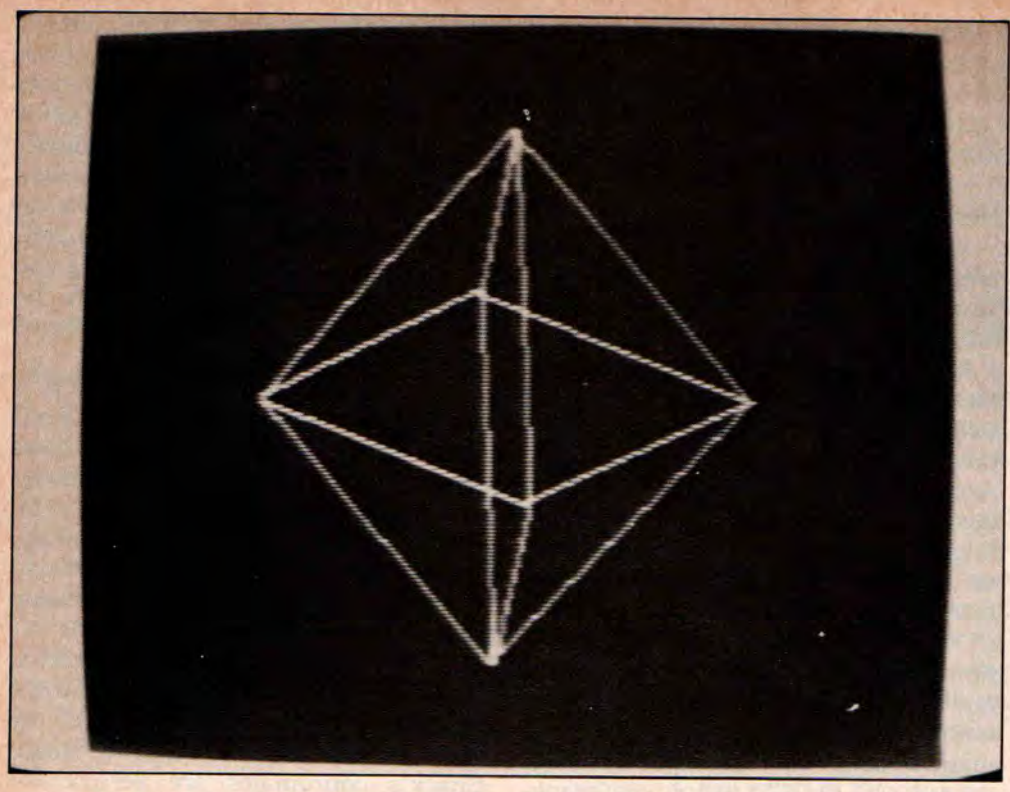
LEFT = −$80 = −128
RIGHT = $7F = 127
TOP = $5F = 95
BOTTOM = −$60 = −96

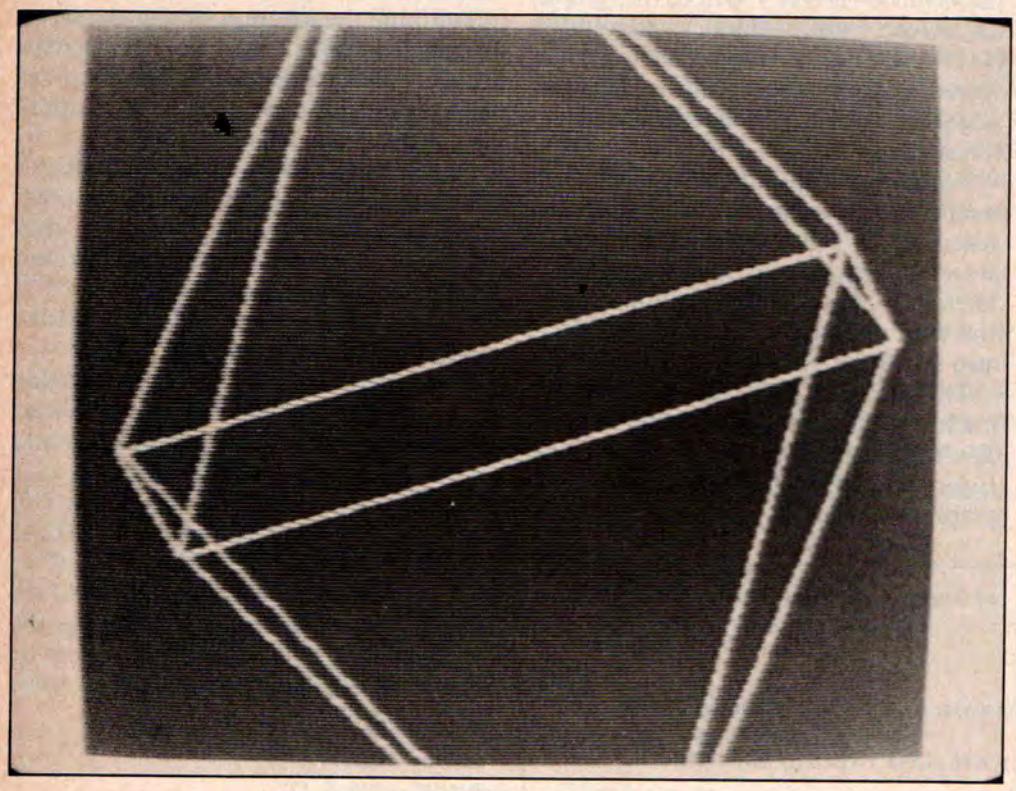
Then I execute the following routine to skip invisible lines:

IF X1≥RIGHT OR Y1≥TOP OR X2≤LEFT OR Y2≤BOTTOM THEN RETURN FROM SUBROUTINE LINE (No part of the line falls into the visible screen)

This test eliminates only some of the invisible lines. Others will be eliminated below.

Next, I clip the line. That is, I cut off that part of the line that doesn't show





Photos 1a and 1b. Normal and Enlarged Views of a Diamond Shape

on the screen by making the values of (X1,Y1) and (X2,Y2) equal to the endpoints of the visible portion of the line to be drawn. An algorithm analogous to a binary search does the clipping.

I will describe in pseudocode how (X1,Y1) are reset, if necessary, from their original values to the left endpoint of the visible portion of the line. A similar process resets (X2,Y2) equal to the right endpoint of the visible portion of the line. Of course, if an endpoint of the line is already visible on the screen, then the clipping process leaves it unchanged.

Note finally that while clipping away the invisible part of a line, I may discover that the entire line lies outside the visible screen. In that case I quietly execute a RTS and return from the subroutine LINE.

Table 3 shows the pseudocode version of the routine that clips the left end of the line from (X1,Y1) to (X2,Y2). Remember, $X1 \leq X2$ and $Y1 \leq Y2$.

When I have clipped the endpoints of my line where necessary, I know that I have endpoints (X1,Y1) and (X2,Y2) on the visible screen, with (X2,Y2) northeast of (X1,Y1). More precisely:

$-\$80 \leq X1 \leq X2 \leq \$7F$
$-\$60 \leq Y1 \leq Y2 \leq \$5F$

The routine that sets the pixels on the screen for the line from (X1,Y1) to (X2,Y2) mimics the Basic instruction LINE(X1,Y1)-(X2,Y2),PSET. Actually, there are two routines: one for the case $Y2-Y1<X2-X1$, and the other for $Y2-Y1 \geq X2-X1$. The first case corresponds to the routine in Line beginning at ENELN (east north east line). The second is similar.

If you wanted to draw a line on the screen, you might try something like this:

```
SLOPE=(Y2-Y1)/(X2-X1)
Y=Y1
FOR X=X1 TO X2 DO
  YINT=INT(Y) (The integer part of Y).
  PSET(X,YINT) (X is always an integer—PSET
    turns on the pixel at (X,YINT) ).
  Y=Y + SLOPE
ENDFOR
```

You could write this routine in Basic using the PSET function. You would draw a straight line, but the processing would be painfully slow. Even a machine-language routine modeled on this algorithm is slow, because using a function like PSET requires that you recompute the byte and bit corresponding to (X,YINT) for each different value of X between X1 and X2.

There is a better way. As X is incremented, the byte and bits corresponding to the points (X,YINT) follow each other in a regular manner. The basic idea is as follows:

```
SLOPE = (Y2-Y1)/(X2 - X1)
Start at pixel corresponding to (X1,Y1)
DO X2 - X1 + 1 TIMES
  light up the pixel
  move over one pixel and up slope pixels
ENDDO
```

Unfortunately, slope is a fraction. Since you can't move up a fraction of a pixel, you must do something like this:

```
SLOPE = (Y2- Y1)/(X2 - X1)
B = 0
Start at pixel corresponding to (X1,Y1)
DO X2-X1 + 1 TIMES
  light up the pixel
  move over one pixel
  B = B + SLOPE
  IF B ≥ 1 THEN B = B - 1; move up one pixel
ENDDO
```

This is the algorithm realized by the routine ENELN. To explain how I coded it, I must first explain which byte and bit correspond to the pixel at a point (X,Y).

Let TL be the address of the byte corresponding to the top left corner of the high-resolution screen. Either TL = $600 or TL = $1E00, depending on which graphics screen you are drawing.

Bit 7 of byte TL controls the pixel at the top left corner of the screen, and the next seven pixels of the top row are controlled by bits 6–0 of byte TL. Bits 7–0 of byte TL + 1 control the next eight pixels. Bits 7–0 of byte TL + $20 = TL + 32 control the first eight pixels of the second row.

Since there are 192 rows of pixels on the high-resolution graphics screen, the pixel in the lower right corner of the screen is controlled by bit 0 of byte TL+ $17FF = TL + 6143.

Computing the bit and byte corresponding to a point (X,Y) uses integer division (DIV) and the remainder function (MOD). Remember that N DIV M is the largest integer not greater than N/M. For example, 8 DIV 2 = 4; 7 DIV 3 = 2; and −7 DIV 3 = −3. N MOD M = M*(N/M − N DIV M) so 8 MOD 2 = 0; 7 MOD 3 = 1; and −7 MOD 3 = 2.

Since I have placed (0,0) at the center of the screen, and since I let the Y-coordinate increase as it goes up the screen, I compute the byte and bit corresponding to the point (X,Y) as follows:

```
BYTE(X,Y) = TL + ($59 − Y)*$20 +
  ($80 + X) DIV 8
BIT(X,Y) = 7 − X MOD 8
```

Table 4 shows a new version of the line-drawing routine. The algorithm in this table faithfully mirrors the Assembly routine beginning at ENELN. The main complication comes in computing BIT (X,Y) and setting the bit to 1.

The U-register contains none of the numbers 0–7, but rather one of the addresses, BITS...LASTBT. BITS contains the binary number 10000000; BITS+1 contains 01000000, and so forth, until BITS+7=LASTBT contains 00000001.

The U-register contains the address of one of these words, which are called *masks*, indicating which bit is to be set to 1. The address of the byte to be modified is in the X-register, and the instructions bit UREG of byte XREG = 1 is expanded into the following sequence:

```
LDA ,X  get the byte to be modified
ORA ,U  set the required bit to 1
STA ,X  return the modified byte to screen
  memory.
```

Program Listing 1. Basic Portion of Display

```
100 REM PROGRAM DISPLAY
110 REM COPYRIGHT DAVID MEREDITH L983
120 REM BASIC PROGRAM PERMITS INPUT OF A 3-D PICTURE AS POINTS A
ND LINE SEGMENTS.  MACHINE LANGUAGE COMPONENT DISPLAYS THE PICTU
RE
130 REM AND ALLOWS PANNING, SCALING, AND ROTATING
140 CLS:PRINT@73,"D I S P L A Y":PRINT@135,"BY DAVID MEREDITH":P
RINT@201,"COPYRIGHT 1983"
145 PRINT@416,"READING MACHINE LANGUAGE PART..."
150 CLOADM"DISPLAYM":PCLEAR8:CLEAR 100,&H7000
160 PO=&H7000:LI=PO+9*26:MA=LI+121:REM ADDRESSES OF POINTS BUFFE
R, LINES BUFFER, AND MAIN DISPLAY ROUTINE
165 DIM CH(25)
170 GOSUB 1000: REM NEW PICTURE
175 CS$="HELPNEWDISPLAY":LD$="LOAD":SV$="SAVE":AA=ASC("A"):QU$=C
HR$(34)
180 CLS:PRINT"ENTER help ANYTIME FOR GUIDANCE"
190 LINEINPUT A$:REM GET NEW POINT, LINE, OR COMMAND
200 I=INSTR(A$," "):IFI<>0THENA$=LEFT$(A$,I-1)+RIGHT$(A$,LEN(A$)
-I):GOTO200:REM ELIMINATE BLANKS
210 IFLEN(A$)>2THENI=INSTR(CS$,A$):IFI=0THEN215ELSEIFI=1THENGOSU
B1600:GOTO190ELSEIFI=5THENGOSUB1000:GOTO190ELSEIFI=8THENGOSUB180
0:GOTO190
215 IF INSTR(A$,LD$)=1THENGOSUB1200:GOTO190ELSEIFINSTR(A$,SV$)=1
THENGOSUB1400:GOTO190
220 IFINSTR(A$,"=")=2THENGOSUB2000:GOTO190:REM DEFINE A POINT
230 IFINSTR(A$,"??")=1 THEN GOSUB 2200:GOTO190:REM PRINT LINE SE
GMENTS
240 IFINSTR(A$,"?")=1 THENGOSUB2400:GOTO190:REM PRINT POINTS
250 REM AT THIS POINT A$ EITHER DEFINES LINES OR IS INCORRECT
260 L=LEN(A$):IFL<2THENGOSUB2600:GOTO190:REM IF LEN(A$)<2 THEN L
INE INCORRECT
270 IFL<2THEN190ELSEA1=ASC(LEFT$(A$,1))-AA:A2=ASC(MID$(A$,2,1))-
AA:IF A1<0 OR A1>25 OR A2<0 OR A2>25 THEN GOSUB2600:GOTO190ELSE
L=L-2:A$=RIGHT$(A$,L)
274 IFA1=A2 THENPRINT:PRINT"ENDPOINTS MUST BE DISTINCT":PRINT:GO
TO190
275 IFINSTR(A$,"#")=1THENL=L-1:A$=RIGHT$(A$,L):GOTO400:REM DELET
E A POINT
280 I=LI:A1=9*A1:A2=9*A2
```

Listing continued

The entire routine LINE seems to produce lines identical to those drawn by the Basic command, LINE.

Conclusion

Here are some sample objects you can try with Display. The first is a diamond. Enter points and line segments as follows:

```
A =0,0,50
B =0,0,−50
C =0,50,0
D =0,−50,0
E =50,0,0
F =−50,0,0
AC AD AE AF BC BD BE BF CE CF DE DF
```

Note: Press enter after typing each line segment for this and all other sample objects.

Here are the specifications for a cube (when it first appears on the screen, all you see is a square, because the back half is hiding behind the front half):

```
A =0,0,0
B =50,0,0
C =50,50,0
D =0,50,0
E =0,0,50
F =50,0,50
G =50,50,50
H =0,50,50
AB BC CD DA EF FG GH HE
AE BF CG DH
```

Finally, you can draw groups of objects on the screen. Here, for example, are the four letters, W, O, R, D. You can make them spin and move just like the animated logos in a TV commercial:

```
A =−64,0,0
B =−64,31,0
C =−55,15,0
D =−45,0,0
E =−45,31,0
F =−32,0,0
G =−32,31,0
H =−13,31,0
I =−13,0,0
J =0,0,0
K =0,15,0
L =0,31,0
M =19,31,0
N =19,15,0
O =19,0,0
P =32,0,0
Q =32,31,0
R =48,31,0
S =51,28,0
T =51,3,0
U =48,0,0
AB AC CD DE FG GH HI IF JK KL LM MN
NK KO PQ QR RS ST TU UP
```

Display has two limitations. The first is a software limitation that could be remedied by any reader. You cannot

program moving graphics displays with this program; you can only control them in real time. However, you could program a graphics display by calling separately the subroutines that pan, scale, rotate, and display objects. Just remember to disable the regular interrupt each time you call one of the subroutines.

The second limitation is due to the hardware. The display cannot be made to move faster or to show more complex animation. The 6809 is a wonderful chip, but it can only work so fast.

Moving graphics systems more powerful than Display take lots of computing power—more than can be expected of an 8-bit processor unsupported by special graphics hardware. I do not believe this program could be recoded to run more than twice as fast as it does now. Most of the running time is spent in the line-drawing routine, which seems fairly efficient to me.

Professional graphics systems use special-purpose hardware and often some form of parallel processing to speed up their displays. However, Display contains some of the basic software elements of a moving graphics system, and the hardware requirement is certainly reasonable—about 10 percent of the cost of a high-quality color-graphics terminal. So, enjoy experimenting with moving graphics, and write to me if you find some interesting ways to use this program, or if you just create some unusual objects.

Final Notes

1) Executing PCLEAR8 before running Display seems to overcome an obscure bug in the Basic instruction EXEC.

Display is loaded from tape into memory beginning with the fifth graphics page. Executing PCLEAR8 in line 150 should move the program beyond the eighth graphics page, so that the machine-language routines that alter the memory corresponding to the graphics pages should not affect the program. But it does not work that way. Try the following program and watch it crash:

```
10 CLEAR 100, &H3900
20 PCLEAR 8
30 FOR I=&H3900 TO &H390F
40 READ X
50 POKE I,X
60 NEXT I
70 EXEC &H3900
80 PRINT "SUCCESSFUL RETURN
```

Continues on p. 127

Listing continued

```
290 P=PEEK(I):Q=PEEK(I+1):IFP<>255THENIF(P=A1 ANDQ=A2)OR(P=A2 AN
DQ=A1)THENPRINT:PRINT"LINE ";CHR$(AA+A1/9);CHR$(AA+A2/9);" ALREA
DY DEFINED":PRINT:GOTO190ELSEI=I+2:GOTO290:REM FIND NEXT OPEN SP
ACE IN LINES BUFFERCHECKING FOR DUPLICATION
300 IF I=MA-1 THENPRINT:PRINT"NO ROOM FOR ANOTHER LINE":PRINT:GO
TO190
310 IFPEEK(PO+A1)=128THENPRINT:PRINT"POINT ";CHR$(A1/9+AA);" NOT
 DEFINED":PRINT:GOTO270
320 IFPEEK(PO+A2)=128THENPRINT:PRINT"POINT ";CHR$(A2/9+AA);" NOT
 DEFINED":PRINT:GOTO270
330 POKEI,A1:POKEI+1,A2:POKEI+2,255:IFL>0THEN270ELSE190:REM PUT
LINE SEGMENT IN LINE BUFFER AND GET NEXT SEGMENT IF ANY
400 REM DELETE A LINE SEGMENT A1,A2
410 I=LI:A1=9*A1:A2=9*A2
420 IFPEEK(I)=255 THENPRINT:PRINT"LINE SEGMENT ";CHR$(A1/9+AA);C
HR$(A2/9+AA);" NOT DEFINED":PRINT:GOTO270
430 P=PEEK(I):Q=PEEK(I+1):IF(P<>A1 ORQ<>A2)AND(P<>A2 ORQ<>A1)THE
NI=I+2:GOTO420
440 P=PEEK(I+2):POKEI,P:IFP=255THEN270ELSEI=I+1:GOTO440:REM DELE
TE THE POINT BY MOVING  DATA DOWN THE BUFFER
1000 REM MAKE A BLANK PICTURE
1010 FORI=PO TO PO+9*25 STEP 9:POKEI,128:NEXT:REM MARK ALL POINT
S AS UNDEFINED
1020 POKELI,255:REM CLEAR LINE BUFFER
1030 RETURN
1200 REM LOAD A PICTURE FROM TAPE
1210 I=INSTR(A$,QU$):IFI<>0THENJ=INSTR(I+1,A$,QU$):IFJ<>0THENNA$
=MID$(A$,I,J-I+1)ELSENA$=RIGHT$(A$,LEN(A$)-I)ELSENA$=""
1220 IFLEN(NA$)>8THENNA$=LEFT$(NA$,8)
1230 CLOADM NA$:GOTO1800:REM DISPLAY PICTURE AFTER LOADING
1400 REM SAVE CURRENT PICTURE ON TAPE
1410 I=INSTR(A$,QU$):IFI<>0THENJ=INSTR(I+1,A$,QU$):IFJ<>0THENNA$
=MID$(A$,I,J-I+1)ELSENA$=RIGHT$(A$,LEN(A$)-I)ELSENA$=""
1420 IFLEN(NA$)>8THENNA$=LEFT$(NA$,8)
1430 CSAVEM NA$,PO,MA-1,PO:RETURN
1600 REM HELP ROUTINE
1610 CLS:PRINT"DISPLAY PICTURE:  display"
1620 PRINT"ERASE PICTURE:  new"
1630 PRINT"SAVE PIX ON TAPE:  save";QU$;"NAME";QU$
1640 PRINT"READ PIX FROM TAPE:  load";QU$;"NAME";QU$
1650 PRINT"ENTER POINT P:  P = X,Y,Z"
1660 PRINT"ENTER LINE SEGMENT AB:  AB"
1670 PRINT"DELETE LINE SEGMENT CD:  CD#"
1680 PRINT"PRINT POINTS A TO H:  ?A-H"
1690 PRINT"PRINT LINE SEGMENTS:  ??"
1700 PRINT:RETURN
1800 REM DISPLAY THE PICTURE
1810 REM FIRST DECLARE ALL UNUSED POINTS AS UNDEFINED
1820 FORI=0TO25:CH(I)=0:NEXT
1830 I=LI
1840 P=PEEK(I):IFP<>255THENCH(P/9)=1:I=I+1:GOTO1840:REM MARK ALL
 POINT NAMES USED
1850 FORI=0TO25:IFCH(I)=0THENPOKEPO+9*I,128:NEXT:REM MARK UNUSED
 POINTS AS UNDEFINED
1860 POKE 282,0:EXEC MA:POKE282,255:PRINT:RETURN
2000 REM INPUT A POINT
2010 A=ASC(LEFT$(A$,1))-AA:IFA<0ORA>25THENGOSUB 2600:RETURN
2020 AD=PO+9*A
2030 A$=RIGHT$(A$,LEN(A$)-2)
2040 IFA$=""THENGOSUB2600:RETURN
2050 VA=VAL(A$):IFABS(VA)>80THENPRINT:PRINT"COORDINATES MUST BE
BETWEEN -80 AND 80":PRINT:RETURNELSEB=LEN(A$)-LEN(STR$(VA)):IFVA
<0THENB=B-1
2055 IFB<3THENGOSUB2600:GOTO190ELSEA$=RIGHT$(A$,B):IFVA>=0THENV1
=0:V2=VA ELSEV1=255:V2=256+VA
2060 POKE AD,V1:POKEAD+1,V2:POKEAD+2,0:AD=AD+3
2070 IFA$=""THENGOSUB2600:RETURN
2080 VA=VAL(A$):IFABS(VA)>80THENPRINT:PRINT"COORDINATES MUST BE
BETWEEN -80 AND 80":PRINT:RETURNELSEB=LEN(A$)-LEN(STR$(VA)):IFVA
<0THENB=B-1
2085 IFB<1THENGOSUB2600:GOTO190ELSEA$=RIGHT$(A$,B):IFVA>=0THENV1
=0:V2=VA ELSE V1=255:V2=256+VA
2090 POKE AD,V1:POKEAD+1,V2:POKEAD+2,0:AD=AD+3
2140 VA=VAL(A$):IFABS(VA)>80 THEN PRINT:PRINT"COORDINATES MUST B
E BETWEEN -80 AND 80":PRINT:RETURN
2145 IFVA>=0THENV1=0:V2=VA ELSEV1=255:V2=256+VA
2150 POKE AD,V1:POKEAD+1,V2:POKEAD+2,0:RETURN
2200 REM PRINT LINES
2210 I=LI
2220 P=PEEK(I):IF P=255 THEN PRINT:RETURN
2230 PRINTCHR$(P/9+AA);CHR$(PEEK(I+1)/9+AA);" ";:I=I+2:GOTO2220
2400 REM PRINT POINTS
2410 A1=0:A2=25:I=INSTR(A$,"-"):IFI=3THENA1=ASC(MID$(A$,2,1))-AA
:IFA1<0ORA1>25THENA1=0:GOTO2030
2420 IFI<LEN(A$)ANDI<=3THENA2=ASC(MID$(A$,I+1,1))-AA:IFA2<0ORA2>
25THENA2=25
2430 FORI=A1 TO A2:AD=PO+9*I:IFPEEK(AD)=128THENNEXT:RETURN
2440 PRINTCHR$(I+AA);" = ";:FORJ=0TO6STEP3
```

Listing continued

Listing continued

```
2450 VA=PEEK(AD+J)*256+PEEK(AD+J+1)+PEEK(AD+J+2)/256:IFVA>32767T
HENVA=VA-65536
2460 V$=STR$(VA):PRINTV$;:IFJ<>6THENPRINT",";
2465 NEXT:PRINT:NEXT:RETURN
2600 PRINT:PRINT"UNRECOGNIZED COMMAND":PRINT:RETURN
```

Program Listing 2. Assembly Portion of Display

```
                    00100 *
                    00110 * PROGRAM <<DISPLAY>>
                    00120 * PAN, SCALE, AND ROTATE CONTENTS OF *POINTS* THEN DISP
LAY THE CONTENTS OF *LINES*
                    00130 *
7000                00140         ORG     $7000
                    00150 *
                    00160 * BASIC DATA STRUCTURES--THE POINTS AND LINES TO BE DRA
WN.  POINTS MAINTAINS THE COORDINATES OF THE POINTS
                    00170 *
7000                00180 POINTS  RMB     9*26    STORE THREE COORDS OF THREE BYT
ES EACH LABELED A..Z FOR THE USER
                    00190 * UNUSED POINTS HAVE $80 IN THE FIRST BYTE OF EACH COOR
DINATE
70EA                00200 LINES   RMB     121     UP TO 30 PAIRS OF POINTS, LAST
PAIR FOLLOWED BY -1, POINT L REPRESENTED BY 9*(ASC(L)-ASC(A))
                    00210 * INTRODUCTORY HOUSEKEEPING TO DISABLE BASIC INTERRUPTS
 AND CREATE NEW STACK THEN RESTORE ENVIRONMENT FOR RETURN TO BASIC
                    00220 *
7163 1A   10        00230         ORCC    #$10    DISABLE REGULAR INTERRUPT
7165 33   E4        00240         LEAU    ,S
7167 10CE 8000      00250         LDS     #$8000  USE HIGH MEMORY FOR HARDWARE ST
ACK
716B 34   40        00260         PSHS    U
716D BD   7177      00270         JSR     MAIN
7170 35   40        00280         PULS    U
7172 32   C4        00290         LEAS    ,U      RESTORE HARDWARE STACK
7174 1C   EF        00300         ANDCC   #$EF    ENABLE REGULAR INTERRUPT
7176 39             00310         RTS
                    00320 *
                    00330 * MAIN DISPLAY LOOP--LOOKS FOR A KEY COMMANDING ROTATIO
N, PANNING, OR SCALING AND ORDERS SSAME.  INCLUDES AUTO REPEAT
                    00340 *
7177 B7   FFC0      00350 MAIN    STA     $FFC0   SET VDG = 110
717A B7   FFC3      00360         STA     $FFC3
717D B7   FFC5      00370         STA     $FFC5
7180 B6   FF22      00380         LDA     $FF22   SET HI BITS CONTROL REGISTER =
$F0
7183 84   07        00390         ANDA    #7
7185 8A   F0        00400         ORA     #$F0
7187 B7   FF22      00410         STA     $FF22
718A BD   7221      00420         JSR     DISPLA  DISPLAY THE PICTURE
718D AD   9F A000   00430 KEY     JSR     [$A000]
7191 27   FA        00440         BEQ     KEY
7193 BD   7731      00445         JSR     KEYVAL
7196 81   08        00450         CMPA    #$8     LEFT ARROW
7198 26   05        00460         BNE     MAIN1
719A 8E   72AF      00470         LDX     #PANL
719D 20   68        00480         BRA     REPEAT
719F 81   09        00490 MAIN1   CMPA    #$9     RIGHT ARROW
71A1 26   05        00500         BNE     MAIN2
71A3 8E   72E5      00510         LDX     #PANR
71A6 20   5F        00520         BRA     REPEAT
71A8 81   5E        00530 MAIN2   CMPA    #$5E    UP ARROW
71AA 26   05        00540         BNE     MAIN3
71AC 8E   7351      00550         LDX     #PANU
71AF 20   56        00560         BRA     REPEAT
71B1 81   0A        00570 MAIN3   CMPA    #$0A    DOWN ARROW
71B3 26   05        00580         BNE     MAIN4
71B5 8E   731B      00590         LDX     #PAND
71B8 20   4D        00600         BRA     REPEAT
71BA 81   78        00610 MAIN4   CMPA    #$78    UNSHIFTED X
71BC 26   05        00620         BNE     MAIN5
71BE 8E   7387      00630         LDX     #ROTX
71C1 20   44        00640         BRA     REPEAT
71C3 81   58        00650 MAIN5   CMPA    #$58    SHIFTED X
71C5 26   05        00660         BNE     MAIN6
71C7 8E   73D0      00670         LDX     #ROTMX
71CA 20   3B        00680         BRA     REPEAT
71CC 81   79        00690 MAIN6   CMPA    #$79    UNSHIFTED Y
71CE 26   05        00700         BNE     MAIN7
71D0 8E   7419      00710         LDX     #ROTY
71D3 20   32        00720         BRA     REPEAT
71D5 81   59        00730 MAIN7   CMPA    #$59    SHIFTED Y
71D7 26   05        00740         BNE     MAIN8
71D9 8E   7460      00750         LDX     #ROTMY
71DC 20   29        00760         BRA     REPEAT
71DE 81   7A        00770 MAIN8   CMPA    #$7A    UNSHIFTED Z
71E0 26   05        00780         BNE     MAIN9
71E2 8E   74A7      00790         LDX     #ROTZ
71E5 20   20        00800         BRA     REPEAT
71E7 81   5A        00810 MAIN9   CMPA    #$5A    SHIFTED Z
71E9 26   05        00820         BNE     MAIN10
71EB 8E   74EE      00830         LDX     #ROTMZ
71EE 20   17        00840         BRA     REPEAT
71F0 81   62        00850 MAIN10  CMPA    #$62    UNSHIFTED B
71F2 26   05        00860         BNE     MAIN11
71F4 8E   7618      00870         LDX     #BIGGER
71F7 20   0E        00880         BRA     REPEAT
71F9 81   73        00890 MAIN11  CMPA    #$73    UNSHIFTED S
71FB 26   05        00900         BNE     MAIN12
71FD 8E   76AB      00910         LDX     #SMALLR
7200 20   05        00920         BRA     REPEAT
7202 81   40        00930 MAIN12  CMPA    #$40    @
7204 26   87        00940         BNE     KEY     NO COMMAND RECOGNIZED
7206 39             00950         RTS
7207 34   14        00960 REPEAT  PSHS    X,B     SAVE ADDRESS OF SUB CALLED BY L
AST KEY AND SUM OF KEY BUFFERS
7209 AD   F8 01     00970 REP1    JSR     [1,S]   1,S POINTS TO ADDRESS OF SUBROU
TINE CALLED BY LAST KEY PRESSED
720C BD   7221      00980         JSR     DISPLA  DISPLAY UPDATED POINTS
720F AD   9F A000   00990         JSR     [$A000]
7213 BD   7731      01000         JSR     KEYVAL
7216 E1   E4        01010         CMPB    ,S
7218 27   EF        01090         BEQ     REP1    KEY STILL DEPRESSED
721A 32   63        01100         LEAS    3,S     RESTORE STACK
721C 16   FF6E      01110         LBRA    KEY     LOOK FOR ANOTHER COMMAND
                    01120 *
                    01130 * CREATE A PICTURE USING CURRENT POINTS AND LINES ON TH
E UNSEEN GRAPHICS SCREEN THEN DISPLAY THAT SCREEN
                    01140 *
721F      0600      01150 TL      FDB     $600    CONTAINS THE ADDRESS OF THE FIR
ST BYTE OF THE PRESENTLY UNSEEN GRAPHICS SCREEN--INITIALIZED SO
                    01160 *
                    01170 * PLOT THE CURRENT LINES ON THE UNSEEN GRAPHICS SCREEN
THEN DISPLAY THE SCREEN
                    01180 *
7221 4F             01190 DISPLA  CLRA    BLANK SCREEN
7222 C6   0C        01200         LDB     #$C
7224 108E 0000      01210         LDY     #0
7228 BE   721F      01220         LDX     TL
```

Listing continued

```
722B 10AF 81      01230 DISP1   STY     ,X++
722E 4A           01240         DECA
722F 26 FA        01250         BNE     DISP1
7231 5A           01260         DECB
7232 26 F7        01270         BNE     DISP1
                  01280 *
                  01290 * PLOT EACH LINE
                  01300 *
7234 CE 70EA      01310         LDU     #LINES
7237 E6 C0        01320 DISP3   LDB     ,U+
7239 C1 FF        01330         CMPB    #$FF
723B 27 2F        01340         BEQ     DISP2   LAST POINT?
723D 4F           01350         CLRA
723E              01360 P1300
723E C3 7000      01370         ADDD    #POINTS
7241 1F 02        01380         TFR     D,Y     ADDRESS OF FIRST POINT
7243 AE A4        01390         LDX     ,Y
7245 BF 7752      01400         STX     X1
7248 31 23        01410         LEAY    3,Y
724A AE A4        01420         LDX     ,Y
724C BF 7754      01430         STX     Y1
724F E6 C0        01440         LDB     ,U+
7251 4F           01450         CLRA
7252 C3 7000      01460         ADDD    #POINTS
7255 1F 02        01470         TFR     D,Y     ADDRESS OF SECOND POINT
7257 AE A4        01480         LDX     ,Y
7259 BF 7756      01490         STX     X2
725C 31 23        01500         LEAY    3,Y
725E AE A4        01510         LDX     ,Y
7260 BF 7758      01520         STX     Y2
7263 34 40        01530         PSHS    U
7265 BD 775B      01540         JSR     LINE
7268 35 40        01550         PULS    U
726A 20 CB        01560         BRA     DISP3
                  01570 *
                  01580 * SWITCH SCREENS TO SHOW NEW PICTURE
                  01590 *
726C BE 721F      01600 DISP2   LDX     TL
726F 8C 0600      01610         CMPX    #$600
7272 27 1F        01620         BEQ     DISP4
7274 B7 FFC7      01630         STA     $FFC7   VIDEO OFFSET=$1E00/$200=$F
7277 B7 FFC9      01640         STA     $FFC9
727A B7 FFCB      01650         STA     $FFCB
727D B7 FFCD      01660         STA     $FFCD
7280 B7 FFCE      01670         STA     $FFCE
7283 B7 FFD0      01680         STA     $FFD0
7286 B7 FFD2      01690         STA     $FFD2
7289 B7 FFD4      01700         STA     $FFD4
728C 8E 0600      01710         LDX     #$600
728F BF 721F      01720         STX     TL      SET TL = FIRST BYTE OF UNUSED S
CREEN
7292 39           01730         RTS
7293 B7 FFC7      01740 DISP4   STA     $FFC7   SET VIDEO OFFSET = $600/$200=$3
7296 B7 FFC9      01750         STA     $FFC9
7299 B7 FFCA      01760         STA     $FFCA
729C B7 FFCC      01770         STA     $FFCC
729F B7 FFCE      01780         STA     $FFCE
72A2 B7 FFD0      01790         STA     $FFD0
72A5 B7 FFD2      01800         STA     $FFD2
72A8 8E 1E00      01810         LDX     #$1E00
72AB BF 721F      01820         STX     TL      FIRST BYTE OF UNSEEN VIDEO SCRE
EN
72AE 39           01830         RTS
                  01840 *
                  01850 * ROUTINES TO MOVE POINTS
                  01860 *
                  01870 *
                  01880 * PAN LEFT
                  01890 *
          70E1    01900 LASTPT  SET     POINTS+$9*$19
72AF 8E 7000      01910 PANL    LDX     #POINTS
72B2 A6 84        01920 PANL2   LDA     ,X
72B4 81 80        01930         CMPA    #$80
72B6 27 0B        01940         BEQ     PANL1
72B8 EC 84        01950         LDD     ,X      GET X COORDINATE AND SUBTRACT 2
72BA 83 0002      01960         SUBD    #2
72BD ED 84        01970         STD     ,X
72BF 81 80        01980         CMPA    #-$80
72C1 2F 08        01990         BLE     PANL4   POINT OUT OF BOUNDS
72C3 30 09        02050 PANL1   LEAX    9,X
72C5 8C 70E1      02060         CMPX    #LASTPT
72C8 23 E8        02070         BLS     PANL2
72CA 39           02080         RTS
72CB A6 84        02090 PANL4   LDA     ,X      SINCE ONE POINT OUT OF BOUNDS,
RESTORE ALL POINTS TO ORIGINAL VALUES
72CD 81 80        02100         CMPA    #$80
72CF 27 0C        02110         BEQ     PANL5
72D1 A6 01        02120         LDA     1,X
72D3 8B 02        02130         ADDA    #2
72D5 A7 01        02140         STA     1,X
72D7 A6 84        02150         LDA     ,X
72D9 89 00        02160         ADCA    #0
72DB A7 84        02170         STA     ,X
72DD 30 17        02180 PANL5   LEAX    -9,X
72DF 8C 7000      02190         CMPX    #POINTS
72E2 24 E7        02200         BHS     PANL4
72E4 39           02210         RTS
                  02220 *
                  02230 * PAN RIGHT
                  02240 *
72E5 8E 7000      02250 PANR    LDX     #POINTS
72E8 A6 84        02260 PANR2   LDA     ,X
72EA 81 80        02270         CMPA    #$80
72EC 27 0B        02280         BEQ     PANR1   IF POINT UNDEFINED
72EE EC 84        02290         LDD     ,X      ADD 2 TO X COORDINATE
72F0 C3 0002      02300         ADDD    #2
72F3 ED 84        02310         STD     ,X
72F5 81 10        02320         CMPA    #$10
72F7 2C 08        02330         BGE     PANR4   POINT OUT OF BOUNDS
72F9 30 09        02390 PANR1   LEAX    9,X
72FB 8C 70E1      02400         CMPX    #LASTPT
72FE 23 E8        02410         BLS     PANR2
7300 39           02420         RTS
7301 A6 84        02430 PANR4   LDA     ,X      SINCE ONE POINT OUT OF BOUNDS,
RESTORE ALL POINTS  TO ORIGINAL VALUES
7303 81 80        02440         CMPA    #$80
7305 27 0C        02450         BEQ     PANR5
7307 A6 01        02460         LDA     1,X
7309 80 02        02470         SUBA    #2
730B A7 01        02480         STA     1,X
730D A6 84        02490         LDA     ,X
730F 82 00        02500         SBCA    #0
7311 A7 84        02510         STA     ,X
7313 30 17        02520 PANR5   LEAX    -9,X
7315 8C 7000      02530         CMPX    #POINTS
7318 24 E7        02540         BHS     PANR4
731A 39           02550         RTS
                  02560 *
                  02570 * PAN DOWN
                  02580 *
731B 8E 7000      02590 PAND    LDX     #POINTS
731E A6 84        02600 PAND2   LDA     ,X
7320 81 80        02610         CMPA    #$80
```

Listing continued

Listing continued

```
7322 27   0B       02620          BEQ     PAND1   IF POINT UNDEFINED
7324 EC   03       02630          LDD     3,X     SUBTRACT 2 FROM Y COORDINATE
7326 83   0002     02640          SUBD    #2
7329 ED   03       02650          STD     3,X
732B 81   F0       02660          CMPA    #-$10
732D 2F   08       02670          BLE     PAND4   IF POINT OUT OF BOUNDS
732F 30   09       02730 PAND1    LEAX    9,X
7331 8C   70E1     02740          CMPX    #LASTPT
7334 23   E8       02750          BLS     PAND2
7336 39            02760          RTS
7337 A6   84       02770 PAND4    LDA     ,X      SINCE ONE POINT OUT OF BOUNDS,
RESTORE ALL POINTS  TO ORIGINAL VALUES
7339 81   80       02780          CMPA    #$80
733B 27   0C       02790          BEQ     PAND5
733D A6   04       02800          LDA     4,X
733F 8B   02       02810          ADDA    #2
7341 A7   04       02820          STA     4,X
7343 A6   03       02830          LDA     3,X
7345 89   00       02840          ADCA    #0
7347 A7   03       02850          STA     3,X
7349 30   17       02860 PAND5    LEAX    -9,X
734B 8C   7000     02870          CMPX    #POINTS
734E 24   E7       02880          BHS     PAND4
7350 39            02890          RTS
                   02900 *
                   02910 * PAN UP
                   02920 *
7351 8E   7000     02930 PANU     LDX     #POINTS
7354 A6   84       02940 PANU2    LDA     ,X
7356 81   80       02950          CMPA    #$80
7358 27   0B       02960          BEQ     PANU1   IF POINT UNDEFINED
735A EC   03       02970          LDD     3,X     ADD 2 TO Y COORDINATE
735C C3   0002     02980          ADDD    #2
735F ED   03       02990          STD     3,X
7361 81   10       03000          CMPA    #$10
7363 2C   08       03010          BGE     PANU4   POINT OUT OF BOUNDS
7365 30   09       03070 PANU1    LEAX    9,X
7367 8C   70E1     03080          CMPX    #LASTPT
736A 23   E8       03090          BLS     PANU2
736C 39            03100          RTS
736D A6   84       03110 PANU4    LDA     ,X      SINCE ONE POINT OUT OF BOUNDS,
RESTORE ALL POINTS  TO ORIGINAL VALUES
736F 81   80       03120          CMPA    #$80
7371 27   0C       03130          BEQ     PANU5
7373 A6   04       03140          LDA     4,X
7375 80   02       03150          SUBA    #2
7377 A7   04       03160          STA     4,X
7379 A6   03       03170          LDA     3,X
737B 82   00       03180          SBCA    #0
737D A7   03       03190          STA     3,X
737F 30   17       03200 PANU5    LEAX    -9,X
7381 8C   7000     03210          CMPX    #POINTS
7384 24   E7       03220          BHS     PANU4
7386 39            03230          RTS
                   03240 *
                   03250 * ROTATE POSITIVELY ABOUT X AXIS
                   03260 *
7387 8E   7000     03270 ROTX     LDX     #POINTS
738A A6   84       03280 ROTX1    LDA     ,X
738C 81   80       03290          CMPA    #$80    CHECK FOR UNDEFINED POINT
738E 27   1F       03300          BEQ     ROTX2
7390 34   10       03310          PSHS    X
7392 30   03       03320          LEAX    3,X     ADDRESS OF Y-COORDINATE OF CURR
ENT POINT
7394 31   03       03330          LEAY    3,X     ADDRESS OF Z-COORDINATE OF CURR
ENT POINT
7396 BD   7535     03340          JSR     ROTATE
7399 35   10       03350          PULS    X
739B A6   03       03360          LDA     3,X
739D 81   10       03370          CMPA    #$10
739F 2C   16       03380          BGE     ROTX3
73A1 81   F0       03390          CMPA    #-$10
73A3 2F   12       03400          BLE     ROTX3   POINT TOO BIG
73A5 A6   06       03410          LDA     6,X
73A7 81   10       03420          CMPA    #$10
73A9 2C   0C       03430          BGE     ROTX3
73AB 81   F0       03440          CMPA    #-$10
73AD 2F   08       03450          BLE     ROTX3
73AF 30   09       03460 ROTX2    LEAX    9,X     GET NEXT POINT
73B1 8C   70E1     03470          CMPX    #LASTPT
73B4 2F   D4       03480          BLE     ROTX1
73B6 39            03490          RTS
73B7 34   10       03500 ROTX3    PSHS    X       UNDO ROTATIONS AS  ONE POINT TO
O LARGE
73B9 A6   84       03510          LDA     ,X
73BB 81   80       03520          CMPA    #$80
73BD 27   07       03530          BEQ     RX
73BF 31   03       03540          LEAY    3,X
73C1 30   06       03550          LEAX    6,X
73C3 BD   7535     03560          JSR     ROTATE
73C6 35   10       03570 RX       PULS    X
73C8 30   17       03580          LEAX    -9,X
73CA 8C   7000     03590          CMPX    #POINTS
73CD 24   E8       03600          BHS     ROTX3
73CF 39            03610          RTS
                   03620 *
                   03630 * ROTATE NEGATIVELY ABOUT X AXIS
                   03640 *
73D0 8E   7000     03650 ROTMX    LDX     #POINTS
73D3 A6   84       03660 ROTMX1   LDA     ,X
73D5 81   80       03670          CMPA    #$80    CHECK FOR UNDEFINED POINT
73D7 27   1F       03680          BEQ     ROTMX2
73D9 34   10       03690          PSHS    X
73DB 31   03       03700          LEAY    3,X     ADDRESS OF Y-COORDINATE OF CURR
ENT POINT
73DD 30   06       03710          LEAX    6,X     ADDRESS OF Z-COORDINATE OF CURR
ENT POINT
73DF BD   7535     03720          JSR     ROTATE
73E2 35   10       03730          PULS    X
73E4 A6   03       03740          LDA     3,X
73E6 81   10       03750          CMPA    #$10
73E8 2C   16       03760          BGE     ROTMX3
73EA 81   F0       03770          CMPA    #-$10
73EC 2F   12       03780          BLE     ROTMX3  POINT TOO BIG
73EE A6   06       03790          LDA     6,X
73F0 81   10       03800          CMPA    #$10
73F2 2C   0C       03810          BGE     ROTMX3
73F4 81   F0       03820          CMPA    #-$10
73F6 2F   08       03830          BLE     ROTMX3
73F8 30   09       03840 ROTMX2   LEAX    9,X     GET NEXT POINT
73FA 8C   70E1     03850          CMPX    #LASTPT
73FD 2F   D4       03860          BLE     ROTMX1
73FF 39            03870          RTS
7400 34   10       03880 ROTMX3   PSHS    X       UNDO ROTATIONS AS  ONE POINT TO
O LARGE
7402 A6   84       03890          LDA     ,X
7404 81   80       03900          CMPA    #$80
7406 27   07       03910          BEQ     RMX
7408 30   03       03920          LEAX    3,X
740A 31   03       03930          LEAY    3,X
740C BD   7535     03940          JSR     ROTATE
740F 35   10       03950 RMX      PULS    X
```

Listing continued

Listing continued

```
7411 30   17     03960          LEAX   -9,X
7413 8C   7000   03970          CMPX   #POINTS
7416 24   E8     03980          BHS    ROTMX3
7418 39          03990          RTS
                 04000 *
                 04010 * ROTATE POSITIVELY ABOUT Y AXIS
                 04020 *
7419 8E   7000   04030 ROTY     LDX    #POINTS
741C A6   84     04040 ROTY1    LDA    ,X
741E 81   80     04050          CMPA   #$80    CHECK FOR UNDEFINED POINT
7420 27   1F     04060          BEQ    ROTY2
7422 34   10     04070          PSHS   X
7424 31   84     04080          LEAY   ,X      ADDRESS OF X-COORDINATE OF CURR
ENT POINT
7426 30   06     04090          LEAX   6,X     ADDRESS OF Z-COORDINATE OF CURR
ENT POINT
7428 BD   7535   04100          JSR    ROTATE
742B 35   10     04110          PULS   X
742D A6   84     04120          LDA    ,X
742F 81   10     04130          CMPA   #$10
7431 2C   16     04140          BGE    ROTY3
7433 81   F0     04150          CMPA   #-$10
7435 2F   12     04160          BLE    ROTY3   POINT TOO BIG
7437 A6   06     04170          LDA    6,X
7439 81   10     04180          CMPA   #$10
743B 2C   0C     04190          BGE    ROTY3
743D 81   F0     04200          CMPA   #-$10
743F 2F   08     04210          BLE    ROTY3
7441 30   09     04220 ROTY2    LEAX   9,X     GET NEXT POINT
7443 8C   70E1   04230          CMPX   #LASTPT
7446 2F   D4     04240          BLE    ROTY1
7448 39          04250          RTS
7449 34   10     04260 ROTY3    PSHS   X       UNDO ROTATIONS AS  ONE POINT TO
O LARGE
744B A6   84     04270          LDA    ,X
744D 81   80     04280          CMPA   #$80
744F 27   05     04290          BEQ    RY
7451 31   06     04300          LEAY   6,X
7453 BD   7535   04310          JSR    ROTATE
7456 35   10     04320 RY       PULS   X
7458 30   17     04330          LEAX   -9,X
745A 8C   7000   04340          CMPX   #POINTS
745D 24   EA     04350          BHS    ROTY3
745F 39          04360          RTS
                 04370 *
                 04380 * ROTATE NEGATIVELY ABOUT Y AXIS
                 04390 *
7460 8E   7000   04400 ROTMY    LDX    #POINTS
7463 A6   84     04410 ROTMY1   LDA    ,X
7465 81   80     04420          CMPA   #$80    CHECK FOR UNDEFINED POINT
7467 27   1D     04430          BEQ    ROTMY2
7469 34   10     04440          PSHS   X
746B 31   06     04450          LEAY   6,X     ADDRESS OF Z-COORDINATE OF CURR
ENT POINT--X CONTAINS X-COORD
746D BD   7535   04460          JSR    ROTATE
7470 35   10     04470          PULS   X
7472 A6   84     04480          LDA    ,X
7474 81   10     04490          CMPA   #$10
7476 2C   16     04500          BGE    ROTMY3
7478 81   F0     04510          CMPA   #-$10
747A 2F   12     04520          BLE    ROTMY3  POINT TOO BIG
747C A6   06     04530          LDA    6,X
747E 81   10     04540          CMPA   #$10
7480 2C   0C     04550          BGE    ROTMY3
7482 81   F0     04560          CMPA   #-$10
7484 2F   08     04570          BLE    ROTMY3
7486 30   09     04580 ROTMY2   LEAX   9,X     GET NEXT POINT
7488 8C   70E1   04590          CMPX   #LASTPT
748B 2F   D6     04600          BLE    ROTMY1
748D 39          04610          RTS
748E 34   10     04620 ROTMY3   PSHS   X       UNDO ROTATIONS AS  ONE POINT TO
O LARGE
7490 A6   84     04630          LDA    ,X
7492 81   80     04640          CMPA   #$80
7494 27   07     04650          BEQ    RMY
7496 31   84     04660          LEAY   ,X
7498 30   06     04670          LEAX   6,X
749A BD   7535   04680          JSR    ROTATE
749D 35   10     04690 RMY      PULS   X
749F 30   17     04700          LEAX   -9,X
74A1 8C   7000   04710          CMPX   #POINTS
74A4 24   E8     04720          BHS    ROTMY3
74A6 39          04730          RTS
                 04740 *
                 04750 * ROTATE POSITIVELY ABOUT Z AXIS
                 04760 *
74A7 8E   7000   04770 ROTZ     LDX    #POINTS
74AA A6   84     04780 ROTZ1    LDA    ,X
74AC 81   80     04790          CMPA   #$80    CHECK FOR UNDEFINED POINT
74AE 27   1D     04800          BEQ    ROTZ2
74B0 34   10     04810          PSHS   X
74B2 31   03     04820          LEAY   3,X     ADDRESS OF Y-COORDINATE OF CURR
ENT POINT--X HAS X-COORDINATE
74B4 BD   7535   04830          JSR    ROTATE
74B7 35   10     04840          PULS   X
74B9 A6   84     04850          LDA    ,X
74BB 81   10     04860          CMPA   #$10
74BD 2C   16     04870          BGE    ROTZ3
74BF 81   F0     04880          CMPA   #-$10
74C1 2F   12     04890          BLE    ROTZ3   POINT TOO BIG
74C3 A6   03     04900          LDA    3,X
74C5 81   10     04910          CMPA   #$10
74C7 2C   0C     04920          BGE    ROTZ3
74C9 81   F0     04930          CMPA   #-$10
74CB 2F   08     04940          BLE    ROTZ3
74CD 30   09     04950 ROTZ2    LEAX   9,X     GET NEXT POINT
74CF 8C   70E1   04960          CMPX   #LASTPT
74D2 2F   D6     04970          BLE    ROTZ1
74D4 39          04980          RTS
74D5 34   10     04990 ROTZ3    PSHS   X       UNDO ROTATIONS AS  ONE POINT TO
O LARGE
74D7 A6   84     05000          LDA    ,X
74D9 81   80     05010          CMPA   #$80
74DB 27   07     05020          BEQ    RZ
74DD 31   84     05030          LEAY   ,X
74DF 30   03     05040          LEAX   3,X
74E1 BD   7535   05050          JSR    ROTATE
74E4 35   10     05060 RZ       PULS   X
74E6 30   17     05070          LEAX   -9,X
74E8 8C   7000   05080          CMPX   #POINTS
74EB 24   E8     05090          BHS    ROTZ3
74ED 39          05100          RTS
                 05110 *
                 05120 * ROTATE NEGATIVELY ABOUT Z AXIS
                 05130 *
74EE 8E   7000   05140 ROTMZ    LDX    #POINTS
74F1 A6   84     05150 ROTMZ1   LDA    ,X
74F3 81   80     05160          CMPA   #$80    CHECK FOR UNDEFINED POINT
74F5 27   1F     05170          BEQ    ROTMZ2
74F7 34   10     05180          PSHS   X
74F9 31   84     05190          LEAY   ,X      ADDRESS OF X-COORDINATE OF CURR
ENT POINT
74FB 30   03     05200          LEAX   3,X     ADDRESS OF Y-COORDINATE OF CURR
ENT POINT
```

Listing continued

Listing continued

```
74FD BD  7535    05210          JSR    ROTATE
7500 35  10      05220          PULS   X
7502 A6  84      05230          LDA    ,X
7504 81  10      05240          CMPA   #$10
7506 2C  16      05250          BGE    ROTMZ3
7508 81  F0      05260          CMPA   #-$10
750A 2F  12      05270          BLE    ROTMZ3  POINT TOO BIG
750C A6  03      05280          LDA    3,X
750E 81  10      05290          CMPA   #$10
7510 2C  0C      05300          BGE    ROTMZ3
7512 81  F0      05310          CMPA   #-$10
7514 2F  08      05320          BLE    ROTMZ3
7516 30  09      05330 ROTMZ2   LEAX   9,X      GET NEXT POINT
7518 8C  70E1    05340          CMPX   #LASTPT
751B 2F  D4      05350          BLE    ROTMZ1
751D 39          05360          RTS
751E 34  10      05370 ROTMZ3   PSHS   X        UNDO ROTATIONS AS  ONE POINT TO
O LARGE
7520 A6  84      05380          LDA    ,X
7522 81  80      05390          CMPA   #$80
7524 27  07      05400          BEQ    RMZ
7526 31  03      05410          LEAY   3,X
7528 BD  7535    05420          JSR    ROTATE
752B 35  10      05430          PULS   X
752D 30  17      05440 RMZ      LEAX   -9,X
752F 8C  7000    05450          CMPX   #POINTS
7532 24  EA      05460          BHS    ROTMZ3
7534 39          05470          RTS
                 05480 *
                 05490 * ROTATE AROUND SOME AXIS AS FOLLOWS:  X = COORD POINTE
D AT BY X-REG, Y=COORD POINTED AT BY Y-REG
                 05500 * X,Y=(127/128)X-(1/8)Y,(1/8)X+127/128)Y
                 05510 *
7535 33  7A      05520 ROTATE   LEAU   -6,S
7537 EC  84      05530          LDD    ,X
7539 ED  C4      05540          STD    ,U
753B ED  43      05550          STD    3,U
753D A6  02      05560          LDA    2,X
753F A7  42      05570          STA    2,U
7541 A7  45      05580          STA    5,U
7543 67  43      05590          ASR    3,U
7545 66  44      05600          ROR    4,U
7547 66  45      05610          ROR    5,U
7549 67  43      05620          ASR    3,U
754B 66  44      05630          ROR    4,U
754D 66  45      05640          ROR    5,U
754F 67  43      05650          ASR    3,U
7551 66  44      05660          ROR    4,U
7553 66  45      05670          ROR    5,U
7555 67  43      05680          ASR    3,U
7557 66  44      05690          ROR    4,U
7559 66  45      05700          ROR    5,U
755B 67  43      05710          ASR    3,U
755D 66  44      05720          ROR    4,U
755F 66  45      05730          ROR    5,U
7561 67  43      05740          ASR    3,U
7563 66  44      05750          ROR    4,U
7565 66  45      05760          ROR    5,U
7567 67  43      05770          ASR    3,U
7569 66  44      05780          ROR    4,U
756B 66  45      05790          ROR    5,U
756D A6  42      05800          LDA    2,U
756F A0  45      05810          SUBA   5,U
7571 A7  42      05820          STA    2,U
7573 A6  41      05830          LDA    1,U
7575 A2  44      05840          SBCA   4,U
7577 A7  41      05850          STA    1,U
7579 A6  C4      05860          LDA    ,U
757B A2  43      05870          SBCA   3,U
757D A7  C4      05880          STA    ,U
757F EC  A4      05890          LDD    ,Y
7581 ED  43      05900          STD    3,U
7583 A6  22      05910          LDA    2,Y
7585 A7  45      05920          STA    5,U
7587 67  43      05930          ASR    3,U
7589 66  44      05940          ROR    4,U
758B 66  45      05950          ROR    5,U
758D 67  43      05960          ASR    3,U
758F 66  44      05970          ROR    4,U
7591 66  45      05980          ROR    5,U
7593 67  43      05990          ASR    3,U
7595 66  44      06000          ROR    4,U
7597 66  45      06010          ROR    5,U
7599 A6  42      06020          LDA    2,U
759B A0  45      06030          SUBA   5,U
759D A7  42      06040          STA    2,U
759F A6  41      06050          LDA    1,U
75A1 A2  44      06060          SBCA   4,U
75A3 A7  41      06070          STA    1,U
75A5 A6  C4      06080          LDA    ,U
75A7 A2  43      06090          SBCA   3,U
75A9 A7  C4      06100          STA    ,U
75AB EC  C4      06110          LDD    ,U
75AD ED  84      06120          STD    ,X
75AF A6  42      06130          LDA    2,U
75B1 A7  02      06140          STA    2,X
75B3 EC  A4      06150          LDD    ,Y
75B5 ED  43      06160          STD    3,U
75B7 A6  22      06170          LDA    2,Y
75B9 A7  45      06180          STA    5,U
75BB 67  C4      06190          ASR    ,U
75BD 66  41      06200          ROR    1,U
75BF 66  42      06210          ROR    2,U
75C1 67  C4      06220          ASR    ,U
75C3 66  41      06230          ROR    1,U
75C5 66  42      06240          ROR    2,U
75C7 67  C4      06250          ASR    ,U
75C9 66  41      06260          ROR    1,U
75CB 66  42      06270          ROR    2,U
75CD EC  41      06280          LDD    1,U
75CF E3  44      06290          ADDD   4,U
75D1 ED  41      06300          STD    1,U
75D3 A6  C4      06310          LDA    ,U
75D5 A9  43      06320          ADCA   3,U
75D7 A7  C4      06330          STA    ,U
75D9 67  43      06340          ASR    3,U
75DB 66  44      06350          ROR    4,U
75DD 66  45      06360          ROR    5,U
75DF 67  43      06370          ASR    3,U
75E1 66  44      06380          ROR    4,U
75E3 66  45      06390          ROR    5,U
75E5 67  43      06400          ASR    3,U
75E7 66  44      06410          ROR    4,U
75E9 66  45      06420          ROR    5,U
75EB 67  43      06430          ASR    3,U
75ED 66  44      06440          ROR    4,U
75EF 66  45      06450          ROR    5,U
75F1 67  43      06460          ASR    3,U
75F3 66  44      06470          ROR    4,U
75F5 66  45      06480          ROR    5,U
75F7 67  43      06490          ASR    3,U
75F9 66  44      06500          ROR    4,U
```

Listing continued

Listing continued

```
75FB 66   45      06510          ROR     5,U
75FD 67   43      06520          ASR     3,U
75FF 66   44      06530          ROR     4,U
7601 66   45      06540          ROR     5,U
7603 EC   41      06550          LDD     1,U
7605 E3   44      06560          ADDD    4,U
7607 ED   41      06570          STD     1,U
7609 A6   C4      06580          LDA     ,U
760B A9   43      06590          ADCA    3,U
760D A7   C4      06600          STA     ,U
760F EC   C4      06610          LDD     ,U
7611 ED   A4      06620          STD     ,Y
7613 A6   42      06630          LDA     2,U
7615 A7   22      06640          STA     2,Y
7617 39           06650          RTS
                  06660 *
                  06670 * MAKE THE FIGURE 1/32 BIGGER
                  06680 *
7618 8E   7000    06690 BIGGER   LDX     #POINTS
761B A6   84      06700 BIG1     LDA     ,X      CHECK IF POINT DEFINED
761D 81   80      06710          CMPA    #$80
761F 27   24      06720          BEQ     BIG2
7621 BD   766C    06730          JSR     GETBIG  INCREASE ALL THREE COORDINATES
7624 30   03      06740          LEAX    3,X
7626 BD   766C    06750          JSR     GETBIG
7629 30   03      06760          LEAX    3,X
762B BD   766C    06770          JSR     GETBIG
762E 30   1A      06780          LEAX    -6,X
7630 31   84      06790          LEAY    ,X      CHEC IF ANY OF 3 COORDS TOO BIG
7632 BD   76A1    06800          JSR     TOOBIG
7635 2C   16      06810          BGE     BIG3
7637 31   23      06820          LEAY    3,Y
7639 BD   76A1    06830          JSR     TOOBIG
763C 2C   0F      06840          BGE     BIG3
763E 31   23      06850          LEAY    3,Y
7640 BD   76A1    06860          JSR     TOOBIG
7643 2C   08      06870          BGE     BIG3
7645 30   09      06880 BIG2     LEAX    9,X     GET NEXT POINT
7647 8C   70E1    06890          CMPX    #LASTPT
764A 2F   CF      06900          BLE     BIG1
764C 39           06910          RTS
764D A6   84      06920 BIG3     LDA     ,X      RESTORE POINTS IF ONE MADE OUT
OF BOUNDS
764F 81   80      06930          CMPA    #$80
7651 26   04      06940          BNE     BIG8
7653 30   17      06950          LEAX    -9,X
7655 20   0F      06960          BRA     BIG9
7657 BD   76FC    06970 BIG8     JSR     GETSML
765A 30   03      06980          LEAX    3,X
765C BD   76FC    06990          JSR     GETSML
765F 30   03      07000          LEAX    3,X
7661 BD   76FC    07010          JSR     GETSML
7664 30   11      07020          LEAX    -$F,X
7666 8C   7000    07030 BIG9     CMPX    #POINTS
7669 24   E2      07040          BHS     BIG3
766B 39           07050          RTS
766C 33   7D      07060 GETBIG   LEAU    -3,S    MAKE ONE COORDINATE 1/32 BIGGER
766E EC   84      07070          LDD     ,X
7670 ED   C4      07080          STD     ,U
7672 A6   02      07090          LDA     2,X
7674 A7   42      07100          STA     2,U
7676 67   C4      07110          ASR     ,U
7678 66   41      07120          ROR     1,U
767A 66   42      07130          ROR     2,U
767C 67   C4      07140          ASR     ,U
767E 66   41      07150          ROR     1,U
7680 66   42      07160          ROR     2,U
7682 67   C4      07170          ASR     ,U
7684 66   41      07180          ROR     1,U
7686 66   42      07190          ROR     2,U
7688 67   C4      07200          ASR     ,U
768A 66   41      07210          ROR     1,U
768C 66   42      07220          ROR     2,U
768E 67   C4      07230          ASR     ,U
7690 66   41      07240          ROR     1,U
7692 66   42      07250          ROR     2,U
7694 EC   01      07260          LDD     1,X
7696 E3   41      07270          ADDD    1,U
7698 ED   01      07280          STD     1,X
769A A6   84      07290          LDA     ,X
769C A9   C4      07300          ADCA    ,U
769E A7   84      07310          STA     ,X
76A0 39           07320          RTS
76A1 A6   A4      07330 TOOBIG   LDA     ,Y      CHECK IF A COORDINATE TOO BIG--
IF FIRST BYTE >= $80 OR <= -$80
76A3 81   10      07340          CMPA    #$10
76A5 2C   03      07350          BGE     TB1
76A7 40           07360          NEGA
76A8 81   10      07370          CMPA    #$10
76AA 39           07380 TB1      RTS             BGE WILL GO IF NUMBER LOADED IN
TO A WAS >=$10 OR <=-$10
                  07390 *
                  07400 * MAKE THE FIGURE 1/32 SMALLER
                  07410 *
76AB 8E   7000    07420 SMALLR   LDX     #POINTS
76AE A6   84      07430 SML1     LDA     ,X      CHECK IF POINT DEFINED
76B0 81   80      07440          CMPA    #$80
76B2 27   11      07450          BEQ     SML2
76B4 BD   76FC    07460          JSR     GETSML
76B7 30   03      07470          LEAX    3,X
76B9 BD   76FC    07480          JSR     GETSML
76BC 30   03      07490          LEAX    3,X
76BE BD   76FC    07500          JSR     GETSML
76C1 30   03      07510          LEAX    3,X
76C3 20   02      07520          BRA     SMLR1
76C5 30   09      07530 SML2     LEAX    9,X     GET NEXT POINT
76C7 8C   70E1    07540 SMLR1    CMPX    #LASTPT
76CA 2F   E2      07550          BLE     SML1
                  07560 * CHECK FOR LINE SEGMENT LENGTH < 1 -- IF ONE FOUND, RE
STORE POINTS TO ORIGINAL VALUES
76CC CE   70EA    07570          LDU     #LINES
76CF 4F           07580 SMLR6    CLRA
76D0 E6   C0      07590          LDB     ,U+
76D2 C1   FF      07600          CMPB    #$FF
76D4 27   25      07610          BEQ     SMLR5   LAST LINE SEGMENT TESTED
76D6 C3   7000    07620          ADDD    #POINTS
76D9 1F   01      07630          TFR     D,X     ADDRESS OF FIRST POINT
76DB 4F           07640          CLRA
76DC E6   C0      07650          LDB     ,U+
76DE C3   7000    07660          ADDD    #POINTS
76E1 1F   02      07670          TFR     D,Y     ADDRESS OF SECOND POINT
76E3 EC   84      07680          LDD     ,X
76E5 10A3 A4      07690          CMPD    ,Y      TEST FIRST COORD
76E8 26   E5      07700          BNE     SMLR6   INTEGER PARTS NOT EQUAL
76EA EC   03      07710          LDD     3,X
76EC 10A3 23      07720          CMPD    3,Y
76EF 26   DE      07730          BNE     SMLR6   INTEGER PARTS OF SECOND COORD N
OT EQUAL
76F1 EC   06      07740          LDD     6,X
76F3 10A3 26      07750          CMPD    6,Y
76F6 26   D7      07760          BNE     SMLR6   INTEGER PARTS OF THIRD COORD NO
T EQUAL
                  07770 * ALL COMPONENTS OF LINE SEGMENT HAVE EQUAL INTEGER PAR
```

Listing continued

Listing continued

```
TS, SO UNDO SMALLER AND RTS FROM BIGGER
76F8 16   FF1D     07780          LBRA    BIGGER
76FB 39            07790 SMLR5    RTS             ALL LINE  SEGMENTS CHECKED AND
LONG ENOUGH
76FC 33   7D       07800 GETSML   LEAU    -3,S
76FE EC   84       07810          LDD     ,X
7700 ED   C4       07820          STD     ,U
7702 A6   02       07830          LDA     2,X
7704 A7   42       07840          STA     2,U
7706 67   C4       07850          ASR     ,U
7708 66   41       07860          ROR     1,U
770A 66   42       07870          ROR     2,U
770C 67   C4       07880          ASR     ,U
770E 66   41       07890          ROR     1,U
7710 66   42       07900          ROR     2,U
7712 67   C4       07910          ASR     ,U
7714 66   41       07920          ROR     1,U
7716 66   42       07930          ROR     2,U
7718 67   C4       07940          ASR     ,U
771A 66   41       07950          ROR     1,U
771C 66   42       07960          ROR     2,U
771E 67   C4       07970          ASR     ,U
7720 66   41       07980          ROR     1,U
7722 66   42       07990          ROR     2,U
7724 EC   01       08000          LDD     1,X
7726 A3   41       08010          SUBD    1,U
7728 ED   01       08020          STD     1,X
772A A6   84       08030          LDA     ,X
772C A2   C4       08040          SBCA    ,U
772E A7   84       08050          STA     ,X
7730 39            08060          RTS
                   09000 *
                   09010 * SUM KEY BUFFERS TO B-REGISTER.  USED TO DETERMINE IF
KEY STILL DEPRESSED
                   09020 *
7731 F6   0152     09030 KEYVAL   LDB     $152
7734 FB   0153     09040          ADDB    $153
7737 FB   0154     09050          ADDB    $154
773A FB   0155     09060          ADDB    $155
773D FB   0156     09070          ADDB    $156
7740 FB   0157     09080          ADDB    $157
7743 FB   0158     09090          ADDB    $158
7746 FB   0159     09100          ADDB    $159
7749 39            09110          RTS
                   10000 *
                   10010 * DRAW A LINE.  X1,X2 TO Y1,Y2 - COORDINATES ARE SIGNED
 16 BIT INTEGERS, VISIBLE SCREEN IS -128 TO 127, -95 TO 95 ON USUAL
                   10020 * X,Y GRAPH
                   10030 *
774A       80      10040 BITS     FCB     $80     FOR SETTING POINTS ON GRAPHICS
SCREEN
774B       40      10050          FCB     $40
774C       20      10060          FCB     $20
774D       10      10070          FCB     $10
774E       08      10080          FCB     $8
774F       04      10090          FCB     $4
7750       02      10100          FCB     $2
7751       01      10110 LASTBT   FCB     $1
           005F    10120 TOP      EQU     $5F     LIMITS OF VIRTUAL SCREEN
           FFA0    10130 BOTTOM   EQU     -$60
           FF80    10140 LEFT     EQU     -$80
           007F    10150 RIGHT    EQU     $7F
7752               10170 X1       RMB     1       2-BYTE COORDINATES FOR TWO POIN
TS
7753               10180 X1P      RMB     1
7754               10190 Y1       RMB     1
7755               10200 Y1P      RMB     1
7756               10210 X2       RMB     1
7757               10220 X2P      RMB     1
7758               10230 Y2       RMB     1
7759               10240 Y2P      RMB     1
775A               10250 SLOPE    RMB     1
                   10260 *
                   10270 * BEGIN LINE DRAWING ALGORITHM
                   10280 *
                   10290 * MAKE SURE X1<X2 OR REVERSE COORDINATES
                   10300 *
775B FC   7756     10310 LINE     LDD     X2
775E 10B3 7752     10320          CMPD    X1
7762 2C   17       10330          BGE     LN1
7764 BE   7752     10340          LDX     X1
7767 FD   7752     10350          STD     X1
776A BF   7756     10360          STX     X2
776D BE   7754     10370          LDX     Y1
7770 10BE 7758     10380          LDY     Y2
7774 BF   7758     10390          STX     Y2
7777 10BF 7754     10400          STY     Y1
                   10410 *
                   10420 * CHECK FOR SIGN OF SLOPE
                   10430 *
777B FC   7758     10440 LN1      LDD     Y2
777E 10B3 7754     10450          CMPD    Y1
7782 102D 018E     10460          LBLT    SELINE
                   10470 *
                   10480 *BEGIN DRAWING LINE WITH POSITIVE SLOPE
                   10490 *
                   10500 * CHECK FOR NO VISIBLE LINE
                   10510 *
7786 FC   7752     10520          LDD     X1
7789 1083 007F     10530          CMPD    #RIGHT
778D 102C 0313     10540          LBGE    LNDONE
7791 FC   7754     10550          LDD     Y1
7794 1083 005F     10560          CMPD    #TOP
7798 102C 0308     10570          LBGE    LNDONE
779C FC   7756     10580          LDD     X2
779F 1083 FF80     10590          CMPD    #LEFT
77A3 102F 02FD     10600          LBLE    LNDONE
77A7 FC   7758     10610          LDD     Y2
77AA 1083 FFA0     10620          CMPD    #BOTTOM
77AE 102F 02F2     10630          LBLE    LNDONE
                   10640 *
                   10650 * CHECK IF MUST CLIP LEFT END OF LINE
                   10660 *
77B2 FC   7752     10670          LDD     X1
77B5 1083 FF80     10680          CMPD    #LEFT
77B9 2D   09       10690          BLT     LN2
77BB FC   7754     10700          LDD     Y1
77BE 1083 FFA0     10710          CMPD    #BOTTOM
77C2 2C   6F       10720          BGE     LN3
                   10730 *
                   10740 * CLIP LEFT END OF LINE
                   10750 *
77C4 BE   7756     10760 LN2      LDX     X2
77C7 10BE 7758     10770          LDY     Y2
77CB 34   30       10780          PSHS    X,Y
77CD FC   7752     10790 LN4      LDD     X1      X=(X1+X2)/2
77D0 F3   7756     10800          ADDD    X2
77D3 47            10810          ASRA
77D4 56            10820          RORB
77D5 1F   01       10830          TFR     D,X
77D7 FC   7754     10840          LDD     Y1      Y=(Y1+Y2)/2
77DA F3   7758     10850          ADDD    Y2
77DD 47            10860          ASRA
```

Listing continued

Listing continued

```
77DE 56          10870          RORB
77DF 1F   02     10880          TFR     D,Y
77E1 8C   FF80   10890          CMPX    #LEFT
77E4 2E   08     10900          BGT     LN5
77E6 108C 005F   10910          CMPY    #TOP
77EA 102C 02B4   10920          LBGE    LNDON1  NO VISIBLE LINE
77EE 8C   007F   10930 LN5      CMPX    #RIGHT
77F1 2D   08     10940          BLT     LN6
77F3 108C FFA0   10950          CMPY    #BOTTOM
77F7 102F 02A7   10960          LBLE    LNDON1  NO VISIBLE LINE
77FB 8C   FF80   10970 LN6      CMPX    #LEFT
77FE 2D   06     10980          BLT     LN7
7800 108C FFA0   10990          CMPY    #BOTTOM
7804 2C   09     11000          BGE     LN8
7806 BF   7752   11010 LN7      STX     X1      REPLACE LEFT POINT BY MIDPOINT
AND REPEAT
7809 10BF 7754   11020          STY     Y1
780D 20   BE     11030          BRA     LN4
780F 8C   FF80   11040 LN8      CMPX    #LEFT
7812 2F   0F     11050          BLE     LN9
7814 108C FFA0   11060          CMPY    #BOTTOM
7818 2F   09     11070          BLE     LN9
781A BF   7756   11080          STX     X2      REPLACE RIGHT POINT BY MIDPOINT
781D 10BF 7758   11090          STY     Y2
7821 20   AA     11100          BRA     LN4
7823 BF   7752   11110 LN9      STX     X1      X1,Y1 NOW ON LEFT OR BOTTOM BOU
NDARY
7826 10BF 7754   11120          STY     Y1
782A 35   30     11130          PULS    X,Y     RECOVER ORIGINAL RIGHT POINT
782C BF   7756   11140          STX     X2
782F 10BF 7758   11150          STY     Y2
                 11160 * CHECK IF MUST CLIP RIGHT END OF LINE
                 11170 *
7833 FC   7756   11180 LN3      LDD     X2
7836 1083 007F   11190          CMPD    #RIGHT
783A 2E   09     11200          BGT     LN10
783C FC   7758   11210          LDD     Y2
783F 1083 005F   11220          CMPD    #TOP
7843 2F   55     11230          BLE     LN18
                 11240 *
                 11250 * CLIP RIGHT END OF LINE
                 11260 *
7845 BE   7752   11270 LN10     LDX     X1
7848 10BE 7754   11280          LDY     Y1
784C 34   30     11290          PSHS    X,Y
784E FC   7752   11300 LN11     LDD     X1      X=(X1+X2)/2
7851 F3   7756   11310          ADDD    X2
7854 47          11320          ASRA
7855 56          11330          RORB
7856 1F   01     11340          TFR     D,X
7858 FC   7754   11350          LDD     Y1      Y=(Y1+Y2)/2
785B F3   7758   11360          ADDD    Y2
785E 47          11370          ASRA
785F 56          11380          RORB
7860 1F   02     11390          TFR     D,Y
7862 8C   007F   11400          CMPX    #RIGHT
7865 2E   06     11410          BGT     LN12
7867 108C 005F   11420          CMPY    #TOP
786B 2F   09     11430          BLE     LN13
786D BF   7756   11440 LN12     STX     X2      MOVE RIGHT POINT TO MIDPOINT AN
D REPEAT
7870 10BF 7758   11450          STY     Y2
7874 20   D8     11460          BRA     LN11
7876 8C   007F   11470 LN13     CMPX    #RIGHT
7879 2C   0F     11480          BGE     LN14
787B 108C 005F   11490          CMPY    #TOP
787F 2C   09     11500          BGE     LN14
7881 BF   7752   11510          STX     X1      MOVE LEFT POINT TO MIDPOINT AND
 REPEAT
7884 10BF 7754   11520          STY     Y1
7888 20   C4     11530          BRA     LN11
788A BF   7756   11540 LN14     STX     X2      MOVE X2,Y2 TO TOP OR RIGHT BOUN
DARY, RECOVER X1,Y1
788D 10BF 7758   11550          STY     Y2
7891 35   30     11560          PULS    X,Y
7893 BF   7752   11570          STX     X1
7896 10BF 7754   11580          STY     Y1
                 11590 *
                 11600 * DECIDE IF SLOPE > 1
                 11610 *
789A B6   7757   11620 LN18     LDA     X2P     X1P, ETC HOLD CORRECT ONE-BYTE
SIGNED VALUES FOR THE COORDINATES, WHICH ARE NOW ON THE VIRTUAL SCREEN
789D B0   7753   11630          SUBA    X1P
78A0 34   02     11640          PSHS    A
78A2 F6   7759   11650          LDB     Y2P
78A5 F0   7755   11660          SUBB    Y1P
78A8 E1   E0     11670          CMPB    ,S+
78AA 25   33     11680          BLO     ENELN
                 11690 *
                 11700 * DRAW A LINE WITH POSITIVE SLOPE
                 11710 *
78AC 34   04     11720          PSHS    B       SAVE NUMBER OF POINTS ON LINE -
 1
78AE BD   7AA5   11730          JSR     GETSLP  SLOPE = DX/DY, AN 8-BIT FRACTIO
N
78B1 BD   7AC0   11740          JSR     FRSTBT  SET X,U TO POINT TO BYTE AND BI
T ON GRAPHICS SCREEN CORRESP. TO X1,Y1
78B4 35   04     11750          PULS    B
78B6 4F          11760          CLRA
78B7 1F   02     11770          TFR     D,Y
78B9 31   21     11780          LEAY    1,Y     Y=#POINTS ON LINE
78BB 5F          11790          CLRB
78BC A6   84     11800 LN15     LDA     ,X      SET THE POINTS ON THE GRAPHICS
SCREEN FOR THE LINE
78BE AA   C4     11810          ORA     ,U      GET THE BYTE AND OR IN THE BIT
78C0 A7   84     11820          STA     ,X
78C2 31   3F     11830          LEAY    -1,Y    REDUCE POINT COUNT
78C4 1027 01DC   11840          LBEQ    LNDONE
78C8 30   88 E0  11850          LEAX    -$20,X  INCREASE Y-COORDINATE BY 1
78CB FB   775A   11860          ADDB    SLOPE   INCREASE X-COORDINATE BY SLOPE
(A FRACTION)
78CE 24   EC     11870          BCC     LN15    B=X MOD 1
78D0 33   41     11880          LEAU    1,U     INCREASE X-COORDINATE BY 1
78D2 1183 7751   11890          CMPU    #LASTBT TRY TO MOVE RIGHT ONE BIT
78D6 23   E4     11900          BLS     LN15
78D8 CE   774A   11910          LDU     #BITS   ELSE USE NEXT BYTE AND LEFT BIT
78DB 30   01     11920          LEAX    1,X
78DD 20   DD     11930          BRA     LN15
                 11940 *
                 11950 * DRAW LINE WITH POSITIVE SLOPE <1
                 11960 *
78DF 34   02     11970 ENELN    PSHS    A       SAVE # POINTS ON  LINE - 1
78E1 1E   89     11980          EXG     A,B
78E3 BD   7AA5   11990          JSR     GETSLP  SLOPE = DY/DX
78E6 BD   7AC0   12000          JSR     FRSTBT  X,U ARE BYTE AND BIT CORRESP. T
O X1,Y1
78E9 35   04     12010          PULS    B
78EB 4F          12020          CLRA
78EC 1F   02     12030          TFR     D,Y
78EE 31   21     12040          LEAY    1,Y     Y=#POINTS ON LINE
78F0 5F          12050          CLRB
78F1 A6   84     12060 LN16     LDA     ,X      DRAW THE LINE
```

Listing continued

```
78F3 AA   C4       12070          ORA     ,U       GET  THE  BYTE, OR IN  THE BIT
78F5 A7   84       12080          STA     ,X
78F7 31   3F       12090          LEAY    -1,Y     REDUCE THE POINT COUNT
78F9 1027 01A7     12100          LBEQ    LNDONE
78FD 33   41       12110          LEAU    1,U      MOVE ONE BIT TO RIGHT
78FF 1183 7751     12120          CMPU    #LASTBT  IF POSSIBLE
7903 23   05       12130          BLS     LN17
7905 30   01       12140          LEAX    1,X      ELSE MOVE TO NEXT BYTE AND FIRS
T BIT
7907 CE   774A     12150          LDU     #BITS
790A FB   775A     12160 LN17     ADDB    SLOPE
790D 24   E2       12170          BCC     LN16     NO OVERFLO TO NEXT INTEGER WHEN
 ADDING SLOPE TO Y
790F 30   88 E0    12180          LEAX    -$20,X   ELSE ADD 1 TO Y-COORDINATE
7912 20   DD       12190          BRA     LN16
                   12200 *
                   12210 * BEGIN DRAWING LINE WITH NEGATIVE SLOPE
                   12220 *
                   12230 * CHECK FOR NO VISIBLE LINE
                   12240 *
7914 FC   7752     12250 SELINE   LDD     X1
7917 1083 007F     12260          CMPD    #RIGHT
791B 102C 0185     12270          LBGE    LNDONE
791F FC   7754     12280          LDD     Y1
7922 1083 FFA0     12290          CMPD    #BOTTOM
7926 102F 017A     12300          LBLE    LNDONE
792A FC   7756     12310          LDD     X2
792D 1083 FF80     12320          CMPD    #LEFT
7931 102F 016F     12330          LBLE    LNDONE
7935 FC   7758     12340          LDD     Y2
7938 1083 005F     12350          CMPD    #TOP
793C 102C 0164     12360          LBGE    LNDONE
                   12370 *
                   12380 * CHECK IF MUST CLIP LEFT END OF LINE
                   12390 *
7940 FC   7752     12400          LDD     X1
7943 1083 FF80     12410          CMPD    #LEFT
7947 2D   09       12420          BLT     LN2A
7949 FC   7754     12430          LDD     Y1
794C 1083 005F     12440          CMPD    #TOP
7950 2F   6F       12450          BLE     LN3A
                   12460 *
                   12470 * CLIP 'LEFT END OF LINE FT END OF LI
                   12480 *
7952 BE   7756     12490 LN2A     LDX     X2
7955 10BE 7758     12500          LDY     Y2
7959 34   30       12510          PSHS    X,Y
795B FC   7752     12520 LN4A     LDD     X1       X=(X1+X2)/2
795E F3   7756     12530          ADDD    X2
7961 47            12540          ASRA
7962 56            12550          RORB
7963 1F   01       12560          TFR     D,X
7965 FC   7754     12570          LDD     Y1       Y=(Y1+Y2)/2
7968 F3   7758     12580          ADDD    Y2
796B 47            12590          ASRA
796C 56            12600          RORB
796D 1F   02       12610          TFR     D,Y
796F 8C   FF80     12620          CMPX    #LEFT
7972 2E   08       12630          BGT     LN5A
7974 108C FFA0     12640          CMPY    #BOTTOM
7978 102F 0126     12650          LBLE    LNDON1   NO VISIBLE LINE
797C 8C   007F     12660 LN5A     CMPX    #RIGHT
797F 2D   08       12670          BLT     LN6A
7981 108C 005F     12680          CMPY    #TOP
7985 102C 0119     12690          LBGE    LNDON1   NO VISIBLE LINE
7989 8C   FF80     12700 LN6A     CMPX    #LEFT
798C 2D   06       12710          BLT     LN7A
798E 108C 005F     12720          CMPY    #TOP
7992 2F   09       12730          BLE     LN8A
7994 BF   7752     12740 LN7A     STX     X1       REPLACE LEFT POINT BY MIDPOINT
AND REPEAT
7997 10BF 7754     12750          STY     Y1
799B 20   BE       12760          BRA     LN4A
799D 8C   FF80     12770 LN8A     CMPX    #LEFT
79A0 2F   0F       12780          BLE     LN9A
79A2 108C 005F     12790          CMPY    #TOP
79A6 2C   09       12800          BGE     LN9A
79A8 BF   7756     12810          STX     X2       REPLACE RIGHT POINT BY MIDPOINT
79AB 10BF 7758     12820          STY     Y2
79AF 20   AA       12830          BRA     LN4A
79B1 BF   7752     12840 LN9A     STX     X1       X1,Y1 NOW ON LEFT OR BOTTOM BOU
NDARY
79B4 10BF 7754     12850          STY     Y1
79B8 35   30       12860          PULS    X,Y      RECOVER ORIGINAL RIGHT POINT
79BA BF   7756     12870          STX     X2
79BD 10BF 7758     12880          STY     Y2
                   12890 * CHECK IF MUST CLIP RIGHT END OF LINE
                   12900 *
79C1 FC   7756     12910 LN3A     LDD     X2
79C4 1083 007F     12920          CMPD    #RIGHT
79C8 2E   09       12930          BGT     LN10A
79CA FC   7758     12940          LDD     Y2
79CD 1083 FFA0     12950          CMPD    #BOTTOM
79D1 2C   55       12960          BGE     LN18A
                   12970 *
                   12980 * CLIP RIGHT END OF LINE
                   12990 *
79D3 BE   7752     13000 LN10A    LDX     X1
79D6 10BE 7754     13010          LDY     Y1
79DA 34   30       13020          PSHS    X,Y
79DC FC   7752     13030 LN11A    LDD     X1       X=(X1+X2)/2
79DF F3   7756     13040          ADDD    X2
79E2 47            13050          ASRA
79E3 56            13060          RORB
79E4 1F   01       13070          TFR     D,X
79E6 FC   7754     13080          LDD     Y1       Y=(Y1+Y2)/2
79E9 F3   7758     13090          ADDD    Y2
79EC 47            13100          ASRA
79ED 56            13110          RORB
79EE 1F   02       13120          TFR     D,Y
79F0 8C   007F     13130          CMPX    #RIGHT
79F3 2E   06       13140          BGT     LN12A
79F5 108C FFA0     13150          CMPY    #BOTTOM
79F9 2C   09       13160          BGE     LN13A
79FB BF   7756     13170 LN12A    STX     X2       MOVE RIGHT POINT TO MIDPOINT AN
D REPEAT
79FE 10BF 7758     13180          STY     Y2
7A02 20   D8       13190          BRA     LN11A
7A04 8C   007F     13200 LN13A    CMPX    #RIGHT
7A07 2C   0F       13210          BGE     LN14A
7A09 108C FFA0     13220          CMPY    #BOTTOM
7A0D 2F   09       13230          BLE     LN14A
7A0F BF   7752     13240          STX     X1       MOVE LEFT POINT TO MIDPOINT AND
 REPEAT
7A12 10BF 7754     13250          STY     Y1
7A16 20   C4       13260          BRA     LN11A
7A18 BF   7756     13270 LN14A    STX     X2       MOVE X2,Y2 TO TOP OR RIGHT BOUN
DARY, RECOVER X1,Y1
7A1B 10BF 7758     13280          STY     Y2
7A1F 35   30       13290          PULS    X,Y
7A21 BF   7752     13300          STX     X1
7A24 10BF 7754     13310          STY     Y1
```

Listing continued

Listing continued

```
                        13320 *
                        13330 * DECIDE IF SLOPE > 1
                        13340 *
7A28 B6   7757      13350 LN18A   LDA     X2P     X1P, ETC HOLD CORRECT ONE-BYTE
SIGNED VALUES FOR THE COORDINATES, WHICH ARE NOW ON THE VIRTUAL SCREEN
7A2B B0   7753      13360         SUBA    X1P
7A2E 34   02        13370         PSHS    A
7A30 F6   7755      13380         LDB     Y1P
7A33 F0   7759      13390         SUBB    Y2P
7A36 E1   E0        13400         CMPB    ,S+
7A38 25   33        13410         BLO     ESELN
                        13420 *
                        13430 * DRAW A LINE WITH NEGATIVE SLOPE <= -1
                        13440 *
7A3A 34   04        13450         PSHS    B       SAVE NUMBER OF POINTS ON LINE -
 1
7A3C BD   7AA5      13460         JSR     GETSLP  SLOPE = -DX/DY, AN 8-BIT FRACTI
ON
7A3F BD   7AC0      13470         JSR     FRSTBT  SET X,U TO POINT TO BYTE AND BI
T ON GRAPHICS SCREEN CORRESP. TO X1,Y1
7A42 35   04        13480         PULS    B
7A44 4F             13490         CLRA
7A45 1F   02        13500         TFR     D,Y
7A47 31   21        13510         LEAY    1,Y     Y=#POINTS ON LINE
7A49 5F             13520         CLRB
7A4A A6   84        13530 LN15A   LDA     ,X      SET THE POINTS ON THE GRAPHICS
SCREEN FOR THE LINE
7A4C AA   C4        13540         ORA     ,U      GET THE BYTE AND OR IN THE BIT
7A4E A7   84        13550         STA     ,X
7A50 31   3F        13560         LEAY    -1,Y    REDUCE POINT COUNT
7A52 1027 004E      13570         LBEQ    LNDONE
7A56 30   88 20     13580         LEAX    $20,X   DECREASE Y-COORDINATE BY 1
7A59 FB   775A      13590         ADDB    SLOPE   INCREASE X-COORDINATE BY SLOPE
(A FRACTION)
7A5C 24   EC        13600         BCC     LN15A   B=X MOD 1
7A5E 33   41        13610         LEAU    1,U     INCREASE X-COORDINATE BY 1
7A60 1183 7751      13620         CMPU    #LASTBT TRY TO MOVE RIGHT ONE BIT
7A64 23   E4        13630         BLS     LN15A
7A66 CE   774A      13640         LDU     #BITS   ELSE USE NEXT BYTE AND LEFT BIT
7A69 30   01        13650         LEAX    1,X
7A6B 20   DD        13660         BRA     LN15A
                        13670 *
                        13680 * DRAW LINE WITH NEGATIVE SLOPE > -1
                        13690 *
7A6D 34   02        13700 ESELN   PSHS    A       SAVE # POINTS ON  LINE - 1
7A6F 1E   89        13710         EXG     A,B
7A71 BD   7AA5      13720         JSR     GETSLP  SLOPE = -DY/DX
7A74 BD   7AC0      13730         JSR     FRSTBT  X,U ARE BYTE AND BIT CORRESP. T
O X1,Y1
7A77 35   04        13740         PULS    B
7A79 4F             13750         CLRA
7A7A 1F   02        13760         TFR     D,Y
7A7C 31   21        13770         LEAY    1,Y     Y=#POINTS ON LINE
7A7E 5F             13780         CLRB
7A7F A6   84        13790 LN16A   LDA     ,X      DRAW THE LINE
7A81 AA   C4        13800         ORA     ,U      GET  THE  BYTE, OR IN  THE BIT
7A83 A7   84        13810         STA     ,X
7A85 31   3F        13820         LEAY    -1,Y    REDUCE THE POINT COUNT
7A87 1027 0019      13830         LBEQ    LNDONE
7A8B 33   41        13840         LEAU    1,U     MOVE ONE BIT TO RIGHT
7A8D 1183 7751      13850         CMPU    #LASTBT IF POSSIBLE
7A91 23   05        13860         BLS     LN17A
7A93 30   01        13870         LEAX    1,X     ELSE MOVE TO NEXT BYTE AND FIRS
T BIT
7A95 CE   774A      13880         LDU     #BITS
7A98 FB   775A      13890 LN17A   ADDB    SLOPE
7A9B 24   E2        13900         BCC     LN16A   NO OVERFLO TO NEXT INTEGER WHEN
 ADDING SLOPE TO Y
7A9D 30   88 20     13910         LEAX    $20,X   ELSE SUBTRACT 1 FROM Y-COORDINA
TE
7AA0 20   DD        13920         BRA     LN16A
7AA2 35   30        13925 LNDON1  PULS    X,Y
7AA4 39             13930 LNDONE  RTS
                        13940 *
                        13950 * ASSUME A<B CONTAIN UNSIGNED INTEGERS, PUT A/B INTO SL
OPE
                        13960 *
7AA5 34   04        13970 GETSLP  PSHS    B
7AA7 5F             13980         CLRB
7AA8 8E   0008      13990         LDX     #8
7AAB 58             14000 GET5    ASLB
7AAC 48             14010         ASLA
7AAD 25   04        14020         BCS     GET1
7AAF A1   E4        14030         CMPA    ,S
7AB1 25   03        14040         BLO     GET4
7AB3 A0   E4        14050 GET1    SUBA    ,S
7AB5 5C             14060         INCB
7AB6 30   1F        14070 GET4    LEAX    -1,X
7AB8 26   F1        14080         BNE     GET5
7ABA 32   61        14090         LEAS    1,S
7ABC F7   775A      14100         STB     SLOPE
7ABF 39             14110         RTS
                        14120 *
                        14130 * FIND BYTE X AND BIT (U) ON  GRAPHICS SCREEN CORRESPON
DING TO X1,Y1
                        14140 * X=$600+Y1*32+X1DIV8 AND U=#BITS+X1MOD8
                        14150 * WE USE $600+Y1*(256/8)+X1DIV8 FOR X, AND AS WE DIVIDE
 X1 BY THREE SUCCESSIVE 2'S, WE PICK UP THE BITS FOR MODIFYING U.
                        14160 *
7AC0 BE   721F      14170 FRSTBT  LDX     TL
7AC3 86   5F        14180         LDA     #95
7AC5 B0   7755      14190         SUBA    Y1P
7AC8 F6   7753      14200         LDB     X1P
7ACB CB   80        14210         ADDB    #$80
7ACD CE   774A      14220         LDU     #BITS
7AD0 1C   FE        14230         ANDCC   #$FE
7AD2 46             14240         RORA
7AD3 56             14250         RORB
7AD4 24   02        14260         BCC     FRST1
7AD6 33   41        14270         LEAU    1,U
7AD8 47             14280 FRST1   ASRA
7AD9 56             14290         RORB
7ADA 24   02        14300         BCC     FRST2
7ADC 33   42        14310         LEAU    2,U
7ADE 47             14320 FRST2   ASRA
7ADF 56             14330         RORB
7AE0 24   02        14340         BCC     FRST3
7AE2 33   44        14350         LEAU    4,U
7AE4 30   8B        14360 FRST3   LEAX    D,X
7AE6 39             14370         RTS
          0000      14380         END
00000 TOTAL ERRORS

BIG1    761B
BIG2    7645
BIG3    764D
BIG8    7657
BIG9    7666
BIGGER  7618
BITS    774A
BOTTOM  FFA0
DISP1   722B
DISP2   726C
DISP3   7237
DISP4   7293
```

Listing continued

Listing continued

```
DISPLA  7221
ENELN   78DF
ESELN   7A6D
FRST1   7AD8
FRST2   7ADE
FRST3   7AE4
FRSTBT  7AC0
GET1    7AB3
GET4    7AB6
GET5    7AAB
GETBIG  766C
GETSLP  7AA5
GETSML  76FC
KEY     718D
KEYVAL  7731
LASTBT  7751
LASTPT  70E1
LEFT    FF80
LINE    775B
LINES   70EA
LN1     777B
LN10    7845
LN10A   79D3
LN11    784E
LN11A   79DC
LN12    786D
LN12A   79FB
LN13    7876
LN13A   7A04
LN14    788A
LN14A   7A18
LN15    78BC
LN15A   7A4A
LN16    78F1
LN16A   7A7F
LN17    790A
LN17A   7A98
LN18    789A
LN18A   7A28
LN2     77C4
LN2A    7952
LN3     7833
LN3A    79C1
LN4     77CD
LN4A    795B
LN5     77EE
LN5A    797C
LN6     77FB
LN6A    7989
LN7     7806
LN7A    7994
LN8     780F
LN8A    799D
LN9     7823
LN9A    79B1
LNDON1  7AA2
LNDONE  7AA4
MAIN    7177
MAIN1   719F
MAIN10  71F0
MAIN11  71F9
MAIN12  7202
MAIN2   71A8
MAIN3   71B1
MAIN4   71BA
MAIN5   71C3
MAIN6   71CC
MAIN7   71D5
MAIN8   71DE
MAIN9   71E7
P1300   723E
PAND    731B
PAND1   732F
PAND2   731E
PAND4   7337
PAND5   7349
PANL    72AF
PANL1   72C3
PANL2   72B2
PANL4   72CB
PANL5   72DD
PANR    72E5
PANR1   72F9
PANR2   72E8
PANR4   7301
PANR5   7313
PANU    7351
PANU1   7365
PANU2   7354
PANU4   736D
PANU5   737F
POINTS  7000
REP1    7209
REPEAT  7207
RIGHT   007F
RMX     740F
RMY     749D
RMZ     752D
ROTATE  7535
ROTMX   73D0
ROTMX1  73D3
ROTMX2  73F8
ROTMX3  7400
ROTMY   7460
ROTMY1  7463
ROTMY2  7486
ROTMY3  748E
ROTMZ   74EE
ROTMZ1  74F1
ROTMZ2  7516
ROTMZ3  751E
ROTX    7387
ROTX1   738A
ROTX2   73AF
ROTX3   73B7
ROTY    7419
ROTY1   741C
ROTY2   7441
ROTY3   7449
ROTZ    74A7
ROTZ1   74AA
ROTZ2   74CD
ROTZ3   74D5
RX      73C6
RY      7456
RZ      74E4
SELINE  7914
SLOPE   775A
SMALLR  76AB
SML1    76AE
SML2    76C5
SMLR1   76C7
SMLR5   76FB
SMLR6   76CF
TB1     76AA
TL      721F
TOOBIG  76A1
TOP     005F
X1      7752
X1P     7753
X2      7756
X2P     7757
Y1      7754
Y1P     7755
Y2      7758
Y2P     7759
```

```
SLOPE = (Y2 - Y1)/(X2 = X1)
BREG = 0
XREG = BYTE(X1,Y1)
UREG = BIT(X1,Y1)
DO X2 - X1 + 1 TIMES
  Bit UREG of byte XREG = 1 (turn on pixel)
  UREG = UREG - 1 (move one pixel to the right)
  IF UREG = - 1 THEN (moving to the next byte if necessary)
    UREG = 7
    XREG = XREG + 1
  BREG = BREG + SLOPE
  IF BREG≥1 THEN (have we moved far enough to the right to justify)
    BREG = BREG - 1 (going up one pixel?)
    XREG = XREG - $20
ENDDO
```

Table 4. A New Version of the Line-Drawing Routine

```
        ORG $3900
        LDY #0
        LDX #$1E00
LABEL   STY ,X++    STORE ZEROES INTO
        CMPX #$3600 MEMORY $1E00#$3600
        BNE LABEL
        RTS
```

Table 5. Assembly-Language Routine to Clear Graphics Pages

Continued from p. 115

```
    FROM EXEC"
 90 STOP
100 DATA 16, 142, 0, 0, 142, 30, 0, 16,
    175, 129, 140, 54, 0, 38, 248, 57
```

The machine-language portion of this program is a compilation of the Assembly-language routine that clears graphics pages 5–8 (bytes $1E00–$35FF) (see Table 5).

Type PCLEAR 8: RUN. The program will run and you should see on the screen "Successful return from Basic." Now type PCLEAR 4:RUN. Watch the program crash; it never reaches line 80. Run it again. The program works. Type PCLEAR 4: RUN. Crash.

What is going on here?

When you execute PCLEAR N, your program is moved so that its text begins after the Nth graphics page. Memory locations 25–26 contain the starting address of your program. Try executing PCLEARs and PEEKing this location. It changes as the PCLEAR changes your program's location.

If your program is located after graphics page 4 when you run it, then, even though you move it with PCLEAR 8 in the program, clearing graphics pages 5–8 causes the program to crash. If you run your program after moving it to the end of graphics page 8, the program executes flawlessly.

2) I will describe the Basic commands CSAVEM and CLOADM, since Radio Shack omitted them from the CoCo manuals. To save a block of memory between address ADDR1 and ADDR2 in a tape file called "name," use this command: CSAVEM"name",ADDR1, ADDR2,ADDR3. ADDR3, the "execution address," is any value between 0 and $7FFF. It is syntactically necessary but has no apparent effect.

To recover the information from tape and place it back into memory (in the same location), use CLOADM "name". No address parameters are necessary, and the execution address will be stored in memory locations $9D–$9E (decimal 158–159). The files saved with CSAVEM are called *binary files*. They are useful because they load and save very quickly, and any information can be saved, even if it is not recognizable by Basic as ASCII or numeric data. ■

David Meredith is a professor of mathematics at San Francisco State University, 1600 Holloway Ave., San Francisco, CA 94132.

A WAYNE GREEN PUBLICATION
August 1983 USA $2.95

HOT CoCo

THE MAGAZINE FOR TRS-80 COLOR COMPUTER AND TDP-100 USERS T.M.

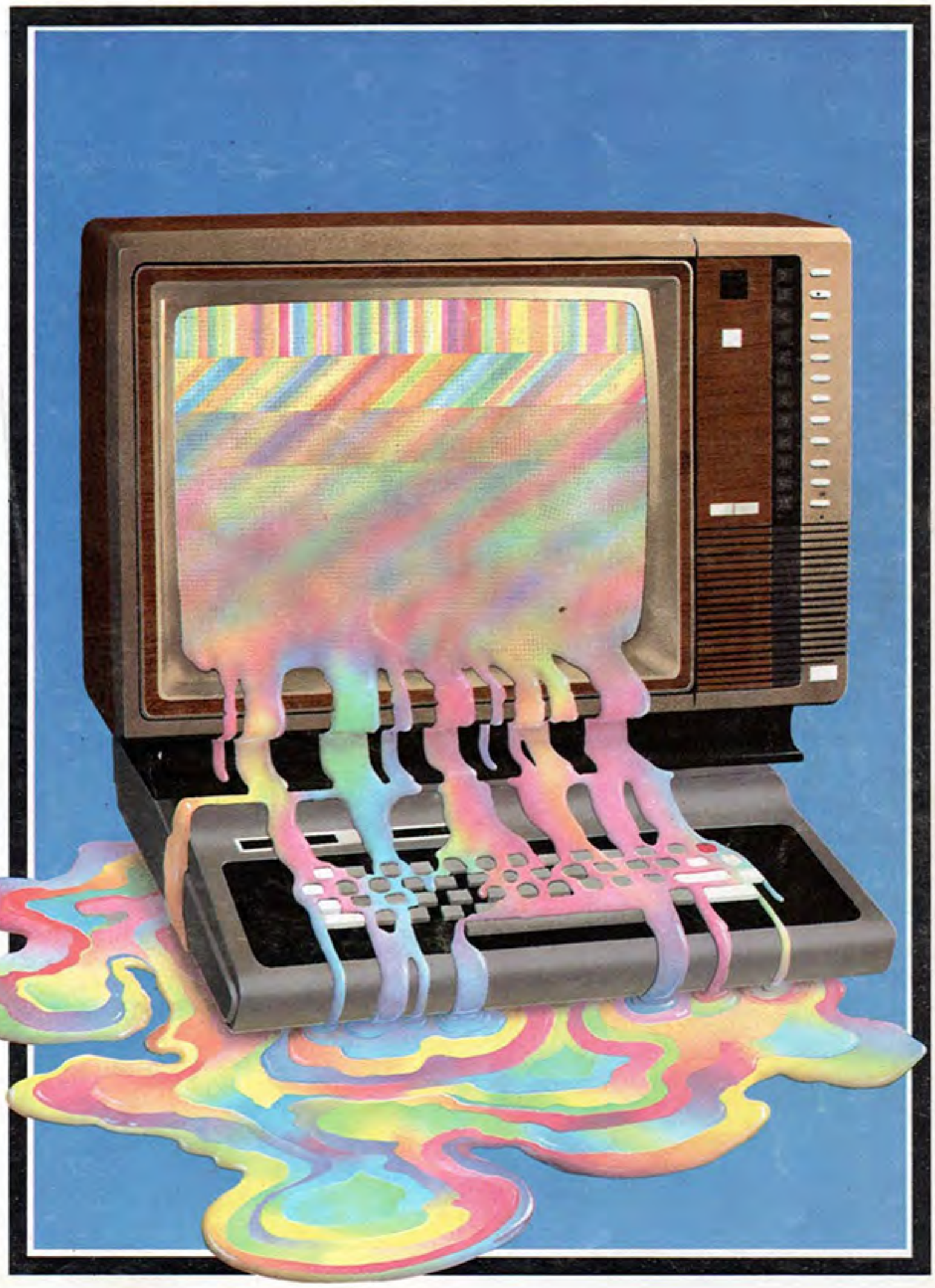
Graphics! Create a Masterpiece

Does Your CoCo Have 1,000 Colors? It Can!

Draw and Manipulate Your Own 3-D Graphics

Photograph Your Screen Displays

How to Print Multicolor Bar Graphs



PLUS: Reviews, Commentary, and Technical Tips