Faster Circles for Apples

Daniel Lee's article, "Fast Circle Routine," in *DDJ* No. 79 (May 1983) inspired me to create a similar circle-making approach for the Apple. Although I first wrote an Applesoft BASIC program to test the logic, I chose variable names that could serve for both BASIC and assembly language. Thus "X" and "Y" refer to the X and Y registers of the Apple's 6502 chip, and "Xcrd" and "Ycrd" are the coordinates for the points of the circle.

A circle of radius R centered at (Xmid, Ymid) is described by the familiar formula

by Myron L. Pulier

Myron L. Pulier, M.D., 101 Cedar Lane, Teaneck, New Jersey 07666.

```
(Ycrd-Ymid)^2
    + (Xcrd-Xmid)^2 = R^2
```

where Xcrd is the horizontal variable, Ycrd is the vertical variable, and Xmid and Ymid are constants. Differentiating with respect to Xcrd gives

```
2*(Ycrd-Ymid)*dYcrd/dXcrd
    + 2*(Xcrd-Xmid) = 0
```

whence

```
dYcrd/dXcrd =
    - (Xmid-Xcrd)/(Ymid-Ycrd)
```

The last equation implies that, in drawing the circle, if we increase Xcrd by 1 to plot the next point we must decrease Ycrd by

```
(Xmid-Xcrd)/(Ymid-Ycrd)
```

The slowest operation here is division by Ymid-Ycrd, which must be performed each time we want a new value for Ycrd. We can reduce the number of these divisions by evaluating the expression for only one eighth of the circle and by plotting the rest of the circle symmetrically about the coordinate axes and about a diagonal.

It is best to select the upper left extreme of the circle as the starting point. According to the Apple coordinate system, where point (0,0) is the upper left corner of the screen, our starting point is given by

```
(Xmid-R/SQR(2),Ymid-R/SQR(2))
```

From here we move to the right and stop at the extreme top of the circle, which is point (Xmid,Ymid-R). This choice of starting and ending points facilitates a simple FOR-NEXT program loop (FOR Xcrd = Xmid-R/SQR(2) to Xmid) and avoids the divide-by-zero error we might encounter at the extreme right and left of the circle, where the slope is undefined.

Reader Commentary

More Fast Circles . . .

Dear *DDJ*,

Daniel L. Lee's algorithm has got to be faster than Microsoft's pedestrian CIRCLE command, but both suffer from the same malady: they reinvent the wheel – only this time it's square!

When I think I've discovered a marvelous algorithm, I wonder if I've outsmarted the professionals. I usually haven't. But hope springs eternal. I search the literature anyway. My brainchild is at least 17 years old [B. K. P. Horn, "Circle Generators for Display Devices," *Computer Graphics and Image Processing* (5), pp. 280-288 (1976)].

Neither trigonometry nor calculus is needed to devise a circle generator. For a circle of radius R one wants to plot points (X,Y) with integer coordinates which most nearly solve the equation

$$X^2 + Y^2 = R^2$$

The difference between the left side and R^2 is a measure of nearness. A suitable circle generator simply chooses successive points to minimize this difference. The enclosed listing (see Listing Three, page 30) is a rendering of such an algorithm. It generates points for about one eighth of the circle and, using the symmetry of the circle, plots eight points for each point generated. For use with digital plotters, the algorithm is invoked eight times forward and backward so that the points are of concentric circles; the low algorithm is plotted in counter-clockwise order. I enclose a plot of concentric circles in low resolution to exhibit the algorithm's behavior (see Figure 2, below).

William A. McWorter, Jr.
Mathematics Department
Ohio State University
231 W. 18th Avenue
Columbus, OH 43210

(Listing Three begins on page 30)

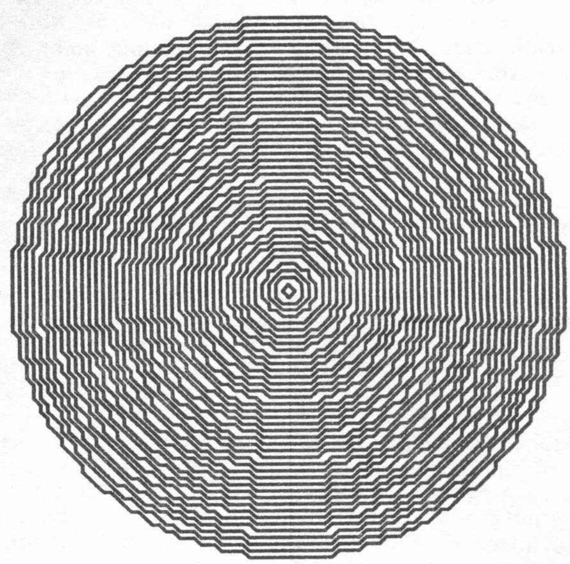

Figure 2.

See Figure 1 (at right).

For each point that we find by the above method, we can generate seven symmetric points by reflection through the horizontal and vertical diameters of the circle and by exchanging Xcrd with Ycrd. A point (Xcrd,Ycrd) is ABS(Xcrd-Xmid) distant from the vertical axis of the circle. Since its reflection should be the same distance from this axis, its abscissa is 2*Xmid-Xcrd. Similarly, reflection through the horizontal diameter gives an ordinate of 2*Ymid-Ycrd. To reflect a point through the diagonal line that runs from upper left to lower right, we find the point whose distance to the vertical axis equals the distance of the original point to the horizontal axis and whose distance to the horizontal axis is the same as the original point's distance from the vertical axis, namely

(Xmid+Ymid-Ycrd,Xmid+Ymid-Xcrd)

This new point can now be reflected as before through the vertical and horizontal axes.

In Listing One (page 21), subroutine 9000 plots four points symmetrically about the horizontal and vertical axes. Lines 10050 and 10055 switch the X and Y coordinates, then line 10060 calls 9000 to plot the four new points. The parame-

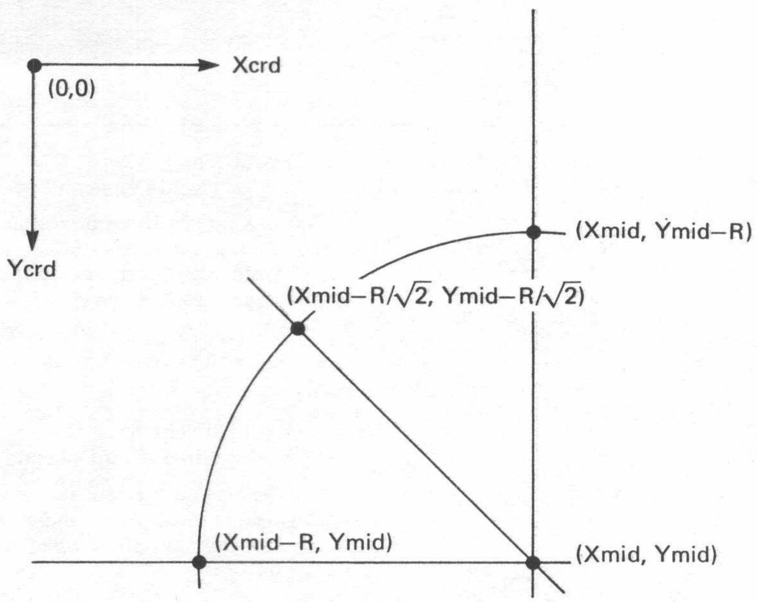

Figure 1.
Faster Circle

Reader Commentary

. . . And Fast Ellipses

Dear Sirs:

Mr. Daniel Lee, in the May '83 issue, presented a fast circle generator. It compensated for any given screen aspect ratio, and as such may be used as an ellipse generator. I submit the algorithm described below as an even faster alternative. The speed improvement results from the elimination of all division and most of the multiplication. The approach taken could easily be modified to allow the generation of arcs.

The method which I present here is based on the equation of the circle, and a trick which eliminates a great deal of multiplication. There is no calculus or trigonometry involved, implicitly or explictly.

The equation of the circle is well known:

$$x^2 + y^2 = r^2 \qquad [1]$$

where r is the radius. Since we want to minimize multiplication, we have to use "magic." A magical property of the positive integers is that the square of a positive integer *n* is the sum of the first *n* odd numbers. This means that if we want to compute x^2 for each x we can actually plot (i.e., each integer x), we only need to know which odd numbers to add up. The same applies to y^2.

In order to plot a circle, we might start at the point (0,r) and plot towards (r,0), using symmetry to generate the other arcs of the circle. This would mean that x would go from 0 to r, y would go from r to 0, x^2 would go from 0 to r^2, and y^2 would go from r^2 to 0. It is easier than it first appears to calculate y^2. Note that y^2 is the sum of the odd numbers from 1 to $2y-1$. In the initialization phase it will be necessary (perhaps) to compute y^2 directly, but for $y' = y-1$, $y'^2 = y^2 - (2y-1)$.

Above I said "perhaps" because it develops that one does not need to refer directly to y^2 or even to x^2. The procedure for drawing the circle requires that we assume, as we did above, that we will draw primarily from (r,0) to (0,r) and use symmetry to generate the rest of the points. As we compute the points for the primary arc, we maintain a total *e*. The total starts at 0; for every time we actually move in the positive x direction, we add $2x-1$ to *e*; for every time we actually move in the negative y direction, we subtract $2y-1$ from *e*. We decide precisely which step or combination of steps to take by insisting that the *e* that would result from the step or combination of steps be as close to 0 as possible.

An Ellipse

To generate an ellipse is a slightly more complex matter, but in the end we lose little speed. The equation for an ellipse centered at the origin is

$$b^2x^2 + a^2y^2 = a^2b^2 \qquad [2]$$

where b is the positive y-intercept, a is the positive x-intercept, and a/b is the resulting aspect ratio. I claim that in order to successfully trace the ellipse we need only do exactly as we do for the circle, but we must multiply every reference to x by b^2 and every reference to y by a^2. In other words, every time we actually move in the positive x direction, we add $b^2(2x-1)$ to *e*; for every time we actually move in the negative y direction, we must subtract $a^2(2y-1)$ from *e*. Again we decide which step or combination of steps to take by insisting that the *e* that would result from the step or combination of steps be as close to 0 as possible. *In this case we are plotting* from (0,b) to (a,0).

If perhaps the terms $b^2(2x-1)$ and $a^2(2y-1)$ look like they involve too much multiplication, please realize that in fact no multiplication is required. For example, we would already know the evaluation of $b^2(2x-1)$ to

(Continued in box on page 20)

ters R2, X2, Y2, and XY have been introduced to speed computation.

In the segment of the circle from the upper left point through the upper middle, the change in Ycrd is fractional for each unit change in Xcrd. Because the Apple plotting routine deals with integers, the decrement in Ycrd builds until it causes the line being drawn to move up one full position.

The assembly program in Listing Two (page 22) runs much faster than its Applesoft equivalent. Since Xcrd can range from 0 through 279, it must be a double-precision variable. It occupies locations XCRDH and XCRDL. Ycrd is supplemented by a fractional portion stored in YCRDF. Names of other double-precision integer parameters are terminated with -H or -L for the high- and low-order portions, respectively. Single-precision assignment is indicated in the comments by "<—", while double precision is "<<—".

The TEST program plots a circle of radius 40 and midpoint (120,80). It initializes the hires screen by calling TURNON. The subroutine called EIGHTH performs calculations for the one-eighth circle. Here the first order of business is to approximate the value of R/SQR(2) by using R*3/4 instead. Note that 3/4 in decimal is 1/2 + 1/4, or 0.11 in binary.

The next lines of the assembly program are a straightforward translation of their Applesoft equivalents. Lines 75 and 76 initialize the value of YCRDF to 0. PLOTFOUR is called in lines 104 and 119 to place four points symmetrically about the horizontal and vertical axes of the circle. PLOTFOUR uses the Applesoft HPLOT routine to perform the actual plotting. HPLOT requires that the horizontal coordinate be in the Y and X registers,

(Continued from page 19)

be, say, e_x. To determine $e_{x'}$ when $x' = x+1$, note that

$$b^2(2x'-1) = b^2[2(x+1)-1] = b^2(2x-1)+2b^2;$$

in other words,

$$e_{x'} = e_x + 2b^2.$$

A similar result obtains for the negative y direction, which we will simply state:

$$e_{y'} = e_y - 2a^2.$$

Algorithm Summary

To summarize the algorithm: start with the point (0,b). Initialize e to 0, e_x to b^2, e_y to $2a^2b-a^2$, e_{xy} to e_x+e_y. Plot the current point and corresponding points in the other quadrants of the ellipse. Choose the next point so that e plus $e_{whatever}$ is minimized. Set e according to that choice, and update e_x, e_y, and e_{xy}. When the point (a,0) is arrived at, the ellipse is complete.

The Listing

The program shown in Listing Four (page 30) is an MBASIC program intended to interface to an LSI ADM-3A terminal. Obviously, if speed is a concern, BASIC is not the language of choice. I chose it to permit the program to be tried out basically anywhere, since my facilities for computer graphics are one-of-a-kind.

Lines 1050-1240 are the routine itself. The point-plotting routine is on lines 1310-1341.

Caveat

There is one thing that the implementor should be aware of before he or she starts, to prevent untraceable bugs. The formulae for e_x, e_y, and e_{xy} include squares of a and b. These squares accumulate to a large total rather quickly. The solution is to use a wide word to store the total, and perhaps (depending on the size of your screen in pixels) the values of e_x, e_y, and e_{xy} as well.

Drawing Arcs

The method can be modified to draw arcs (see Figure 3, below) elliptical or otherwise, with careful initialization and a well-considered termination condition. The initialization involves calculating e_x, e_y, and e_{xy} for the initial point of the arc to be drawn. The routine should terminate when the last point of the arc is drawn. The actual coordinates of the final point should be calculated in some fashion that allows for rational numbers, and then a point with integer coordinates should be chosen that approximates the actual point. This can be done by using the equation of the ellipse. In other words, the best integer approximation (x_i,y_i) of the terminating point (x,y) is the one for which $(bx_i)^2+(ay_i)^2$ is closest to $(ab)^2$. Again, the integer coordinates of the final point should be computed in the initialization phase and used as the termination condition.

Conclusion

This routine can draw an ellipse quickly, using no multiplication once initialized. It should be easily implemented in 68000 assembly language, owing to that processor's 32-bit register operations. A little more difficulty should be anticipated by users of the 8086, 6809 or Z80, though their 16-bit addition capabilities can be used to advantage. HLLs can speedily draw circles with this routine, as well, because of its incremental nature. And finally, the algorithm can draw arcs easily.

Michael T. Enright
2360 Hosp Way, #132
Carlsbad, CA 92008

(Listing Four begins on page 30)

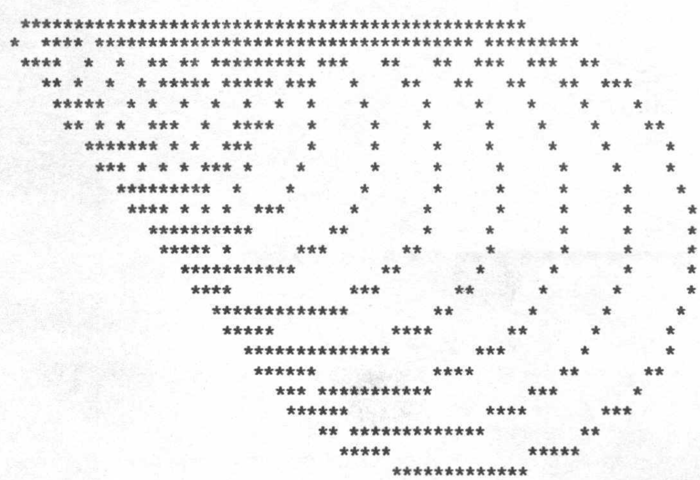

Figure 3.

and the vertical coordinate in the A register. HPLOT returns these coordinate values at the zero-page locations named LASTHH, LASTHL, and LASTV.

The division sequence starts on line 135. Since the divisor and dividend are single precision, we can use a technique that divides a one-byte divisor stored in DIVISOR into a one-byte dividend held in the A register. The eight shift operations and eight subtractions required are counted via the X register. The result of the division is a binary fraction generated in QUOTIENT. This quotient is subtracted from the previous value of YCRDF. If a borrow is required, we decrement the integer portion of Ycrd. In any case, Xcrd must be incremented by 1 in a double-precision operation. The FOR-NEXT loop of the BASIC version is implemented in assembly language by counting the value of R2 down through zero, since R/SQR(2) points will be plotted for one eighth of the circle.

Using zero-page locations for variables and parameters and a faster division algorithm will increase speed, but the bottleneck is the Applesoft HPLOT routine, which maps horizontal and vertical coordinates into the Apple video locations. Replacing that routine with a table lookup results in very fast circle generation.

With some modification the "faster circle" technique can produce filled-in disks or wedges for pie charts. It can also rotate and translate shapes and objects quickly for animation effects.

(Listings begin below)

Circles (Text begins on page 18)

Listing One

```
1000    ***************TURNON

1005  Hgr
1010  Hcolor = 3


2000    ***************TEST

2005  R = 40
2010  XMID = 120
2015  YMID = 80
2020  Gosub 10000
2025  End


9000    ***************PLOTFOUR

9005  Hplot H,V
9010  Hplot X2 - H,V
9015  Hplot X2 - H,Y2 - V
9020  Hplot H,Y2 - V
9025  Return


10000   **************EIGHTH

10005  R2 = R * .7
10010  YCRD = YMID - R2
10015  X2 = XMID + XMID
10020  Y2 = YMID + YMID
10025  XY = XMID + YMID
10030  For XCRD = XMID - R2 To XMID
10035        H = XCRD
10040        V = YCRD
10045        Gosub 9000
                      * PLOT 4 POINTS
10050        V = XY - XCRD
10055        H = XY - YCRD
10060        Gosub 9000
                  * PLOT 4 POINTS

10065        YCRD = YCRD -
                  (XMID - XCRD) /
                          (YMID - YCRD)

10070  Next
10075  Return

END OF LISTING

PROGRAM LENGTH = 421 BYTES,
           TOTAL OF 31 LINE NUMBERS


27 TOTAL NON-REM STATEMENTS,
                   6 TOTAL REMARKS


END
PR#0
```

End Listing One

Listing Two

```
0800                            LST
0800 2C 50 C0     TURNON      BIT GRAPHICS
0803 2C 52 C0                 BIT FULL
0806 2C 54 C0                 BIT PAGE1
0809 2C 57 C0                 BIT HIRES
080C A9 FF                    LDA ##$FF
080E 85 E4                    STA HCOLOR
0810 A9 20                    LDA ##$20
0812 85 E6                    STA HPAGE
0814              ;
0814 A9 28        TEST        LDA #40             ;R<-40
0816 8D 2F 09                 STA R
0819              ;
0819 A9 78                    LDA #120            ;XMID<<-120
081B 8D 37 09                 STA XMIDL
081E A9 00                    LDA #0
0820 8D 36 09                 STA XMIDH
0823              ;
0823 A9 50                    LDA #80             ;YMID<-80
0825 8D 3C 09                 STA YMID
0828              ;
0828 20 64 08                 JSR EIGHTH          ;DRAW CIRCLE
082B 60                       RTS
082C              ;
082C AC 2D 09     PLOTFOUR    LDY HH              ;HPLOT H,V
082F AE 2E 09                 LDX HL
0832 AD 31 09                 LDA V
0835 20 57 F4                 JSR HPLOT
0838 AD 33 09                 LDA X2L             ;HPLOT X2-H,V
083B 38                       SEC
083C E5 E0                    SBC LASTHL
083E AA                       TAX
083F AD 32 09                 LDA X2H
0842 E5 E1                    SBC LASTHH
0844 A8                       TAY
0845 AD 31 09                 LDA V
0848 20 57 F4                 JSR HPLOT
084B A4 E1                    LDY LASTHH          ;HPLOT H2-H,Y2-V
084D A6 E0                    LDX LASTHL
084F AD 3B 09                 LDA Y2L
0852 38                       SEC
0853 E5 E2                    SBC LASTV
0855 20 57 F4                 JSR HPLOT
0858 AC 2D 09                 LDY HH
085B AE 2E 09                 LDX HL
085E A5 E2                    LDA LASTV
0860 20 57 F4                 JSR HPLOT
0863 60                       RTS
0864              ;
0864 AD 2F 09     EIGHTH      LDA R               ;R2<-R*3/4
0867 4A                       LSR
0868 18                       CLC
0869 6D 2F 09                 ADC R
```

(Continued on page 24)

Circles (Listing continued, text begins on page 18)

(Listing continued, text begins on page 18)

Listing Two

```
086C 6A              ROR
086D 8D 30 09        STA R2
0870          ;
0870 AD 3C 09        LDA YMID      ;YCRD<-[YMID-R2]
0873 AA              TAX           ;X<-YMID FOR LATER
0874 38              SEC
0875 ED 30 09        SBC R2
0878 8D 3D 09        STA YCRD
087B          ;
087B AD 37 09        LDA XMIDL     ;X2<<-[2*XMID]
087E 0A              ASL
087F 8D 33 09        STA X2L
0882 AD 36 09        LDA XMIDH
0885 2A              ROL
0886 8D 32 09        STA X2H
0889          ;
0889 8A              TXA           ;A<-YMID
088A 0A              ASL           ;Y2<<-[2*YMID]
088B 8D 3B 09        STA Y2L
088E A9 00           LDA #0
0890 8D 3E 09        STA YCRDF     ;YCRDF<-0
0893 2A              ROL
0894 8D 3A 09        STA Y2H
0897          ;
0897 8A              TXA           ;A<-YMID
0898 18              CLC           ;XY<<-[XMID+YMID]
0899 6D 37 09        ADC XMIDL
089C 8D 39 09        STA XYL
089F A9 00           LDA #0
08A1 6D 36 09        ADC XMIDH
08A4 8D 38 09        STA XYH
08A7          ;
08A7 AD 37 09        LDA XMIDL     ;XCRD<<-[XMID-R2]
08AA 38              SEC
08AB ED 30 09        SBC R2
08AE 8D 35 09        STA XCRDL
08B1 AD 36 09        LDA XMIDH
08B4 E9 00           SBC #0
08B6 8D 34 09        STA XCRDH
08B9          ;
08B9 AD 35 09 NXPOINT LDA XCRDL    ;H<<-XCRD
08BC 8D 2E 09        STA HL
08BF AD 34 09        LDA XCRDH
08C2 8D 2D 09        STA HH
08C5          ;
08C5 AD 3D 09        LDA YCRD      ;V<<-YCRD
08C8 8D 31 09        STA V
08CB          ;
08CB 20 2C 08        JSR PLOTFOUR  ;PLOT SET OF POINTS
08CE          ;
08CE AD 39 09        LDA XYL       ;H<<-[XY-YCRD]
08D1 38              SEC
08D2 ED 3D 09        SBC YCRD
```

```
08D5 8D 2E 09                  STA HL
08D8 AD 38 09                  LDA XYH
08DB E9 00                     SBC #0
08DD 8D 2D 09                  STA HH
08E0               ;
08E0 AD 39 09                  LDA XYL          ;V<-[XY-XCRD]
08E3 38                        SEC
08E4 ED 35 09                  SBC XCRDL
08E7 8D 31 09                  STA V
08EA               ;
08EA 20 2C 08                  JSR PLOTFOUR     ;PLOT REMAINING POINTS
08ED               ;
08ED AD 3C 09                  LDA YMID         ;DIVISOR<-[YMID-YCRD]
08F0 38                        SEC
08F1 ED 3D 09                  SBC YCRD
08F4 85 1A                     STA DIVISOR
08F6               ;
08F6 AD 37 09                  LDA XMIDL        ;DIVIDEND<-[XMID-XCRD]*256
08F9 38                        SEC
08FA ED 35 09                  SBC XCRDL
08FD               ;
08FD A2 08                     LDX #8           ;BITCT<-8
08FF               ;
08FF A0 00                     LDY #0           ;CLEAR QUOTIENT
0901 84 1B                     STY QUOTIENT
0903               ;
0903 06 1B         DIVIDE1     ASL QUOTIENT
0905 2A                        ROL
0906 C5 1A                     CMP DIVISOR
0908 90 04                     BCC DIVIDE2
090A E5 1A                     SBC DIVISOR
090C E6 1B                     INC QUOTIENT
090E CA            DIVIDE2     DEX
090F D0 F2                     BNE DIVIDE1
0911               ;
0911 AD 3E 09                  LDA YCRDF        ;YCRDF<-[YCRDF-QUOTIENT]
0914 38                        SEC
0915 E5 1B                     SBC QUOTIENT
0917 8D 3E 09                  STA YCRDF
091A               ;
091A B0 03                     BCS CKX          ;YCRDY< -1?
091C CE 3D 09                  DEC YCRD         ;YES, YCRD<-[YCRD-1]
091F               ;
091F EE 35 09      CKX         INC XCRDL        ;XCRD<<-[XCRD+1]
0922 D0 03                     BNE CKX1
0924 EE 34 09                  INC XCRDH
0927               ;
0927 CE 30 09      CKX1        DEC R2           ;TALLY R2
092A 10 8D                     BPL NXPOINT      ;REPEAT UNTIL XCRD=XMID
092C 60                        RTS
092D               ;
092E               HH          DFS 1            ;HORIZONTAL PLOT VALUE
092F               HL          DFS 1
```

(Continued on page 28)

Circles (Listing continued, text begins on page 18)

Listing Two

```
0930   R         DFS 1          ;RADIUS
0931   R2        DFS 1          ;HOLDS R/SQR(2)
0932   V         DFS 1          ;VERTICAL PLOT
0933   X2H       DFS 1          ;HOLDS XMID*2
0934   X2L       DFS 1
0935   XCRDH     DFS 1          ;X COORDINATE
0936   XCRDL     DFS 1
0937   XMIDH     DFS 1          ;HORIZONTAL CENTER
0938   XMIDL     DFS 1
0939   XYH       DFS 1          ;HOLDS XMID+YMID
093A   XYL       DFS 1
093B   Y2H       DFS 1          ;HOLDS YMID*2
093C   Y2L       DFS 1
093D   YMID      DFS 1          ;VERTICAL CENTER
093E   YCRD      DFS 1          ;Y COORDINATE
093F   YCRDF     DFS 1          ;FRACTIONAL PART OF YCRD
093F   ;
001A   DIVISOR   EPZ $1A
C052   FULL      EQU $C052
C050   GRAPHICS  EQU $C050
00E4   HCOLOR    EPZ $E4
C057   HIRES     EQU $C057
F457   HPLOT     EQU $F457      ;APPLESOFT HIRES PLOT
00E6   HPAGE     EPZ $E6
00E1   LASTHH    EPZ $E1        ;HORIZ COORD OF LAST HPLOT
00E0   LASTHL    EPZ $E0
00E2   LASTV     EPZ $E2        ;VERT COORD OF LAST HPLOT
C054   PAGE1     EQU $C054
001B   QUOTIENT  EPZ $1B
093F   ;
093F             END

***** END OF ASSEMBLY
```

End Listing Two

Listing Three

```
10    '*** CIRCLE PLOT ***
20    '
30    INPUT "CENTER, RADIUS"; CX,CY,R: X=R: Y=0: A=-2*X+1:
      B=1: GOSUB 70: GOTO 30
40    '
50    'PLOT A POINT IN EACH OCTANT.
60    '
70    PSET(X+CX,Y+CY): PSET(Y+CX,X+CY): PSET(-Y+CX,X+CY):
      PSET(-X+CX,Y+CY): PSET(-X+CX,-Y+CY): PSET(-Y+CX,-X+CY):
      PSET(Y+CX,-X+CY): PSET(X+CX,-Y+CY)
80    '
90    'COMPUTE NEXT POINT.  F IS X^2+Y^2-R^2, A IS THE CHANGE
100   'IN X^2 WHEN X IS DECREMENTED BY 1, AND B IS THE CHANGE
110   'IN Y^2 WHEN Y IS INCREMENTED BY 1.  F IS NOT ALLOWED TO
120   'EXCEED  R; EQUIVALENTLY, THE POINT (X,Y) IS KEPT WITHIN
130   'A DISTANCE R+1/2 OF THE CIRCLE CENTER.  THE ALGORITHM
140   'IS DONE WHEN THE CHANGE IN Y^2 REACHES THE NEGATIVE OF
150   'THE CHANGE IN X^2 ( B>=-A ).
160   '
170   IF B>=-A THEN RETURN ELSE Y=Y+1: F=F+B: IF F>R THEN
      F=F+A: A=A+2: X=X+1
180   B=B+2: GOTO 70
```

End Listing Three

Circles

Listing Four

```
10 DEFINT A-Z
20 PRINT CHR$(26) 'CLEAR DUMB TTY SCREEN
50 FOR I=1 TO 11
55 AE=I*2 'WIDTH OF ELLIPSE
56 BE=I*1 'HEIGHT OF ELLIPSE
57 XC=I*4+1 'CENTER.X OF ELLIPSE
58 YC=I*1 'CENTER.Y OF ELLIPSE
60 GOSUB 1060 'PLOT A CIRCLE
70 NEXT I 'PLOT 11 CIRCLES
998 END
1050 '****** CIRCLE SUBROUTINE
1060 XF=0                    'INIT X-OFFSET
1070 YF=BE                   'INIT Y-OFFSET
1080 XD=BE*BE                'INIT COMPUTATION OF X-SQUARED
1090 YD=(2*BE-1)*AE*AE       'INIT COMPUTATION OF Y-SQUARED
1100 DX=2*BE*BE              'DEFINE DELTA-(X-SQUARED)
1110 DY=2*AE*AE              'DEFINE DELTA-(Y-SQUARED)
1120 ER=0                    'INIT ERROR (I.E. ER=AE^2*BE^2-XF^2*BE^2-YF^2*AE^2)
1130 GOSUB 1260              'PLOT THE FOUR POINTS
1140 TX=ER+XD
```

```
        : TY=ER-YD
        : TB=ER+XD-YD
1150 IF ABS(TX)>=ABS(TY) OR ABS(TX)>=ABS(TB) THEN 1170
1160 XF=XF+1
        : ER=TX
        : XD=XD+DX
        : GOTO 1220
1170 IF ABS(TY)>=ABS(TX) OR ABS(TY)>=ABS(TB) THEN 1190
1180 YF=YF-1
        : ER=TY
        : YD=YD-DY
        : GOTO 1220
1190 IF ABS(TB)>=ABS(TX) OR ABS(TB)>=ABS(TY) THEN 1210
1200 XF=XF+1
        : YF=YF-1
        : ER=TB
        : YD=YD-DY
        : XD=XD+DX
        : GOTO 1220
1210 PRINT"OOPS"; 'IF HERE THEN THERE IS A BUG.
1220 GOSUB 1260 'PLOT THE POINTS
1230 IF  YF<>0 THEN 1140
1240 RETURN
1250 '****** ROUTINE TO PLOT FOUR POINTS AT ONCE
1260 XP=XC+XF
        : YP=YC+YF
        : GOSUB 1320
1270 XP=XC+XF
        : YP=YC-YF
        : GOSUB 1320
1280 XP=XC-XF
        : YP=YC+YF
        : GOSUB 1320
1290 XP=XC-XF
        : YP=YC-YF
        : GOSUB 1320
1300 RETURN
1310 '****** ROUTINE TO PLOT A POINT ON A DUMB TERMINAL
1320 C1=YP+32
        : C2=XP+32
1330 IF YP<0 OR YP>23 OR XP<0 OR XP>79 THEN 1360
1340 PRINT CHR$(27);CHR$(61);CHR$(C1);CHR$(C2);"*";
1350 RETURN
1360 PRINT "POINT OUT OF BOUNDS"
        : STOP
```

End Listing Four