The Collision Counter

by Richard Cornelius and Melvin Zandler
Department of Chemistry
Wichita State University
Wichita, KS 67208

The collision counter is a powerful yet little-recognized tool for working with graphics on the Apple. It is one of those mysterious registers that occupies a place on many lists of zero page addresses, but no one ever seems to explain what it actually does. This article will describe what the collision counter is, how to use it, and (most importantly) what it is good for.

WHAT IS IT?

The collision counter is located in memory location **234 ($EA)**. Its value is set to **zero** at the start of each **DRAW** (or **XDRAW**) command. A DRAW command turns on a predefined series of bits called a **shape** on the high-resolution (hi-res) screen. **Whenever DRAW finds that a bit that it should turn on is already on, or that a bit that it should turn off is already off, the collision counter is incremented. PEEK**ing into **234** gives the number of "collisions" of the shape with images which are already on the hi-res screen.

WHAT IS IT GOOD FOR?

Now you know what the collision counter is. Understanding what it is, however, is of little value unless you also understand how it can be used. The following paragraphs describe two programs which give creative examples of how the collison counter can make graphics easier, faster, and more exciting.

The first example shows how figures can be quickly filled by **testing for the borders of the figure** with the collision counter. The second example is a simulation of radioactive decay in which the Hi-Res screen doubles as storage space to indicate which parts of the radioactive sample have already decayed.

FILLING FIGURES

Program **listing 1** gives the program **FILL FIGURES** which randomly HPLOTs a four-sided figure on the Hi-Res screen and then fills it (in color, of course). The part of the program that does the filling does not calculate where the edges of the figure are; **it simply DRAWS a one-bit shape repeatedly until it "collides" with the border** that was created with HPLOTs. After the figure is filled, the program erases it, HPLOTs another one, and fills it with a different color. The same techniques used in this program can be used to find intersection points of lines. (Mathematicians, take note.)

The INITIALIZATION part of FILL FIGURES first does the standard kinds of initialization that would be found in nearly any program that uses graphics. Next, several variables are defined. **MOVEREG** is the location of the subroutine which picks up the values in the registers that indicate where the next shape is to be drawn and places those values into **224-225** (for the **X-coordinate**) and **226** (for the **Y-coordinate**). **HY** is the position which contains the Y-coordinate after MOVEREG is called. **CLR** is the location of the routine that clears the current high resolution screen to black (**HCOLOR=0**). **TABLE** is the position of the shape table which is used in the program, but TABLE can also be used to relocate both the table and a machine language subroutine that the program calls. **C$** contains the name of the color that is used for filling the figure.

The section of the program called **POKES** puts the shape table and the short machine language routine into memory. The variables **MSB** and **LSB** are the most significant and least significant bytes of addresses. The address of the shape table is POKEd into **232** and **233** to tell Applesoft where to find the shape table. The POKEs into **FILL** and **FILL+1** tell the machine language routine where to find **shape number 1**.

A four-sided figure is randomly selected in the next section of the program. The figure is double plotted so that the lines are continuous (necessary so that the filling does not get out of the figure) and so that they are white rather that a mixed pattern of green and violet. The corners are selected so that they are in the general areas of the four corners of the screen, but plenty of freedom is allowed for variety. The point at which filling will begin must be within the figure, so **X0** and **Y0** are calculated as the averages of the **X** and **Y-coordinates** of the four corners of the figure. On the Apple the value of **X0** must be **even** for violet or blue colors and must be **odd** for green or orange.

An explanation of the subroutine at statement number **690** is the next logical step. First, a shape is plotted at the point **X,Y** defined by the body of the program. The statement calls the machine language routine that was POKEd into memory. This routine is only seven statements long, but it makes the program run more than seven times faster than if it were written all in Basic:

```
306-  A2 04      LDX  #$04
308-  A0 03      LDY  #$03
30A-  20 01 F6   JSR  $F601
30D-  A9 00      LDA  #$00
30F-  C5 EA      CMP  $EA
311-  F0 F3      BEQ  $0306
313-  60         RTS
```

The logic here is straightforward: The first two statements load the **X** and **Y** registers with the location of the shape table. The next statement jumps to the **DRAW** subroutine of Applesoft at **$F601**. **After the DRAW is completed, the accumulator is loaded with a zero and the value in $EA (the collision counter) is compared with it.** If the collision counter holds a **zero**, the process is repeated. The next shape is DRAWn where the previous one left off, so that the effect of the loop is to DRAW a line. If the collision counter indicates a collision, control is returned to Basic. Exactly the same operations could have been executed by the following BASIC statements:

```
740 REM
        DRAW SUBROUTINE
750 DRAW 1
760 IF PEEK (234) = 0 THEN 750
770 RETURN
```

If these BASIC statements are used then **statement 710** in the program should be changed to **GOSUB 740**.

The logic for filling the figure appears in **statements 430 to 550**. The values of **X** and **Y** are defined for the first DRAW. The POKE in **statement 460** sets the shape to be used as a single dot with upward movement. The subroutine then DRAWs the shape repeatedly until a collision is encountered. **The point of collision is recorded in TY.** Next the shape is changed to downward movement in **line 490**, and the subroutine is called again. If (in **line 500**) the DRAWing was stopped as soon as it started, then filling must be complete in the right direction, and the filling process is turned around by **line 530**. Otherwise, a new value of **Y** is chosen in between the top and bottom values of **Y** (**TY** and **BY**), and the filling to the right continues. The position for drawing lines is moved two dots at a time so that the filling will be in a color. If the value of **INC** in **line 440** is changed to **1**, then the filling will be in white (and will take twice as long).

The part of the program labelled PREPARE FOR NEXT CHARACTER simply changes the color and pauses. It also allows for a clean exit from the program with the **ESC** key. The HCOLOR for the DRAWing is always done in one of the white colors (HCOLOR = 3 or 7). The effect of color is achieved simply by choosing whether the lines will go in even or odd vertical positions.

RADIOACTIVE DECAY SIMULATION

In the RADIOACTIVE DECAY program, the Hi-Res screen is used both for graphic display and as a form of memory. The collision counter is used to "read" the contents of the screen memory to determine whether a particular portion of the sample has already decayed. The initialization sequence in the program pokes into memory a **shape** that is a **block of 12 dots**. During the operation of the program, an image of a radioactive sample is

The Collision Counter (Cont.)

shown on the screen, and the sample disappears in units of this shape. The lines labelled SET UP SCREEN put a set of axes and the sample block on the Hi-Res screen and place descriptive text at the bottom of the screen.

The heart of the program lies in the section of the program called **REACT**. The **TIME** is incremented each pass through the loop and serves as the value of the horizontal coordinate. Values of **X** and **Y** are randomly selected within the sample block, and a shape is drawn in black at that point. In **statement 490** the collision counter is checked. If it **equals 12** then black was found to already be present everywhere that the DRAW command tried to put black; that particular part of the block had already been erased. If the collision counter is **not equal to 12**, then part of the sample must have been erased with the preceding DRAW command, and a sound is made to simulate a Geiger counter. The plot is extended in **line 530**, and the **Y** value is incremented if a block had been erased. The value of the collision counter is **6** because the radioactive sample is colored, and therefore every other line is already black. The HPLOT is always extended one dot so that the plotted line will appear white. Note that **no array is needed to keep track of which parts of the block have been erased**. Through the use of the collision counter, the Hi-Res screen serves as the **memory** for that purpose.

The remainder of the program simply makes the program more useful in a classroom environment. The space bar can be used at any time to interrupt the PLOT and also to restart it, and the ESC key provides a clean exit from the program at any time. When the plot reaches the right edge of the screen, two options are provided. The space bar puts another plot on top of the previous one so that two plots can be easily compared. The letter "C" is used to clear the screen and begin again.

SUMMARY

The collision counter is a tool that seems at first inspection to be more interesting than useful. The examples presented here show that it is indeed a powerful tool for certain applications. The limitations in its use are only as restrictive as your imagination.

```
]LIST

1  REM  ************************
2  REM  *    FILL FIGURES      *
3  REM  * BY DICK CORNELIUS    *
4  REM  * COPYRIGHT (C) 1983   *
5  REM  * BY MICROSPARC, INC.  *
6  REM  * LINCOLN, MA. 01773   *
7  REM  ************************
160  REM  INITIALIZATION
170  HOME
180  HGR : SCALE= 1: ROT= 0: HCOLOR= 3
190 MOVEREG =  - 2613
200 HV = 226
210 CLR = 62450
220 TABLE = 768: REM THIS VALUE RELOCATES EVERYTHING
230 C$ = "GREEN"
240  REM  POKES
250 SHAPE = TABLE + 4:FILL = TABLE + 6
260  FOR SPOT = (TABLE) TO TABLE + 19: READ CODE: POKE
     SPOT,CODE: NEXT
270  DATA 1,0,4,0,4,0,162,4,160,3,32,1,246,169,0,197,2
     34,240,243,96
280 MSB =  INT (TABLE / 256):LSB = TABLE - 256 * MSB
290  POKE 232,LSB: POKE 233,MSB
300 MSB =  INT (SHAPE / 256):LSB = SHAPE - 256 * MSB
310  POKE FILL + 1,LSB: POKE FILL + 3,MSB
320  REM  HPLOT BOX
330  HOME : CALL CLR
340 X(1) =  RND (1) * 100:X(4) =  RND (1) * 100
350 X(2) =  RND (1) * 100 + 170:X(3) =  RND (1) * 100 +
    170
360 Y(1) =  RND (1) * 70:Y(2) =  RND (1) * 70
370 Y(3) =  RND (1) * 70 + 90:Y(4) =  RND (1) * 70 + 9
    0
380  HPLOT X(1),Y(1) TO X(2),Y(2) TO X(3),Y(3) TO X(4)
    ,Y(4) TO X(1),Y(1)
390  HPLOT X(1) + 1,Y(1) TO X(2) + 1,Y(2) TO X(3) + 1,
    Y(3) TO X(4) + 1,Y(4) TO X(1) + 1,Y(1)
400 X0 = (X(1) + X(2) + X(3) + X(4)) / 4:Y0 = (Y(1) +
    Y(2) + Y(3) + Y(4)) / 4
410 X0 = 2 * ( INT (X0 / 2))
420  IF C$ = "GREEN" OR C$ = "ORANGE" THEN X0 = X0 + 1

430  REM  FILL
440 X = X0:INC = 2
450 Y = Y0
460  POKE TABLE + 4,4: GOSUB 690
470  IF  ABS (CY - Y) < 2 THEN 530
480 TY = CY
490  POKE TABLE + 4,6: GOSUB 690
500  IF  ABS (CY - Y) < 3 THEN 530
510 BY = CY:Y = (TY + BY) / 2
520 X = X + INC: IF X < 280 AND X > 0 THEN 460
530  IF INC > 0 THEN INC =  - INC:X = X0 + INC: GOTO 4
    50
540  IF  ABS (CY - Y) < 3 THEN 560
550 X = X - 1: IF X > 0 THEN 490
560  REM  PREPARE FOR NEXT FIGURE
570  IF C$ = "GREEN" THEN C$ = "VIOLET": GOTO 610
580  IF C$ = "VIOLET" THEN C$ = "ORANGE": GOTO 610
590  IF C$ = "ORANGE" THEN C$ = "BLUE": GOTO 610
600  IF C$ = "BLUE" THEN C$ = "GREEN"
610  HCOLOR= 3: IF C$ = "BLUE" OR C$ = "ORANGE" THEN  HCOLOR= 7
620  FOR PAUSE = 1 TO 100
630  IF  PEEK ( - 16384) < 127 THEN 670
640  GET G$
650  IF G$ = " " THEN I = 200
660  IF G$ =  CHR$ (27) THEN  HOME : TEXT : HOME : END

670  NEXT
680  GOTO 320
690  REM  SUBROUTINE
700  DRAW 1 AT X,Y
710  CALL FILL
720  CALL MOVEREG:CY =  PEEK (HY)
730  RETURN
```

```
]LIST

1  REM  ************************
2  REM  *  RADIOACTIVE DECAY   *
3  REM  * BY DICK CORNELIUS    *
4  REM  * AND MELVIN ZANDLER   *
5  REM  * COPYRIGHT (C) 1983   *
6  REM  * BY MICROSPARC, INC.  *
7  REM  * LINCOLN, MA. 01773   *
8  REM  ************************
150  REM  INITIALIZATION
160  HGR : SCALE= 1: ROT= 1:
170  FOR SPOT = 960 TO 970: READ CODE: POKE SPOT,CODE:
     NEXT
180  DATA 1,0,4,0,45,53,63,55,45,45,0
190  POKE 233,3: POKE 232,192
200  HOME
210 SPEAKER = 49200
220 CC = 234
230 KB =  - 16384:KB =  - 16368
240 ESC$ =  CHR$ (27)
250 TXT$ = "SIMULATION OF RADIOACTIVE DECAY SHOWING S
    AMPLE AND PLOT AS A FUNCTION OF TIME   PRESS THE
    SPACE BAR TO INTERRUPT PLOT."
260  REM  SET UP SCREEN
270  HCOLOR= 3
280  FOR X = 208 TO 262: HPLOT X,2 TO X,64: NEXT
290  HCOLOR= 2
300  FOR X = 215 TO 255: HPLOT X,9 TO X,56: NEXT
310  HCOLOR= 0
320  HPLOT 214,8 TO 256,8 TO 256,56 TO 214,56 TO 214,8

330  HCOLOR= 2
340  HPLOT 0,0 TO 0,159
350  HPLOT 0,0 TO 279,0
360  HPLOT 0,159 TO 279,159
370  FOR X = 2 TO 279 STEP 20
380  FOR Y = 158 TO 0 STEP  - 20
390  HPLOT X,Y: NEXT : NEXT
400  VTAB 21: PRINT TXT$;
410  REM  REACT
420  POKE KS,0
430 TIME = TIME + 1 / 4
440  IF  PEEK (KB) > 127 THEN  GOSUB 680
450 X =  INT ( RND (1) * 10) * 4 + 216
460 Y =  INT ( RND (1) * 16) * 3 + 9
470  IF  RND (1) > 0.5 THEN  FOR PAUSE = 1 TO 10: NEXT
     : GOTO 430
480  HCOLOR= BLACK: DRAW 1 AT X,Y
490  IF  PEEK (CC) = 12 THEN  FOR PAUSE = 1 TO 10:SOUN
     D =  PEEK (NULL): NEXT : GOTO 510
500  FOR I = 1 TO 10:SOUND =  PEEK (SPEAKER): NEXT
510  HCOLOR= 3
520  IF TIME > 278 THEN 570
530  HPLOT OX,OY TO TIME;YY TO TIME + 1,YY
540  IF  PEEK (CC) = 6 THEN YY = YY + 1
550 OX = TIME:OY = YY
560  GOTO 430
570  REM  START AGAIN?
580  HOME
590  POKE KS,0
600  VTAB 22: PRINT "PRESS THE SPACE BAR TO PLOT AGAIN
     ."
610  PRINT "TYPE 'C' TO CLEAR AND RUN AGAIN.";
620  IF  PEEK (KB) < 128 THEN 620
630  GET G$
640  IF G$ = " " THEN  RUN 200
650  IF G$ = "C" THEN  RUN
660  IF G$ = ESC$ THEN  HOME : TEXT : HOME : END
670  GOTO 620
680  REM  INTERRUPT PLOT
690  GET G$
700  IF G$ = ESC$ THEN  GOTO 660
710  HOME : VTAB 22
720  PRINT "PRESS THE SPACE BAR AGAIN TO CONTINUE."
730  IF G$ <  > " " THEN  RETURN
740  IF  PEEK (KB) < 128 THEN 740
750  GET G$
760  IF G$ <  > " " THEN 740
770  VTAB 21: PRINT TXT$;
780  RETURN
```