Squirm Attack

Block shape animation lets you move flicker free across the screen, but what if you want your shape to wriggle as it moves? SQUIRM ATTACK describes pre-shifted animation which lets you do just that.

by Robert R. Devine
P.O. Box 10
Adona, AK 72001

Boy, time really flies when you're having fun! Here it is — time to get back to the fun of making all those great little shapes cavort about the Hi-Res screen.

Before we get into this month's discussion of graphics animation (which I personally think will be the best yet), let's take a moment to review the different types of animation that we've looked at so far. To date, we've experimented with many approaches to animation, each with its own strengths and weaknesses.

[redacted]

ANIMATION METHODS TO DATE

1. The most basic form of animation is the old **ERASE-MOVE-DRAW method**, and this is probably the first type that you experimented with. This works well; however, you need to keep track of your shape's **old** position for the ERASE, and you'll usually see quite a bit of flicker because the shape is only **on the screen** about half of the time.

2. To overcome the flicker problem, we next looked at **page flip animation** to prevent the viewer from ever seeing the shape in its ERASE state. This was a big improvement, as our shapes now began to move very smoothly about the screen. The major problem was that it was necessary to use **both Hi-Res screens**, thus reducing the amount of memory available for shapes and programs by 8K. While the results were better, our programs ran slower because it was necessary to do all our DRAWing operations twice, once on each page.

3. By using horizontal **bit shifting** and vertical **byte shifting**, we began to develop routines that not only ran smoothly without any flicker, but could work quickly as well, since no ERASE actions were needed. Using these methods, we were again able to use only one Hi-Res page. At this point it seemed as though we had finally achieved the best possible animation methods.

Each of the above methods has its own benefits and you'll probably find that one of them will solve almost any animation problem that you encounter. However, they all have one major weakness.

Methods 1 through 3 only deal with static shapes, which simply means that while we may have effective ways of moving our shapes, the shape itself never changes as we move it.

So now we come to another problem: **How can we move and animate our shapes at the same time?** An example of this might be to display a man walking across the screen while bouncing a ball, or perhaps it would be nice to have blinking lights on the spaceship that we're moving around on the screen. None of the methods that we've looked at so far provide a viable way to achieve this type of animation. It is this type of animation that we'll examine now.

INTRODUCING PRE-SHIFTED SHAPE ANIMATION

As we've done in the past, we'll work with some actual examples of the techniques described; in this case, a preliminary test routine that I developed for a program I'm writing, tentatively named SQUIRM ATTACK. The friendly little creatures that we'll attempt to bring to life on your Hi-Res screen could be described as many things, including the creators of Mankind itself!!

These creatures are called SQUIRMS, and as their name implies, they love to squirm about, constantly wiggling their long tails. Since they are also quite aggressive, their mouths are always opening and closing, looking for something tasty to nibble on (no pun intended). For your first look at our shapes, refer to **FIGURE 1**.

When working with **pre-shifted** shapes, we will work exclusively with our block shape DRAW routine. By properly defining our shapes, it will never be necessary to worry about ERASEing as we animate

continued on next page

RUNNING THE DEMONSTRATION PROGRAM

Now let's take this idea, as well as our SQUIRMS, and see how we might use them in a working routine. As mentioned earlier, what we'll look at is the preliminary test of a routine that will be incorporated into a program I'm working on. The finished program will be all machine code. However, for now we'll use an Applesoft CALLing program to get things moving along. You could easily adapt what we'll work with for your own purposes by creating a different set of shapes.

In our test we'll animate 20 shapes on the screen, causing them to move up, down, or forward in a constant state of activity, squirming about. While we won't move them diagonally, we will show how this can be done.

Let's first look at **FIGURE 2** to see how we'll arrange our shapes. There will be three columns of SQUIRMS with columns 1 and 3 having seven shapes per column, and column 2 having six shapes.

The VT value for each shape will be related to a variable that I've called TOPLMT (TOPLIMIT), with each shape being 20 dots below the shape directly above it. The HR and HL values for each column will be kept track of by variables FR (FrontRight), FL (FrontLeft), MR (MiddleRight), ML (MiddleLeft), etc.

We'll also use a variable called X that will keep track of which SQUIRM we're dealing with (1-20) so that we can check the **status** of each SQUIRM to determine if it has been destroyed. (We won't actually destroy the SQUIRMS, but it is a part of what the finished program will do.)

Let's take a few moments to enter the Shape Tables **(LISTING 1)**, Applesoft test program **(LISTING 2)**, and the machine code file **(LISTING 3)** which I've called T5.OBJ. (You can save T5.OBJ with the command **BSAVE T5.OBJ,A$5000,L$F3**.) You will of course need to have your **block shape routines** (documented in past columns; a hex dump appears in **LISTING 4**) on the same disk. Once entered, let it run for a bit to see how it works and we'll go through it to look at how the animation is accomplished.

HOW THE PROGRAM WORKS

Now that you've spent a little time watching all the activity on the screen, let's see how it all works, beginning with the Applesoft CALLing program:

Lines 5-16 should be pretty clear so we won't bother with them.

Line 20 sets all of the bytes in the Squirm Status Table equal to 1. While we won't actually use the Status Table in our test, the machine code routine will check the table to see if a SQUIRM has been destroyed prior to each DRAW operation. If, for example, you were to set the fourth address in the table ($8204) to **zero**, you would find that SQUIRM #4 would be eliminated from the screen.

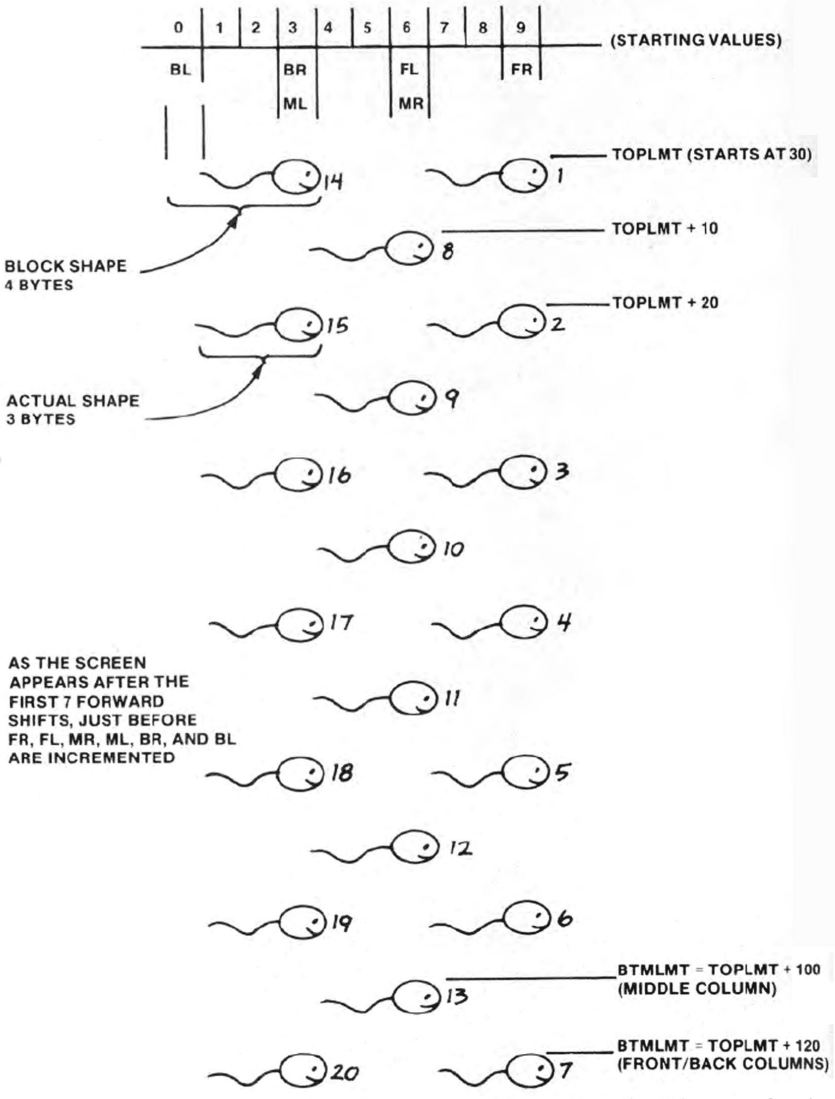
Line 30 disables the EOR function of DRAW which is what you will normally do when working with pre-shifted shapes.

Line 100 sets SHNUM equal to the **high byte** value of the memory page where our **horizontal** (pre-shifted) shapes are located ($8000), and clears the Hi-Res screen.

Line 190 POKEs the TOPLMT value into address 43 ($2B) for use by the machine code routine. You'll note from **FIGURE 2** that our colony of SQUIRMS always begins with the VT of the topmost row of shapes at Y = 30.

Line 200 makes the first CALL to the machine code routine. This CALL sets the proper starting HR and HL values for each column, sets the first direction of movement forward, and moves the shapes forward seven dots.

FIGURE 2



Line 210 selects the random movement of the shapes as they move about the screen. To slow down their forward progress (to keep them on the screen longer), we will only move forward one-ninth of the time.

Line 300 is the beginning of our MOVE UP subroutine. First, address 43 ($2B), TOPLMT is checked to prevent movement upward past VT=16. If we are already as far up as we want to go, then a jump to **line 322** is made to move downward instead.

Line 302 sets SHNUM for our **vertical** movement shapes which begin at $8100. We haven't talked about these shapes yet, so let's do so now.

As you look at **FIGURE 1**, you'll see that while each of the shapes is the same as its counterpart **horizontal** movement shape, none of these shapes are **pre-shifted**, and each shape is only three bytes wide. Since the shapes simply move up or down, we don't want them to move forward as we step through the series. By making them only three bytes wide, we also make them execute 25% faster.

The final thing that you should notice about the shapes is that each has an extra row of empty bytes above and below the actual shape bytes. This serves the same function as the extra rows of bytes we used in our vertical SHIFTing routines and takes care of **erasing** the old shape as we move.

The way our routine is written, we only need to **POKE SHNUM** with the page number (high byte) where the shapes are found. By properly arranging the shape series on each memory page, we can let the machine code routine take care of assigning the proper low byte shape address value.

Line 305 sets the desired direction of movement and CALLs the machine code routine, which animates our shapes on the screen. Address 235 ($EB) is where we store DPTR (Direction PoinTeR); 0=Move Right, 1=Move Up, and 2=Move Down.

Lines 310 and 315 work the same as **lines 300-305**, except that we first **GOSUB 400** to test the present value of FR (Front column, Rightmost byte) to see if we can still move forward and stay on the screen. If we've reached the right edge of the screen, the two GOSUBs that got us there are cancelled and we jump to **line 100** to begin all over again.

Lines 320-325 work the same as **lines 300-305** by testing how far down we can go, and setting our **vertical** movement shapes and direction of travel.

THE MACHINE CODE ROUTINE T5.OBJ

We won't go through this routine in great detail, as it's heavily documented. However, we will hit the important parts so you can see how one routine can move our shapes in any of three selected directions.

First, note the way we've arranged our Shape Tables in memory: in both sets of shapes, the first shape is located at $SHNUM(20), with each progressive shape being $20 bytes higher than the last, and the last shape in each table being at $SHNUM(E0). Therefore, to find the address of each succeeding shape, all we need to do is add $20 to the present shape **low byte**.

To determine if we're through all seven shapes in the series, we simply check to see if we've reached the shape low byte $E0. This is how we step through our shapes. First we **POKE 251,SHNUM** (high byte). Then we insert the proper low byte directly into the second byte, $9300, of the DRAW routine. Every time we complete a series of seven shapes, the DRAW routine is restored to LDA #$00, so it will work properly for other program DRAWing needs.

The next and probably most important thing to be aware of is the status of our **shifting bytes** when we enter or leave the routine. There is **always an empty shifting byte ahead of the shape when we call the routine**. If we enter the routine to **move forward** with the pre-shifted shapes, all we need to do is step through the seven shapes and INCrement FR, FL, MR, ML, BR, and BL when we leave.

If, however, we are entering to move up or down, we need to DECrement FR, MR, and BR before doing any drawing to remove the shifting bytes (remember, the **vertical** shapes are only three bytes wide), and restore the shifting bytes as we leave the routine.

Vertical shape movement is handled much the same as in vertical shift animation. If you'll look at **lines 4221-4230**, you'll see that it is here we test DPTR (Direction PoinTeR) to see where we're going.

If we're moving rightward we just move to the next shape. However, if we're moving up or down, we need to DECrement or INCrement TOPLMT before going on to the next shape in the series. Every time we go up one line we must DECrement TOPLMT, and every time we go down we must INCrement it. That's really all the difference there is between moving up and down!!

"...moving diagonally should be a breeze."

DIAGONAL MOVEMENT WITH PRE-SHIFTED SHAPES

We haven't moved any of our shapes diagonally in this test; but if you understand what we've done so far, then moving diagonally should be a breeze. Consider this: If we were to use our set of pre-shifted shapes in the **vertical** movement routines, what would happen?

THAT'S CORRECT!! As you moved your shape up or down by INCrementing or DECrementing the shapes VT and VB with each new shape in the series, you would also move the shape forward!! Therefore, moving a pre-shifted shape up or down results in diagonal movement.

The only reason that you can't do this in our test routine is that T5.OBJ expects any **vertical** movement shape to be one byte narrower than our **horizontal** movement shapes. Removal of **lines 3372-3379 and 4240-4242** would allow you to substitute our pre-shifted shapes into your vertical movement routines, thus providing for diagonal movement.

A CLOSING NOTE

I hope that you now have a pretty clear idea about how to make your shapes move smoothly and how to animate them as they move.

See you next month!!

LISTING 1 BLOCK/SQUIRMS-5

```
8000.81FF
8000- 00 00 00 00 00 00 07 70
8008- 3F 7E 7F 7F 78 0F 47 71
8010- 3F 7E 7F 7F 7F 7F 3F 7E
8018- 07 70 00 00 00 00 00 00
8020- 00 00 00 00 00 18 00 00
8028- 00 0C 00 78 01 7F 47 4E
8030- 00 4C 6C 00 00 78 38 00
8038- 00 00 00 00 00 00 00 00
8040- 00 00 00 00 00 70 00 00
8048- 00 18 78 00 03 7F 4E 1C
8050- 01 18 03 30 01 70 01 60
8058- 00 00 00 00 00 00 00 00
8060- 00 00 00 00 03 61 70 00
8068- 00 33 18 00 07 7E 0C 18
8070- 02 30 06 30 03 60 03 60
8078- 00 00 00 00 00 00 00 00
8080- 00 00 00 00 07 41 60 00
8088- 01 63 30 00 0F 7E 1C 70
8090- 04 60 07 40 07 40 00 00
8098- 00 00 00 00 00 00 00 00
80A0- 00 00 00 00 0F 00 0E 00
80A8- 07 40 1B 00 1F 7C 71 60
80B0- 09 47 40 00 0F 00 00 00
80B8- 00 00 00 00 00 00 00 00
80C0- 00 00 00 00 1E 00 1E 00
80C8- 1F 00 33 00 3F 70 61 40
80D0- 13 19 40 00 1E 0F 00 00
80D8- 00 00 00 00 00 00 00 00
80E0- 00 00 00 00 04 00 38 00
80E8- 06 00 6C 00 7F 63 47 00
80F0- 26 36 00 00 3C 1C 00 00
80F8- 00 00 00 00 00 00 00 00
8100- 00 00 00 00 00 00 00 00
8108- 00 00 00 00 00 00 00 00
8110- 00 00 00 00 00 00 00 00
8118- 00 00 00 00 00 00 00 00
8120- 00 00 00 0C 00 00 06 00
8128- 3C 7F 63 67 26 36 00 3C
8130- 1C 00 00 00 00 00 00 00
8138- 00 00 00 00 00 00 00 00
8140- 00 00 00 1C 00 00 06 1E
8148- 00 7F 73 47 26 00 6C 3C
8150- 00 38 00 00 00 00 00 00
8158- 00 00 00 00 00 00 00 00
8160- 00 00 00 3C 1E 00 06 33
8168- 00 7F 61 43 26 00 66 3C
8170- 00 3C 00 00 00 00 00 00
8178- 00 00 00 00 00 00 00 00
8180- 00 00 00 3C 0E 00 0E 1B
8188- 00 7F 71 67 26 00 3C 3C
8190- 00 00 00 00 00 00 00 00
8198- 00 00 00 00 00 00 00 00
81A0- 00 00 00 3C 00 38 1E 00
81A8- 6C 7F 73 47 26 1E 00 3C
81B0- 00 00 00 00 00 00 00 00
81B8- 00 00 00 00 00 00 00 00
81C0- 00 00 00 3C 00 3C 3E 00
81C8- 66 7F 61 43 26 33 00 3C
81D0- 1E 00 00 00 00 00 00 00
81D8- 00 00 00 00 00 00 00 00
81E0- 00 00 00 04 00 38 06 00
81E8- 6C 7F 63 47 26 36 00 3C
81F0- 1C 00 00 00 00 00 00 00
81F8- 00 00 00 00 00 00 00 00
```

LISTING 2 — T5 TEST

```
1  REM  **********************
2  REM  *      T5  TEST       *
3  REM  *  BY ROBERT DEVINE   *
4  REM  * COPYRIGHT (C) 1983 *
5  REM  * BY MICROSPARC, INC *
6  REM  * LINCOLN, MA. 01773 *
7  REM  **********************
8  TEXT : HOME
9  VTAB 12: PRINT "** COPYRIGHT 1983 BY MICROSPARC, I
   NC. **"
10  PRINT  CHR$ (4)"BLOAD BLOCK ROUTINES $90AA": CALL
   37799
15  PRINT  CHR$ (4)"BLOAD BLOCK/SQUIRMS-5"
16  PRINT  CHR$ (4)"BLOAD T5.OBJ"
20  FOR X = 33280 TO 33311: POKE X,1: NEXT : REM  Set
    Squirm status pointers at $8200
30  POKE 37696,234: POKE 37697,234: REM  Cancel DRAW
   EOR function
100  POKE 251,128: HOME : HGR :X =  PEEK (49234)
190  POKE 43,30: REM   Set TOPLMT (Columns 1 and 3 be
   gin at Y=30)
200  CALL 20480: REM   DRAW Squirms at starting posit
   ions
210  ON ( INT ( RND (1) * 9) + 1) GOSUB 300,300,300,3
   00,310,320,320,320,320: REM  UP,RIGHT,DOWN
230  GOTO 210
300  IF  PEEK (43) <  = 16 THEN 322: REM  Test TOPLMT
    for moving up
302  POKE 251,129: REM  Set NON-Shifted shapes
305  POKE 235,1: CALL 20502: RETURN : REM  Move up
310  GOSUB 400: POKE 251,128: REM  Set Shifted shapes

315  POKE 235,0: CALL 20502: RETURN : REM  Move right
    -->
320  IF  PEEK (43) >  = 44 THEN 302: REM  Test TOPLMT
    for moving down
322  POKE 251,129: REM  Set NON-Shifted shapes
325  POKE 235,2: CALL 20502: RETURN : REM  Move down
400  IF  PEEK (26) >  = 39 THEN  POP : POP : GOTO 100
   : REM  At right edge of screen-start over again
405  RETURN
```

LISTING 3 T5.OBJ

```
:ASM
                1000  .OR $5000
                1010  .TF T5.OBJ
932F-           1020 DRAW .EQ $932F
9330-           1030 SHPLO .EQ $9330
001A-           1040 FR .EQ $1A
001B-           1050 FL .EQ $1B
001C-           1060 MR .EQ $1C
001E-           1070 ML .EQ $1E
00D6-           1080 BR .EQ $D6
00D7-           1090 BL .EQ $D7
00FB-           1095 SHNUM .EQ $FB
00FC-           1100 VT .EQ $FC
00FD-           1110 VB .EQ $FD
00FE-           1120 HR .EQ $FE
00FF-           1130 HL .EQ $FF
0019-           1140 LOOP .EQ $19
002B-           1160 TOPLMT .EQ $2B
0007-           1170 BTMLMT .EQ $07
8200-           1180 STATUS .EQ $8200
0022-           1190 X .EQ $22
00EB-           1200 DPTR .EQ $EB
5000- A9 09     3000 START LDA #9    * FIRST DRAW ENTRY
 5002- 85 1A    3010  STA FR
 5004- A9 06    3020  LDA #6         * SET ALL
 5006- 85 1B    3030  STA FL
 5008- 85 1C    3040  STA MR         * STARTING
 500A- A9 03    3050  LDA #3
 500C- 85 1E    3060  STA ML         * HR'S AND HL'S
 500E- 85 D6    3070  STA BR
 5010- A9 00    3080  LDA #0         * FOR SQUIRMS
 5012- 85 D7    3090  STA BL
 5014- 85 EB    3100  STA DPTR       * SET DIRECTION -->
 5016- A9 20    3360 L205 LDA #$20 ** NORMAL DRAW
 5018- 8D 30 93 3370  STA SHPLO      * SET STARTING SHAPE
 501B- A4 EB    3372  LDY DPTR       * MOVING --> ?
 501D- F0 06    3374  BEQ L220       * YES-JUMP
 501F- C6 1A    3376  DEC FR         * REMOVE THE
 5021- C6 1C    3378  DEC MR         * SHIFTING
 5023- C6 D6    3379  DEC BR         * BYTES.
 5025- A2 01    3380 L220 LDX #1     * POINT TO
 5027- 86 22    3390  STX X          * FIRST SQUIRM.
 5029- A5 1A    3400  LDA FR         * GET ITS HR
 502B- 85 FE    3410  STA HR         * STORE IT
 502D- A5 1B    3420  LDA FL         * GET ITS HL
 502F- 85 FF    3430  STA HL         * STORE IT
 5031- A5 2B    3440  LDA TOPLMT     * FIND TOPLIMIT
 5033- 18       3450  CLC
 5034- 69 8C    3460  ADC #140       * ADD OFFSET
 5036- 85 07    3470  STA BTMLMT     * SET BOTTOMLIMIT
 5038- A5 2B    3480  LDA TOPLMT     * GET TOPLIMIT
 503A- 85 19    3490  STA LOOP       * SET VT POINTER
 503C- 85 FC    3500 J1 STA VT       * SET CURRENT VT
 503E- 18       3510  CLC
 503F- 69 06    3520  ADC #6         * ADD OFFSET
 5041- 85 FD    3530  STA VB         * SET VB
 5043- A6 22    3540  LDX X          * GET SQUIRM POINTER
 5045- BD 00 82 3550  LDA STATUS,X   * IS SQUIRM DEAD?
 5048- F0 03    3560  BEQ J4         * YES-ABORT DRAW
 504A- 20 2F 93 3570 Z1 JSR DRAW     * DRAW SQUIRM
 504D- E6 22    3580 J4 INC X        * NEXT SQUIRM
 504F- A5 19    3590  LDA LOOP       * GET VT POINTER
 5051- 18       3600  CLC
 5052- 69 14    3610  ADC #20        * ADD OFFSET
 5054- 85 19    3620  STA LOOP       * RESET VT POINTER
 5056- C5 07    3630  CMP BTMLMT     * COLUMN DONE ?
 5058- 90 E2    3640  BCC J1         * NO-NEXT SQUIRM
 505A- A5 1C    3650 L230 LDA MR     * GET MIDDLE HR
 505C- 85 FE    3660  STA HR         * STORE IT
 505E- A5 1E    3670  LDA ML         * GET MIDDLE HL
 5060- 85 FF    3680  STA HL         * STORE IT
 5062- A5 2B    3690  LDA TOPLMT     * GET TOPLIMIT
 5064- 18       3700  CLC
 5065- 69 78    3710  ADC #120       * ADD OFFSET
 5067- 85 07    3720  STA BTMLMT     * SET BOTTOMLIMIT
 5069- A5 2B    3730  LDA TOPLMT     * GET TOPLIMIT
 506B- 18       3740  CLC
 506C- 69 0A    3750  ADC #10        * ADD OFFSET
 506E- 85 19    3760  STA LOOP       * SET VT POINTER
 5070- 85 FC    3770 J2 STA VT       * SET CURRENT VT
 5072- 18       3780  CLC
 5073- 69 06    3790  ADC #6         * ADD OFFSET
 5075- 85 FD    3800  STA VB         * SET VB
 5077- A6 22    3810  LDX X          * GET SQUIRM POINTER
 5079- BD 00 82 3820  LDA STATUS,X   * IS IT DEAD ?
 507C- F0 03    3830  BEQ J5         * YES-ABORT DRAW
 507E- 20 2F 93 3840 Z2 JSR DRAW     * DRAW SQUIRM
 5081- E6 22    3850 J5 INC X        * NEXT SQUIRM
 5083- A5 19    3860  LDA LOOP       * GET VT POINTER
 5085- 18       3870  CLC
 5086- 69 14    3880  ADC #20        * ADD OFFSET
 5088- 85 19    3890  STA LOOP       * SET NEXT VT
 508A- C5 07    3900  CMP BTMLMT     * COLUMN DONE ?
 508C- 90 E2    3910  BCC J2         * NO-NEXT SQUIRM
 508E- A5 D6    3920 L240 LDA BR     * GET BACK HR
 5090- 85 FE    3930  STA HR         * STORE IT
 5092- A5 D7    3940  LDA BL         * GET BACK HL
 5094- 85 FF    3950  STA HL         * STORE IT
 5096- A5 2B    3960  LDA TOPLMT     * GET TOPLIMIT
 5098- 18       3970  CLC
 5099- 69 8C    3980  ADC #140       * ADD OFFSET
 509B- 85 07    3990  STA BTMLMT     * SET BOTTOMLIMIT
 509D- A5 2B    4000  LDA TOPLMT     * GET TOPLIMIT
 509F- 85 19    4010  STA LOOP       * SET VT POINTER
 50A1- 85 FC    4020 J3 STA VT       * SET CURRENT VT
 50A3- 18       4030  CLC
 50A4- 69 06    4040  ADC #6         * ADD OFFSET
 50A6- 85 FD    4050  STA VB         * SET VB
 50A8- A6 22    4060  LDX X          * GET SQUIRM POINTER
 50AA- BD 00 82 4070  LDA STATUS,X   * IS IT DEAD ?
 50AD- F0 03    4080  BEQ J6         * YES-ABORT DRAW
 50AF- 20 2F 93 4090 Z3 JSR DRAW     * DRAW SQUIRM
 50B2- E6 22    4100 J6 INC X        * NEXT SQUIRM
 50B4- A5 19    4110  LDA LOOP       * GET VT POINTER
 50B6- 18       4120  CLC
 50B7- 69 14    4130  ADC #20        * ADD OFFSET
 50B9- 85 19    4140  STA LOOP       * SET NEXT VT
 50BB- C5 07    4150  CMP BTMLMT     * COLUMN DONE ?
 50BD- 90 E2    4160  BCC J3         * NO-NEXT SQUIRM
 50BF- AD 30 93 4170 L250 LDA SHPLO  * GET SHAPE POINTER
 50C2- C9 E0    4180  CMP #$E0       * DONE SHAPE 7 ?
 50C4- F0 17    4190  BEQ L260       * YES-WE'RE DONE
 50C6- 18       4200  CLC
 50C7- 69 20    4210  ADC #$20       * SET NEXT SHAPE
 50C9- 8D 30 93 4220  STA SHPLO      * STORE IT
 50CC- A5 EB    4221  LDA DPTR       * MOVING --> ?
 50CE- F0 0A    4222  BEQ J8         * YES-JUMP
 50D0- C9 01    4223  CMP #1         * MOVING UP ?
 50D2- D0 04    4224  BNE J9         * NO-JUMP
 50D4- C6 2B    4225  DEC TOPLMT     * MOVE ALL UP 1 LINE
 50D6- D0 02    4226  BNE J8         * JUMP
 50D8- E6 2B    4227 J9 INC TOPLMT    * MOVE ALL DOWN 1 LINE
 50DA- 4C 25 50 4230 J8 JMP L220     * AGAIN WITH NEXT SHAPE
 50DD- A5 EB    4240 L260 LDA DPTR   * GET DIRECTION POINTER
 50DF- D0 06    4242  BNE J10        * IF UP OR DOWN-JUMP
 50E1- E6 1B    4245  INC FL         * MOVE ALL HL'S
 50E3- E6 1E    4250  INC ML         * RIGHTWARD
 50E5- E6 D7    4260  INC BL         * 1 BYTE
 50E7- E6 1A    4270 J10 INC FR      * MOVE ALL HR'S
 50E9- E6 1C    4280  INC MR         * RIGHTWARD
 50EB- E6 D6    4290  INC BR         * 1 BYTE
 50ED- A9 00    4300  LDA #0         * RESTORE DRAW
 50EF- 8D 30 93 4310  STA SHPLO      * TO LDA #$0
 50F2- 60       4320  RTS            * SHAPE MOVED 7 DOTS
```

LISTING 4

BLOCK ROUTINES $90AA

```
*90AA.9606
90AA- A5 FD C9 BD B0 2F
90B0- 85 06 20 91 93 A4 FE B1
90B8- 26 85 F9 20 04 F5 A5 F9
90C0- 91 26 20 D5 F4 88 18 C0
90C8- FF F0 04 C4 FF B0 E8 C6
90D0- 06 A5 06 C9 FF F0 04 C5
90D8- FC B0 D7 E6 FC E6 FD 60
90E0- A5 FC C9 01 90 33 E6 FD
90E8- 85 06 20 91 93 A4 FE B1
90F0- 26 85 F9 20 D5 F4 A5 F9
90F8- 91 26 20 04 F5 88 18 C0
9100- FF F0 04 C4 FF B0 E8 E6
9108- 06 A5 06 C9 BE F0 04 C5
9110- FD 90 D7 C6 FD C6 FD C6
9118- FC 60 A9 0E A6 FB 9D 6F
9120- 8F 60 A9 00 8D 54 C0 A9
9128- 40 85 E6 60 A9 00 8D 55
9130- C0 A9 20 85 E6 60 20 22
9138- 91 18 90 03 20 2C 91 20
9140- 0E 92 20 0E 92 20 7A 91
9148- 20 0E 92 20 0E 92 20 7A
9150- 91 A5 E6 C9 40 F0 E5 60
9158- 20 22 91 18 90 03 20 2C
9160- 91 20 B5 91 20 B5 91 20
9168- 8A 91 20 B5 91 20 B5 91
9170- 20 8A 91 A5 E6 C9 40 F0
9178- E5 60 A6 FB DE 6F 8F D0
9180- 08 20 AC 91 A9 0E 9D 6F
9188- 8F 60 A6 FB DE 6F 8F D0
9190- F8 20 97 91 18 90 ED A5
9198- FF C9 02 90 0E C6 FF C6
91A0- FE A5 FF C9 01 90 04 C6
91A8- FF C6 FE 60 E6 FF E6 FE
91B0- E6 FF E6 FE 60 A5 FD 85
91B8- 06 20 91 93 18 A4 FE A9
91C0- 00 85 08 85 09 85 07 90
91C8- 02 E6 08 B1 26 C9 80 90
91D0- 02 E6 09 A5 08 F0 07 B1
91D8- 26 09 80 4C E2 91 B1 26
91E0- 29 7F 6A 91 26 90 02 E6
91E8- 07 A5 09 C9 01 90 06 B1
91F0- 26 09 80 91 26 C4 FF F0
91F8- 08 88 A5 07 C9 01 4C BF
9200- 91 C6 06 A5 06 C9 FF F0
9208- 04 C5 FC B0 AC 60 A5 FD
9210- 85 06 20 91 93 18 A4 FF
9218- A9 00 85 08 85 09 B1 26
9220- 2A 91 26 B0 02 90 02 E6
9228- 08 C9 80 B0 02 90 02 E6
9230- 09 A5 08 D0 09 B1 26 29
9238- 7F 91 26 4C 44 92 B1 26
9240- 09 80 91 26 C4 FE F0 09
9248- C8 18 A5 09 C9 01 4C 18
9250- 92 C6 06 A5 06 C9 FF F0
9258- 04 C5 FC B0 B5 60 38 A5
9260- FC E5 E3 85 FC 38 A5 FD
9268- E5 E3 85 FD 60 18 A5 FC
9270- 65 E3 85 FC 18 A5 FD 65
9278- E3 85 FD 60 A9 00 8D 54
9280- C0 A9 40 85 E6 A5 FC C5
9288- E3 90 0F 20 6D 92 20 2F
9290- 93 20 5E 92 20 5E 92 20
9298- 2F 93 60 A9 00 8D 55 C0
92A0- A9 20 85 E6 20 6D 92 20
92A8- 2F 93 20 5E 92 20 5E 92
92B0- 20 2F 93 60 A9 00 8D 54
92B8- C0 A9 40 85 E6 20 5E 92
92C0- 20 2F 93 20 6D 92 20 6D
92C8- 92 20 2F 93 60 A9 00 8D
92D0- 55 C0 A9 20 85 E6 20 5E
92D8- 92 20 2F 93 20 6D 92 20
92E0- 6D 92 20 2F 93 60 A9 00
92E8- 85 FA A5 FD 85 06 20 91
92F0- 93 A4 FF A2 00 A1 FA C9
92F8- 7F F0 15 C9 01 90 11 86
9300- F9 4A 26 F9 E8 E0 07 90
9308- F8 4A A5 F9 90 02 09 80
9310- 91 26 C8 E6 FA D0 02 E6
9318- FB C4 FE 90 D6 F0 D4 C6
9320- 06 A5 06 C9 FF F0 04 C5
9328- FC B0 C3 20 61 93 60 A9
9330- 00 85 FA A5 FD 85 06 20
9338- 91 93 A4 FE A2 00 A1 FA
9340- 51 26 91 26 88 18 E6 FA
9348- D0 02 E6 FB C0 FF F0 04
9350- C4 FF B0 EA C6 06 A5 06
9358- C9 FF F0 04 C5 FC B0 D7
9360- 60 A9 00 85 FA A5 FD 85
9368- 06 20 91 93 A4 FE A2 00
9370- B1 26 81 FA 88 18 E6 FA
9378- D0 02 E6 FB C0 FF F0 04
9380- C4 FF B0 EC C6 06 A5 06
9388- C9 FF F0 04 C5 FC B0 D9
9390- 60 A4 06 B1 CE 85 26 A5
9398- E6 C9 40 D0 05 B1 DE 85
93A0- 27 60 B1 EE 85 27 60 A9
93A8- 80 85 CE A9 94 85 CF A9
93B0- 40 85 EE A9 95 85 EF A9
93B8- C0 85 DE A9 93 85 DF 60
93C0- 40 44 48 4C 50 54 58 5C
93C8- 40 44 48 4C 50 54 58 5C
93D0- 41 45 49 4D 51 55 59 5D
93D8- 41 45 49 4D 51 55 59 5D
93E0- 42 46 4A 4E 52 56 5A 5E
93E8- 42 46 4A 4E 52 56 5A 5E
93F0- 43 47 4B 4F 53 57 5B 5F
93F8- 43 47 4B 4F 53 57 5B 5F
9400- 40 44 48 4C 50 54 58 5C
9408- 40 44 48 4C 50 54 58 5C
9410- 41 45 49 4D 51 55 59 5D
9418- 41 45 49 4D 51 55 59 5D
9420- 42 46 4A 4E 52 56 5A 5E
9428- 42 46 4A 4E 52 56 5A 5E
9430- 43 47 4B 4F 53 57 5B 5F
9438- 43 47 4B 4F 53 57 5B 5F
9440- 40 44 48 4C 50 54 58 5C
9448- 40 44 48 4C 50 54 58 5C
9450- 41 45 49 4D 51 55 59 5D
9458- 41 45 49 4D 51 55 59 5D
9460- 42 46 4A 4E 52 56 5A 5E
9468- 42 46 4A 4E 52 56 5A 5E
9470- 43 47 4B 4F 53 57 5B 5F
9478- 43 47 4B 4F 53 57 5B 5F
9480- 00 00 00 00 00 00 00 00
9488- 80 80 80 80 80 80 80 80
9490- 00 00 00 00 00 00 00 00
9498- 80 80 80 80 80 80 80 80
94A0- 00 00 00 00 00 00 00 00
94A8- 80 80 80 80 80 80 80 80
94B0- 00 00 00 00 00 00 00 00
94B8- 80 80 80 80 80 80 80 80
94C0- 28 28 28 28 28 28 28 28
94C8- A8 A8 A8 A8 A8 A8 A8 A8
94D0- 28 28 28 28 28 28 28 28
94D8- A8 A8 A8 A8 A8 A8 A8 A8
94E0- 28 28 28 28 28 28 28 28
94E8- A8 A8 A8 A8 A8 A8 A8 A8
94F0- 28 28 28 28 28 28 28 28
94F8- A8 A8 A8 A8 A8 A8 A8 A8
9500- 50 50 50 50 50 50 50 50
9508- D0 D0 D0 D0 D0 D0 D0 D0
9510- 50 50 50 50 50 50 50 50
9518- D0 D0 D0 D0 D0 D0 D0 D0
9520- 50 50 50 50 50 50 50 50
9528- D0 D0 D0 D0 D0 D0 D0 D0
9530- 50 50 50 50 50 50 50 50
9538- D0 D0 D0 D0 D0 D0 D0 D0
9540- 20 24 28 2C 30 34 38 3C
9548- 20 24 28 2C 30 34 38 3C
9550- 21 25 29 2D 31 35 39 3D
9558- 21 25 29 2D 31 35 39 3D
9560- 22 26 2A 2E 32 36 3A 3E
9568- 22 26 2A 2E 32 36 3A 3E
9570- 23 27 2B 2F 33 37 3B 3F
9578- 23 27 2B 2F 33 37 3B 3F
9580- 20 24 28 2C 30 34 38 3C
9588- 20 24 28 2C 30 34 38 3C
9590- 21 25 29 2D 31 35 39 3D
9598- 21 25 29 2D 31 35 39 3D
95A0- 22 26 2A 2E 32 36 3A 3E
95A8- 22 26 2A 2E 32 36 3A 3E
95B0- 23 27 2B 2F 33 37 3B 3F
95B8- 23 27 2B 2F 33 37 3B 3F
95C0- 20 24 28 2C 30 34 38 3C
95C8- 20 24 28 2C 30 34 38 3C
95D0- 21 25 29 2D 31 35 39 3D
95D8- 21 25 29 2D 31 35 39 3D
95E0- 22 26 2A 2E 32 36 3A 3E
95E8- 22 26 2A 2E 32 36 3A 3E
95F0- 23 27 2B 2F 33 37 3B 3F
95F8- 23 27 2B 2F 33 37 3B 20
9600- 3A 3E 22 26 2A 2E 32
```