Chicken Little

Save the Purple Feathered Wing Flappers From Extinction

Most arcade-type games tend to be violent. The object of many classics, such as Space Invaders and many other games, is to kill as many creatures as possible. Mindless destruction is the norm for these games. Peaceful negotiation is not even an option. The poor creatures which we ruthlessly attack are never given an opportunity to contribute positively to society. Perhaps we could learn something from a space invader if we gave him the chance.

USING THE PROGRAM

The object of Chicken Little is to save rather than destroy. You are in charge of a bird sanctuary and are having trouble with one of the species of birds. The Purple Feathered Wing Flappers lay their eggs while in flight. Normally, the eggs land safely and hatch on the ground, but exposure to DDT has weakened the shells; the eggs can no longer hit the ground without breaking.

When you RUN CHICKEN.LITTLE, your job is to keep the eggs in the air by bouncing them off of a spring controlled by the Left- and Right-Arrows or the zero paddle until they hatch. When using the keyboard, you can press the Space bar to stop the spring.

You receive ten points for each bounce of the spring and 25 points when the egg finally hatches. The game becomes quite intense as the screen fills with eggs for you to bounce.

If you miss six eggs you are retired from your job. At the conclusion of the game, you can either start a new game by pressing the space bar, or end the game by pressing the ESC key

ENTERING THE PROGRAM

Note: In order to print the CheckIt checksums, several of the hex listings begin at memory address $800, as shown in the listings. Enter and save each of the hex listings as shown in the instructions below. The hex code will be loaded into the proper locations in memory when the program is run.

Chicken Little was written in several parts. Begin by typing in the Applesoft program in **Listing 1**, and saving it with the command

```
SAVE CHICKEN.LITTLE
```

Next, enter the Applesoft program shown in **Listing 2**. Save it to disk with

```
SAVE CHICKEN.EXEC
```

Jenny Schmidt, 1407 E. Harding Dr., Appleton, WI 54991. This program works under DOS 3.3, only, and will not run on the Apple IIGS.

The CHICKEN.EXEC program creates a text file called CHICKEN.LOADER which is EXEC'd when you run the CHICKEN.LITTLE program. Once you've typed in and checked CHICKEN.EXEC, you can RUN it. After it creates the CHICKEN.LOADER file, you won't need to use the CHICKEN.EXEC program again.

Type the shape table, as shown in **Listing 3**, into the Monitor and save it with

```
BSAVE SHAPES.DAT, A$A50, L$91
```

If you have an assembler, you can enter the assembly-language source code in **Listings 7-9**, and compile them to create CHICKEN.OBJ, SHAPER.OBJ, and SOUNDS.OBJ. The ASM/65 assembler was used to create the assembly-language portions of Chicken Little.

If you do not have an assembler (or opt not to use one), you should type in the hex code in **Listings 4-6**, as follows.

Type the hex code in **Listing 4** into the Monitor, and save it to disk with the command

```
BSAVE CHICKEN.OBJ, A$800, L$41A
```

Type in **Listing 5**, and save it with

```
BSAVE SHAPER.OBJ, A$800, L$1D9
```

Finally, type in **Listing 6** and save it by typing

```
BSAVE SOUNDS.OBJ,A$800,L$5C
```

Chicken Little is now complete and ready to play.

HOW THE PROGRAM WORKS

Chicken Little does not use the Applesoft shape routines. The vector method Applesoft uses is good for line drawings, but they are hard to use for filled designs. Also, Applesoft shapes do not work well in color; entire vertical lines can disappear depending on where the shape is plotted. For these reasons, I have developed my own method for making shapes. You might find this method useful in the development of your own games. The structure and operation of my Medium-Resolution Shape Generator is discussed later in this article.

Chicken Little is started by running the Applesoft CHICKEN.LITTLE program (**Listing 1**). This program, however, simply describes the game, asks whether you'll be using the keyboard or the zero paddle, and then EXEC's the CHICKEN.LOADER file which loads the game into memory.

The source code for the main program is shown in **Listing 7**. Important variables are defined starting in **line 17**. EGGS is an array of eight consecutive memory locations. Each location contains the state of each egg. A

value of 0 means the egg has not been laid yet; a value of 1 means that the egg has been laid, but has not hatched; and a 3 means the egg has hatched. EGGX and EGGY are eight element arrays that contain the X and Y coordinates of the eggs. BIRDX and BIRDY are eight element arrays for the X and Y coordinates of the baby chicks.

YPART is the index to the Y coordinate table for the eggs. It, too, is an eight element array. EGGINC, another eight element array, contains the increment for the X coordinate of the egg. An element in this array contains a value of 0 if the egg is to bounce straight up and down, 2 if it is to bounce to the right, and 254 if it is to bounce to the left. Notice that 254 is the two's complement of two. By adding 254 to the X coordinate, you are in effect subtracting three from it. COMPRS contains the state of the spring: a six if the spring is open and a seven if it is compressed.

The actual program starts at **line 58**. First, it zeroes some locations and sets up the locations needed for the monitor move routine. It then does a JSR MOVE which places zero's in the hi-res graphics page and erases anything that might have been there. The program then toggles the soft switches to turn on the graphics mode.

Next, it stores the address of the address table in SHAPAD ($FB, $FC) and the address of the plot table in SHAPET ($FD, $FE). It then initializes a few more locations and starts the action in **line 121**.

First the spring is forced to the open position. In **lines 124-130** the score is shown. Next, in **lines 132-146**, a mother Wing Flapper is plotted if RNDOM1 contains a 15. DRWDAT is the start of the plot table. The plot information is stored here. **Line 146** calls the plot routine which causes the Wing Flapper to be plotted on the screen. **Lines 148-173** hatch a chick if RNDOM2 is zero. These lines also add 25 points to the score. **Lines 175-215** decide when to drop an egg. If RNDOM1 is zero, the Wing Flapper is erased, a space in EGGS is found, and initial coordinates are defined for that egg. Additionally, new numbers are generated for RNDOM1, RNDOM2, and RNDOM3. **Lines 217-259** set up the erase table data.

Next the program checks boundaries starting at **line 262**. If the egg is at the bottom of the screen, its X coordinate is subtracted from SPRING, the X coordinate of the screen. If the result of the subtraction is between -3 and 6, the egg has landed on the spring and the program jumps to the hit sections. Otherwise it draws a raw egg, sounds the buzzer, and checks the raw egg counter. If COUNT reaches 6, the game stops and waits for the user to hit the Space bar or the Escape key. The hit section starts at **line 325**.

Line 371 begins a section which calculates positions for all the objects. First the program checks INPUT to see if the player wants to use the keyboard or paddles. **Line 405** starts the movement of the eggs and baby chicks.

Line 451-511 sets up the plot table, erases the previous shapes, and draws the new ones. In **line 512-515** the spring sound is made if the spring is compressed. In **line 516** the random number counter is incremented and in **line 517** the entire process is started over with a jump to the top of the program.

Making Decisions

In decision-making, the program compares the coordinates of a shape (an egg in Chicken Little, for example) to some element in the shape's environment, (the edge of the screen, perhaps). If the comparison indicates a critical situation (such as the egg reaching the edge of the screen), the program should take appropriate action. In this case, move the egg to the other side of the screen. In addition to keeping objects from falling off the screen, the program must take care of interactions between objects. The egg bouncing off the spring is an example of interaction between objects.

Deciding when an event that is independent of the environment should take place is another important type of decision making. This type of decision-making must rely on randomly generated numbers. An egg hatching is an example of a random event in Chicken Little. Chicken Little generates random numbers by first setting up a counter. This counter is continuosly incremented in the main loop of the program. When a random number is needed, the program makes a copy of the counter. The copy is then decremented on each pass through the program. When the copy reaches zero, the event takes place and another copy is made of the counter. This process does not insure perfectly distributed numbers and would not stand up to statistical analysis. It does, however, work well for the events in Chicken Little.

MEDIUM-RESOLUTION SHAPE GENERATOR

Editor's note: The remainder of this article examines the Shape Generator, an assembly-language program that can be used for your own programming. You should have some knowlege of assembly language programming, but it is not necessary to understand the programing in order to run Chicken Little.

The Medium-Resolution Shape Generator presented here has many applications, especially for games. With a little imagination and effort you should be able to write some fantastic applications.

The shape generator itself (shown in **Listing 8**) is easy to understand. The program starts by loading the X and Y coordinates, and the colors of the first entry of the plot table into XTEMP, YCOORD, COLOR1, COLOR2. The shape number is multiplied by two and used as an index to find the address of the particular shape. This address is stored in the program by **lines 51-57**.

The program then moves on to the plot section. A byte of shape data is stored in TEMPSH. A copy of XTEMP1 is stored in XCOORD and the program moves on to the BITS soubroutine. BITS peels off a bit from TEMPSH and does a JSR DRAW if the bit is a one. If the bit is a zero, XCOORD is incremented and the next bit is fetched.

DRAW first does a JSR FNDCRD. FNDCRD translates the X and Y coordinates into an address on the Hi-Res page. Next DRAW shifts the color byte until it is in the right position within the Hi-Res byte. The color is then ORAed with the Hi-Res address and stored in the Hi-Res address.

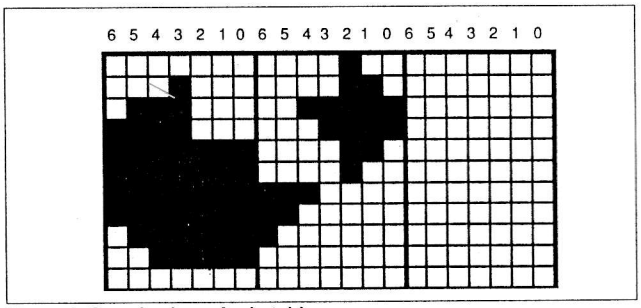

Figure 1: Sample shape for hatching egg.

DRAW returns and BITS fetches the next bit of data. When BITS returns, YCOORD is incremented and BITS is called again with the same shape data byte. The process continues until the end of line byte is reached. Then YCOORD is incremented and XCOORD is set to equal to XTEMP1 again.

Once the entire shape has been plotted, PLOT returns and another line of plot table information is processed. Refer to the commented source code for the complete details.

Shape Formats

The smallest unit in a mid-res shape generator shape is composed of a two-by-two square. Since this is the smallest unit, the resolution is reduced to 140 by 86. Although the resolution is greatly reduced, handling color becomes much easier. The smallest unit is two points across. White is achieved when both points are on. Violet or blue is produced when the first point is on and the second is off. Green or orange is displayed when the first point is off and the second is on. The smallest unit is also two down to make its shape square rather than rectangular. Also, shades of color can be made by letting the first point down be one color and the second point down be another.

The format of the shape data allows each byte to hold seven dots. The high bit signals the end of the line. Let's draw a simple shape and develop shape data for it (**Figure 1**).

We start by dividing each row in the shape into groups of seven. Each group of seven in a row will fit into one byte of shape data. We need to translate the dots and blanks into hexadecimal numbers. Let the blanks be 0s and the dots be 1s. The first group of seven in the first row would be %0000000 in binary. The second group of seven would be %0000100. Converted to hexadecimal the first two groups would yield $00, $04. The end of a row is signaled by an $80. The first row, therefore, is $00, $04, $80. Here is the shape data:

```
00 04 80
08 06 80
38 1F 80
78 0F 80
7F 06 80
7F 04 80
7F 70 80
7F 60 80
3F 40 80
1F 00 C0
```

The $C0 stands for end of shape. Remember, only seven dots and blanks can fit in each byte. If there is more than seven dots wide, one byte for each group of seven plus the end-of-line byte would have to be used for each row.

The shape data may be anywhere in memory except, of course, on top of the shape generator ($80E-9E7). The various tables and value entries are summarized in **Table 1** and you can explore them in conjunction with the source program listings. After running and quitting Chicken Little, you can also browse through the shape and address tables by typing CALL -151 and exploring the various table addresses.

The shape generator may be used in conjunction with an Applesoft program. First, the Applesoft program must be placed out of the way of the shape generator, the data, and the Hi-Res graphics page. You must reset the program pointer. POKE 103,00 : POKE 104,64 sets the program pointer above Hi-Res page 1. The program pointer should be set before a program is typed in or before it is loaded. The Applesoft program can access the shape generator by first poking the page and table locations, then poking in the plot table data (X, Y, color1, color2, and the shape number), and finally calling the shape generator. The rest of the shape data should be loaded from disk or poked in from Applesoft first. Poking in Applesoft, however, is a slow process. If your application depends on speed, you are better off using assembly language.

Assembly is great for programming games, because it is very speedy and it's not as complex as you may think. Any arcade-type game can be broken down into four different problems: positioning, plotting, erasing, and decision making.

Plotting and erasing, are taken care of by the shape generator. Positioning is relatively easy. For example, we could make a shape move across the screen by incrementing the X coordinate on each pass through a loop. Such a program would have to set up the plot and erase tables, call the plotting routine, increment the X coordinate, call the erase routine, and loop back to the call to the plotting routine.

Table 1: Middle-Resolution Shape Generator Formats

(All numbers are in hexadecimal)

Shape Data Formats:

Shape0: data, data,... 80 (end of line), data,... C0 (end of shape)
. . .
Shape*N*: data, data,... 80 (end of line), data,... C0 (end of shape)

Address Table ($FB, $FC):

Shape0: low, high Shape1: low, high ... Shape*N*: low, high
(Store the address of the start of this table in low-byte, high-byte order in $FB, $FC.)

Plot Table ($FD, $FE):

X0 Y0 Color0 Color0 Shape0 ... XN YN ColorN ShapeN
FF FF FF FF
(ShapeN is the shape number of the particular shape you want to use; It corresponds to its position in the address table. An FF in place of the shape number signals the end of the plot table. Store the address of the start of this table in low-byte, high-byte order in $FD,$FE.)

Colors:

Black	=>	00	(transparent)
Violet	=>	01	
Green	=>	02	
White1	=>	03	
Blue	=>	81	
Orange	=>	82	
White2	=>	83	

(Store $00 in $804 for page 1 or $20 for page 2.)

Erase table ($06, $07):

X0 Y0 Width0 Length0... XN YN WidthN LengthN FF
(An FF in place of the X coordinate signals the end of the erase table. The address of the start of this table in low-byte, high-byte order is in $06, $07.)

Draw => $80E (2062 in decimal)
Erase => $9A3 (2467 in decimal)

THE FEBRUARY 1992 DISK CONTAINS THIS PROGRAM

If you'd rather not type in the listing for this program, you can buy it on disk, complete, free of typos, and ready to run. Nibble's February 1992 programs are available on a single disk for an introductory price of $12.95 from Nibble, Box 256 Lincoln, MA 01773. Add $2.50 for shipping/handling within the U.S. and Canada; $7.50 for overseas air mail. Introductory price expires 4/30/92; after that date, the price will be $16.95. See the Nibble Software Directory in this issue for ordering information. **ORDER NO.: Z13**

Listing 1: CHICKEN.LITTLE

```
37 | 10  REM  ************************
C0 | 20  REM  *     CHICKEN LITTLE     *
B9 | 30  REM  *   by Jenny Schmidt    *
AE | 40  REM  *  Copyright (C) 1991   *
CB | 50  REM  * MindCraft Publ. Corp.*
24 | 60  REM  *   Lincoln, MA 01773   *
45 | 70  REM  ************************
B6 | 80  TEXT : HOME : POKE 768,0
C9 | 90  DATA  C,H,I,C,K,E,N," ",L,I,T,T,L,E
BB | 100  FOR I = 1 TO 14: READ A$
E8 | 110  VTAB 22: HTAB 14 + I - 1: PRINT " " +  CHR
       $ (95)
48 | 120  FOR J = 2 TO 21
F4 | 130  HTAB 14 + I: VTAB J: PRINT A$: HTAB 14 + I
       : VTAB J - 1: PRINT " "
6B | 140  NEXT J
1D | 150  FOR J = 21 TO 11 STEP  - 1
91 | 160  HTAB 14 + I: VTAB J: PRINT A$: HTAB 14 + I
       : VTAB J + 1: PRINT " "
0C | 170  NEXT J: NEXT I: VTAB 13: HTAB 20: PRINT "B
       Y"
97 | 180 A$ = " SCHMIDT":B$ = "JENNY "
F6 | 190  FOR I = 2 TO 20: FOR W = 1 TO 10: NEXT W
EC | 200  VTAB 15: HTAB I: PRINT A$: VTAB 15: HTAB 4
       0 - I - 5: PRINT B$
60 | 210  FOR W = 1 TO 30: NEXT W: NEXT I: TEXT
E7 | 220  INVERSE : INPUT "                PRESS <RETURN> T
       O PLAY          ";A$: NORMAL : HOME
C3 | 230  PRINT "Welcome to the 'Fly by Night' Bird"
       : PRINT "Sanctuary. As the new caretaker, y
       ou": PRINT "will be in charge of the Purple
       ": PRINT "Feathered Wing Flappers. This spe
       cies"
9A | 240  PRINT "of bird lays its eggs in flight.":
       PRINT "Normally the eggs land safely, but
       DDT": PRINT "has weakened the egg shells. Y
       our job": PRINT "is to bounce the eggs off
       of a spring": PRINT "until the eggs hatch.
       You are rewarded"
6F | 250  PRINT "ten points for each bounce and 25 p
       oints";: PRINT "for each chick hatched. The
        spring is": PRINT "controlled by the Zero
       Paddle or by the": PRINT "Left/Right Arrows
        and space bar keys.": PRINT "I you miss si
       x eggs, you will be retired";
0A | 260  PRINT "from your job. Hitting the Space ba
       r": PRINT "will restart the game after it h
       as ended";: PRINT "and Escape will send you
        back to": PRINT "BASIC."
E9 | 270  PRINT : PRINT : INVERSE : PRINT "PRESS <P>
       FOR PADDLE OR <K> FOR KEYBOARD";: NORMAL :
       GET A$: PRINT A$
0F | 280  IF A$ = "P" THEN  POKE 768,128: GOTO 290
F2 | 290  PRINT  CHR$ (4);"EXEC CHICKEN.LOADER"
TOTAL: 5E97
END OF LISTING 1
```

Listing 2: CHICKEN.EXEC

```
37 | 10  REM  ************************
C0 | 20  REM  * CHICKEN.EXEC          *
B9 | 30  REM  * COPYRIGHT (C) 1991    *
AE | 40  REM  * MINDCRAFT PUBL. CORP.*
CB | 50  REM  ************************
3D | 60 D$ =  CHR$ (4)
6B | 70  PRINT D$;"OPEN CHICKEN.LOADER"
0D | 80  PRINT D$;"WRITE CHICKEN.LOADER"
74 | 90  PRINT "FP"
5A | 100  PRINT "BLOAD SHAPER.OBJ,A$80E"
3B | 110  PRINT "BLOAD SHAPES.DAT,A$A50"
E8 | 120  PRINT "BLOAD SOUNDS.OBJ,A$9F0"
4B | 130  PRINT "BLOAD CHICKEN.OBJ,A$D41"
2E | 140  PRINT "HOME"
C8 | 150  PRINT "CALL 3393"
8C | 160  PRINT "FP"
50 | 170  PRINT "HOME"
93 | 180  PRINT D$;"CLOSE CHICKEN.LOADER"
TOTAL: 64A9
END OF LISTING 2
```

Listing 3: SHAPES.DAT

```
Start: A50          Length: 91
5C | 0A50:00 0C 80 00 1E 80 7F 7C
96 | 0A58:80 77 78 80 38 78 80 1F
58 | 0A60:78 80 0F 70 C0 0C 80 1E
7F | 0A68:80 3F 80 7F 40 80 7F 40
32 | 0A70:80 7F 40 80 3F 80 1E C0
05 | 0A78:07 80 37 80 7E 80 1C 80
A1 | 0A80:7F 80 76 80 64 C0 00 04
FA | 0A88:80 08 06 80 38 1F 80 78
90 | 0A90:0F 80 7F 06 80 7F 04 80
8C | 0A98:7F 70 80 7F 60 80 3F 40
F4 | 0AA0:80 1F 00 C0 03 40 80 07
1E | 0AA8:60 80 0F 70 C0 80 38 1C
46 | 0AB0:80 70 0E 80 3F 7C C0 7F
66 | 0AB8:7F 80 40 00 80 7F 7F 80
29 | 0AC0:00 01 80 7F 7F C0 80 80
E7 | 0AC8:7F 7F 80 7F 7F 80 7F 7F
93 | 0AD0:C0 50 0A 65 0A 78 0A 86
F9 | 0AD8:0A A4 0A AD 0A B7 0A C6
E6 | 0AE0:0A
TOTAL: BAB1
END OF LISTING 3
```

Listing 4: CHICKEN.OBJ

```
Start: 800          Length: 41A
3D | 0800:A9 00 A8 85 06 8D 3D 0D
EA | 0808:8D 3E 0D 85 3C 8D 04 08
07 | 0810:8D 00 20 A9 20 85 3D 85
5A | 0818:43 A9 01 85 42 A9 3F 85
3F | 0820:3F A9 FF 85 3E 20 2C FE
B8 | 0828:8D 50 C0 8D 53 C0 8D 57
23 | 0830:C0 A9 0A 85 FC A9 D1 85
F3 | 0838:FB A9 0B 85 FE A9 00 85
14 | 0840:FD A9 0C 85 07 A9 00 A0
5F | 0848:07 99 00 0D 88 10 FA A9
54 | 0850:32 8D 3C 0D A9 0F 8D 3A
25 | 0858:0D A9 2F 8D 39 0D A9 06
4A | 0860:8D 40 0D A0 BA C5 D2 CF
9A | 0868:C3 D3 A9 0E 85 24 A9 14
55 | 0870:20 5B FB A2 06 BD A4 0D
0B | 0878:20 ED FD CA 10 F7 A9 06
B4 | 0880:8D 38 0D A9 15 85 24 A9
22 | 0888:14 20 5B FB AD 3D 0D AE
72 | 0890:3E 0D 20 41 F9 AD 3A 0D
FF | 0898:C9 0F D0 20 AD 3C 0D 8D
DF | 08A0:00 0B A9 00 8D 01 0B 8D
51 | 08A8:04 0B A9 01 8D 02 0B A9
AB | 08B0:01 8D 03 0B A9 FF 8D 09
99 | 08B8:0B 20 0E 08 CE 3B 0D D0
A1 | 08C0:38 A0 07 B9 00 0D C9 01
BE | 08C8:D0 2C 18 F8 A9 25 6D 3E
43 | 08D0:0D 8D 3E 0D 90 08 A9 00
8A | 08D8:6D 3D 0D 8D 3D 0D D8 A9
F4 | 08E0:03 99 00 0D B9 08 0D 99
1B | 08E8:18 0D B9 10 0D 99 20 0D
E2 | 08F0:20 33 0A 4C 3A 0E 88 10
C5 | 08F8:CA CE 3A 0D D0 5A AD 3C
90 | 0900:0D 8D 00 0C A9 00 8D 01
E6 | 0908:0C A9 02 8D 02 0C A9 07
5E | 0910:8D 03 0C A9 FF 8D 04 0C
68 | 0918:20 A3 09 A0 07 B9 00 0D
7C | 0920:D0 19 A9 01 99 00 0D AD
FE | 0928:3C 0D 99 08 0D A9 00 99
09 | 0930:30 0D 99 28 0D 99 10 0D
15 | 0938:4C 7F 0E 88 10 DF AD 3F
49 | 0940:0D C9 0F B0 02 A9 64 8D
D9 | 0948:3A 0D 2A 8D 3B 0D 4A C9
61 | 0950:7E 90 02 A9 7D 8D 3C 0D
BF | 0958:AD 39 0D 8D 00 0C A9 3A
D5 | 0960:8D 01 0C A9 02 8D 02 0C
92 | 0968:A9 05 8D 03 0C A2 04 A0
A6 | 0970:07 B9 00 0D F0 3B B9 08
AC | 0978:0D 9D 00 0C E8 B9 10 0D
6B | 0980:9D 00 0C E8 A9 02 9D 00
B4 | 0988:0C E8 A9 0B 9D 00 0C E8
70 | 0990:B9 00 0D C9 03 D0 1A B9
BA | 0998:18 0D 9D 00 0C E8 B9 20
EC | 09A0:0D 9D 00 0C E8 A9 01 9D
BD | 09A8:00 0C E8 A9 07 9D 00 0C
F9 | 09B0:E8 88 10 BD A9 FF 9D 00
92 | 09B8:0C A0 07 B9 00 0D D0 03
5E | 09C0:4C ED 0F B9 10 0D C9 34
96 | 09C8:F0 03 4C ED 0F AD 39 0D
9C | 09D0:38 F9 08 0D C9 06 90 7A
3C | 09D8:C9 F3 90 03 4C BB 0F B9
4F | 09E0:00 0D C9 03 D0 03 4C ED
CB | 09E8:0F A9 00 99 00 0D B9 08
D9 | 09F0:0D 8D 00 0B 8D 05 0B A9
60 | 09F8:49 8D 01 0B 8D 06 0B A9
AC | 0A00:83 8D 02 0B 8D 03 0B A9
06 | 0A08:05 8D 04 0B A9 82 8D 07
EC | 0A10:0B A9 82 8D 08 0B A9 04
85 | 0A18:8D 09 0B A9 FF 8D 0E 0B
63 | 0A20:98 48 20 0E 08 68 A8 20
00 | 0A28:06 0A CE 40 0D D0 7D 20
0E | 0A30:06 0A 20 06 0A 20 06 0A
8B | 0A38:AD 00 C0 10 FB C9 9B D0
30 | 0A40:07 AD 10 C0 AD 51 C0 60
55 | 0A48:C9 A0 D0 EC AD 10 C0 4C
57 | 0A50:41 0D 48 18 F8 A9 10 6D
0C | 0A58:3E 0D 8D 3E 0D 90 08 A9
52 | 0A60:00 6D 3D 0D 8D 3D 0D D8
9E | 0A68:68 C9 06 B0 35 A9 FE 99
83 | 0A70:30 0D A9 07 8D 38 0D 4C
98 | 0A78:ED 0F 48 18 F8 A9 10 6D
AA | 0A80:3E 0D 8D 3E 0D 90 08 A9
58 | 0A88:00 6D 3D 0D 8D 3D 0D D8
32 | 0A90:68 C9 FA B0 0D A9 02 99
89 | 0A98:30 0D A9 07 8D 38 0D 4C
33 | 0AA0:ED 0F A9 00 99 30 0D A9
26 | 0AA8:07 8D 38 0D 88 30 03 4C
58 | 0AB0:FC 0E AD 00 03 30 3D AD
2E | 0AB8:00 C0 C9 95 D0 19 EE 39
92 | 0AC0:0D EE 39 0D EE 39 0D EE
3A | 0AC8:39 0D AD 39 0D 10 35 A9
A9 | 0AD0:00 8D 39 0D 4C 45 10 C9
E1 | 0AD8:88 D0 29 CE 39 0D CE 39
D3 | 0AE0:0D CE 39 0D CE 39 0D AD
F0 | 0AE8:39 0D 10 18 A9 7D 8D 39
C1 | 0AF0:0D 4C 45 10 A2 00 20 1E
61 | 0AF8:FB 98 4A C9 7D 90 02 A9
F6 | 0B00:7D 8D 39 0D A2 07 BD 00
5F | 0B08:0D F0 68 BC 26 0D FE 28
CA | 0B10:0D B9 48 11 D0 07 9D 28
43 | 0B18:0D A8 B9 48 11 9D 10 0D
DC | 0B20:BD 08 0D 18 7D 30 0D C9
55 | 0B28:FA 90 05 A9 82 4C 77 10
BC | 0B30:C9 83 90 02 A9 00 9D 08
4A | 0B38:0D BD 00 0D C9 03 D0 33
5A | 0B40:BD 10 0D C9 34 D0 0F A9
E3 | 0B48:42 9D 10 0D A9 09 9D 28
4D | 0B50:0D A9 00 9D 30 0D DE 18
FF | 0B58:0D DE 18 0D 10 08 A9 00
93 | 0B60:9D 00 0D 4C B4 10 DE 20
C0 | 0B68:0D DE 20 0D 10 05 A9 00
0B | 0B70:9D 00 0D CA 30 03 4C 47
EB | 0B78:10 AD 39 0D 8D 00 0B A9
C3 | 0B80:3A 8D 01 0B A9 02 8D 02
4B | 0B88:0B A9 02 8D 03 0B AD 38
93 | 0B90:0D 8D 04 0B A2 05 A0 07
96 | 0B98:B9 00 0D F0 48 B9 08 0D
9A | 0BA0:9D 00 0B E8 B9 10 0D 9D
12 | 0BA8:00 0B E8 A9 03 9D 00 0B
EE | 0BB0:E8 A9 03 9D 00 0B E8 B9
F3 | 0BB8:00 0D 9D 00 0B E8 B9 00
91 | 0BC0:0D C9 03 D0 20 B9 18 0D
CF | 0BC8:9D 00 0B E8 B9 20 0D 9D
02 | 0BD0:00 0B E8 A9 81 9D 00 0B
48 | 0BD8:E8 A9 81 9D 00 0B E8 A9
7E | 0BE0:02 9D 00 0B E8 88 10 B0
9A | 0BE8:E8 E8 E8 E8 A9 FF 9D 00
58 | 0BF0:0B 20 A3 09 20 0E 08 AD
29 | 0BF8:38 0D C9 07 D0 03 20 F0
30 | 0C00:09 EE 3F 0D 4C BF 0D 0A
FE | 0C08:0C 0F 12 16 1C 23 2A 32
D0 | 0C10:34 32 2A 23 1C 16 12 0F
AF | 0C18:0A 00
TOTAL: 7AC6
END OF LISTING 4
```

Listing 5: SHAPER.OBJ

```
Start: 800          Length: 1D9
BA | 0800:A0 00 B1 FD 8D 08 08 8D
42 | 0808:09 08 C8 B1 FD 0A 8D 00
64 | 0810:08 C8 B1 FD 8D 0C 08 C8
D9 | 0818:B1 FD 8D 0D 08 C8 98 48
0F | 0820:B1 FD 30 1C 0A A8 B1 FB
77 | 0828:8D 53 08 8D 87 08 C8 B1
1A | 0830:FB 8D 54 08 8D 88 08 20
BA | 0838:50 08 68 A8 C8 4C 10 08
AD | 0840:68 60 A2 00 BD 11 11 10
EA | 0848:14 0A 30 10 E8 EE 00 08
11 | 0850:EE 00 08 AD 08 08 8D 09
60 | 0858:08 4C 52 08 60 8D 02 08
4A | 0860:AD 09 08 8D 01 08 AD 0C
D1 | 0868:08 8D 03 08 20 A2 08 EE
E8 | 0870:00 08 AD 09 08 8D 01 08
F0 | 0878:BD 11 11 8D 02 08 AD 0D
D8 | 0880:08 8D 03 08 20 A2 08 AD
37 | 0888:01 08 8D 09 08 E8 CE 00
59 | 0890:08 4C 52 08 0E 02 08 A0
CB | 0898:07 0E 02 08 90 07 98 48
F5 | 08A0:20 BA 08 68 A8 EE 01 08
2C | 08A8:88 D0 EE 60 20 06 09 AD
8E | 08B0:03 08 AC 07 08 F0 08 C0
CD | 08B8:06 F0 12 0A 88 D0 FC 11
D2 | 08C0:F9 91 F9 AD 03 08 29 80
57 | 08C8:11 F9 91 F9 60 AD 03 08
```

```
B8|08D0:29 01 0A 0A 0A 0A 0A 0A
B5|08D8:A0 00 11 F9 91 F9 AD 03
0E|08E0:08 29 80 11 F9 91 F9 E6
0C|08E8:F9 D0 02 E6 FA AD 03 08
AA|08F0:29 02 4A 11 F9 91 F9 60
24|08F8:AD 00 08 4A 4A 4A 0A A8
13|0900:B9 6B 09 85 F9 C8 B9 6B
28|0908:09 85 FA AD 00 08 29 07
95|0910:A8 B9 9B 09 18 65 FA 18
35|0918:6D 04 08 85 FA A9 00 8D
E5|0920:06 08 A9 70 8D 05 08 AD
88|0928:01 08 CD 05 08 90 07 38
DB|0930:ED 05 08 2E 06 08 0A A0
9A|0938:05 CD 05 08 90 04 38 ED
38|0940:05 08 2E 06 08 4E 05 08
19|0948:88 D0 EE 8D 07 08 AD 06
D8|0950:08 18 65 F9 85 F9 A9 00
7B|0958:65 FA 85 FA 60 00 20 80
93|0960:20 00 21 80 21 00 22 80
4D|0968:22 00 23 80 23 28 20 A8
53|0970:20 28 21 A8 21 28 22 A8
29|0978:22 28 23 A8 23 50 20 D0
2E|0980:20 50 21 D0 21 50 22 D0
DB|0988:22 50 23 D0 23 00 04 08
78|0990:0C 10 14 18 1C A0 00 B1
DA|0998:06 C9 FF D0 03 4C E6 09
6B|09A0:8D 01 08 C8 B1 06 0A 8D
8A|09A8:00 08 C8 B1 06 0A 8D 0A
DF|09B0:08 C8 B1 06 0A 8D 0B 06
3C|09B8:98 48 AE 0B 08 20 06 09
3C|09C0:EE 00 08 AC 0A 08 A9 00
46|09C8:91 F9 88 D0 FB 91 F9 CA
C3|09D0:D0 EB 68 A8 C8 4C A5 09
39|09D8:60
TOTAL: F1C9
```

END OF LISTING 5

Listing 6: SOUNDS.OBJ

```
Start: 800    .      Length: 5C
8E|0800:8A 48 98 48 A2 20 8D 30
0D|0808:C0 A0 80 88 D0 FD CA D0
D9|0810:F5 68 A8 68 AA 60 8A 48
2D|0818:98 48 A2 FF 2A 90 03 8D
0D|0820:30 C0 48 8A A8 68 88 D0
43|0828:FD CA D0 F0 A2 01 2A 90
FE|0830:03 8D 30 C0 48 8A A8 68
08|0838:88 D0 FD E8 D0 F0 68 A8
C0|0840:68 AA 60 8A 48 98 48 A2
B2|0848:FF 2A 90 03 8D 30 C0 A0
98|0850:10 88 D0 FD CA D0 F2 68
FC|0858:A8 68 AA 60
TOTAL: 5178
```

END OF LISTING 6

Listing 7: CHICKEN.OBJ.S Source Code

```
0003        ;CHICKEN LITTLE
0004        ;BY JENNY SCHMIDT
0005        ;COPYRIGHT (C) 1991
0006        ;MINDCRAFT PUBLISHING CORP.
0007        A1=$3C              ;FOR MOVE ROUTINE
0008        A2=$3E
0009        A4=$42
0010        MOVE=$FE2C
0011        GRAFIC=$C050          ;SOFT SWITCHES
0012        MIXED=$C053
0013        HIRES=$C057
0014        TEXT=$C051
0015        PAGE=$804           ;SELECTS HIRES PAGE
0016        INPUT=$300           ;SELECT PADDLES OR KEYBD
0017        EGGS=$D00           ;ARRAY FOR EGG STATE
0018        EGGX=$D08           ;ARRAY FOR EGG X COORD
0019        EGGY=$D10           ;ARRAY FOR EGG Y COORD
0020        BIRDX=$D18           ;ARRAY FOR BIRD X COORD
0021        BIRDY=$D20           ;ARRAY FOR BIRD Y COORD
0022        SHAPAD=$FB           ;ADDRESS OF SHAPE TABLE
0023        SHAPET=$FD           ;ADDRESS OF PLOTTING TABLE
0024        PADDLE=$FB1E          ;READ PADDLE ROUTINE
0025        DRAW=$80E           ;DRAW ROUTINE
0026        YPART=$D28           ;ARRAY FOR Y COORDINATE INXS
0027        EGGINC=$D30          ;ARRAY FOR X INCREMENTS
0028        COMPRS=$D38          ;STATE OF SPRING
0029        SPRING=$D39          ;X COORDINATE OF SPRING
0030        RNDOM1=$D3A          ;RANDOM NUMBERS
0031        RNDOM2=$D3B
0032        RNDOM3=$D3C
0033        SCORE=$D3D           ;PLAYER'S SCORE
0034        GENRAT=$D3F          ;INCREMENTED FOR RANDOM #
0035        ERASE=$9A3          ;ERASE ROUTINE
0036        COORDS=$06          ;ADDRESS OF ERASE TABLE
0037        ERSDAT=$C00          ;LOCATION OF ERASE TABLE
0038        DRWDAT=$B00          ;LOCATION OF PLOTTING TABLE
0039        PURPLE=$01          ;COLORS
0040        GREEN=$02
0041        WHITE1=$03
0042        BLUE=$81
0043        ORANGE=$82
0044        WHITE2=$83
0045        COUNT=$D40           ;COUNTS MISSED EGGS
0046        CH=$24               ;HORIZONTAL CURSOR POSN
0047        TABV=$FB5B           ;SETS VERTICAL CURSOR POSN
0048        COUT=$FDED           ;PRINTS LETTERS
0049        PRNTAX=$F941          ;PRINTS NUMBERS
0050        SOUND=$9F0           ;SOUND ROUTINES
0051        SOUND1=$A06
0052        SOUND2=$A33
0053        KEYS=$C000
0054        STROBE=$C010
0055
0056         *=$D41
0057
0058  START  LDA #$00          ;INITIALIZE
0059         TAY
0060         STA COORDS
0061         STA SCORE
0062         STA SCORE+1
0063         STA A1
0064         STA PAGE
0065         STA $2000
0066         LDA #$20
0067         STA A1+1
0068         STA A4+1
0069         LDA #$01
0070         STA A4
0071         LDA #$3F
0072         STA A2+1
0073         LDA #$FF
0074         STA A2
0075         JSR MOVE          ;HGR
0076         STA GRAFIC        ;TURN ON GRAGHICS
0077         STA MIXED         ;TURN ON TEXT/GR MODE
0078         STA HIRES         ;TURN ON HIGH RESOLUTION
0079
0080         LDA #$0A          ;INITIALIZE
0081         STA SHAPAD+1
0082         LDA #$D1
0083         STA SHAPAD
0084         LDA #$0B
0085         STA SHAPET+1
0086         LDA #$00
0087         STA SHAPET
0088         LDA #$0C
0089         STA COORDS+1
0090
0091         LDA #00           ;ZERO EGG STATE
0092         LDY #07
0093  ZERO   STA EGGS,Y
0094         DEY
0095         BPL ZERO
0096
0097         LDA #50           ;INITIALIZE
0098         STA RNDOM3
0099         LDA #15
0100         STA RNDOM1
0101         LDA #47
0102         STA SPRING
0103
0104         LDA #06           ;MAXIMUM NO OF LOST EGGS
0105         STA COUNT
0106
0107  DATA
0108         .BYT $A0,$BA,$C5,$D2,$CF,$C3,$D3
0109
0110         LDA #14           ;SET CURSOR POSITION
0111         STA CH
0112         LDA #20
0113         JSR TABV
0114
0115         LDX #06           ;PRINT "SCORE: "
0116  PRINT  LDA DATA,X
0117         JSR COUT
0118         DEX
0119         BPL PRINT
0120
0121  TOP    LDA #06            ;SPRING IS NORMALLY OPEN
0122         STA COMPRS        ;POSITION
0123
0124         LDA #21           ;PRINT PLAYER'S SCORE
0125         STA CH
0126         LDA #20
0127         JSR TABV
0128         LDA SCORE
0129         LDX SCORE+1
0130         JSR PRNTAX
0131
0132         LDA RNDOM1        ;PLOT HEN?
0133         CMP #15
0134         BNE HATCH
0135         LDA RNDOM3
0136         STA DRWDAT
0137         LDA #00
```

```
0138          STA DRWDAT+1
0139          STA DRWDAT+4
0140          LDA #PURPLE
0141          STA DRWDAT+2
0142          LDA #PURPLE
0143          STA DRWDAT+3
0144          LDA #255
0145          STA DRWDAT+9
0146          JSR DRAW
0147
0148  HATCH   DEC RNDOM2        ;HATCH A CHICK?
0149          BNE NOLAY
0150          LDY #07           ;LOOK FOR EGG
0151  LOOK    LDA EGGS,Y
0152          CMP #01
0153          BNE NOEGG
0154          CLC               ;ADD 25 TO SCORE FOR
0155          SED               ;HATCHING EGGS
0156          LDA #$25
0157          ADC SCORE+1
0158          STA SCORE+1
0159          BCC CLEAR2
0160          LDA #$00
0161          ADC SCORE
0162          STA SCORE
0163 CLEAR2   CLD
0164          LDA #03           ;CHANGE EGG STATE TO
0165          STA EGGS,Y        ;BROKEN SHELL
0166          LDA EGGX,Y
0167          STA BIRDX,Y       ;GET BIRD COORDINATES
0168          LDA EGGY,Y
0169          STA BIRDY,Y
0170          JSR SOUND2        ;PLLAY HATCHED EGG SOUND
0171          JMP NOLAY
0172 NOEGG    DEY
0173          BPL LOOK
0174
0175 NOLAY    DEC RNDOM1        ;DROP EGG?
0176          BNE BLANK
0177          LDA RNDOM3        ;ERASE HEN
0178          STA ERSDAT
0179          LDA #00
0180          STA ERSDAT+1
0181          LDA #02
0182          STA ERSDAT+2
0183          LDA #07
0184          STA ERSDAT+3
0185          LDA #255
0186          STA ERSDAT+4
0187          JSR ERASE
0188
0189          LDY #07           ;FIND AN EGG TO DROP
0190 FIND     LDA EGGS,Y
0191          BNE FULL
0192          LDA #01
0193          STA EGGS,Y
0194          LDA RNDOM3        ;GET X,Y COORDINATES
0195          STA EGGX,Y
0196          LDA #00
0197          STA EGGINC,Y
0198          STA YPART,Y
0199          STA EGGY,Y
0200          JMP RANDOM        ;GET NEW RANDOM NUMBERS
0201 FULL     DEY
0202          BPL FIND
0203
0204 RANDOM   LDA GENRAT        ;GET "RANDOM" NUMBERS
0205          CMP #15
0206          BCS HIGH
0207          LDA #100
0208 HIGH     STA RNDOM1
0209          ROL A
0210          STA RNDOM2
0211          LSR A
0212          CMP #126
0213          BCC WITHIN
0214          LDA #125
0215 WITHIN   STA RNDOM3
0216
0217 BLANK    LDA SPRING        ;SET UP VALUES FOR ERASE
0218          STA ERSDAT
0219          LDA #58
0220          STA ERSDAT+1
0221          LDA #02
0222          STA ERSDAT+2
0223          LDA #05
0224          STA ERSDAT+3
0225          LDX #04
0226          LDY #07
0227 NEXT     LDA EGGS,Y
0228          BEQ NONE          ;DON'T ERASE IF EGG STATE
0229          LDA EGGX,Y        ;IS ZERO
0230          STA ERSDAT,X
0231          INX
0232          LDA EGGY,Y
0233          STA ERSDAT,X
0234          INX
0235          LDA #02
0236          STA ERSDAT,X
0237          INX
0238          LDA #11
0239          STA ERSDAT,X
0240          INX
0241          LDA EGGS,Y
0242          CMP #03
0243          BNE NONE
0244          LDA BIRDX,Y       ;ERASE THE BIRDS, TOO
0245          STA ERSDAT,X
0246          INX
0247          LDA BIRDY,Y
0248          STA ERSDAT,X
0249          INX
0250          LDA #01
0251          STA ERSDAT,X
0252          INX
0253          LDA #07
0254          STA ERSDAT,X
0255          INX
0256 NONE     DEY
0257          BPL NEXT
0258          LDA #255
0259          STA ERSDAT,X
0260
0261
0262 BOTTOM   LDY #07           ;CHECK FOR CRASHED EGGS
0263 CHECK    LDA EGGS,Y
0264          BNE SKIP
0265          JMP NOEGGS
0266 SKIP     LDA EGGY,Y        ;IS EGG NEAR THE BOTTOM
0267          CMP #52           ;OF THE SCREEN
0268          BEQ SKIPPR
0269          JMP NOEGGS
0270 SKIPPR   LDA SPRING        ;IS IT ON THE SPRING?
0271          SEC
0272          SBC EGGX,Y
0273          CMP #06
0274          BCC HIT           ;EGG LANDED ON SPRING
0275          CMP #243
0276          BCC MISS
0277          JMP HIT1          ;EGG LANDED ON SPRING
0278 MISS     LDA EGGS,Y        ;FIND OUT IF EGG ALREADY
0279          CMP #03           ;HATCHED
0280          BNE SKIPS
0281          JMP NOEGGS
0282 SKIPS    LDA #00
0283          STA EGGS,Y
0284          LDA EGGX,Y        ;DRAW RAW EGG
0285          STA DRWDAT
0286          STA DRWDAT+5
0287          LDA #73
0288          STA DRWDAT+1
0289          STA DRWDAT+6
0290          LDA #WHITE2
0291          STA DRWDAT+2
0292          STA DRWDAT+3
0293          LDA #05
0294          STA DRWDAT+4
0295          LDA #ORANGE
0296          STA DRWDAT+7
0297          LDA #ORANGE
0298          STA DRWDAT+8
0299          LDA #04
0300          STA DRWDAT+9
0301          LDA #255
0302          STA DRWDAT+14
0303          TYA
0304          PHA
0305          JSR DRAW
0306          PLA
0307          TAY
0308          JSR SOUND1        ;PLAY SMASHED EGG SOUND
0309          DEC COUNT         ;DECREASE ALLOWED EGGS CCTR
0310          BNE NOEGGS        ;IF ZERO, END THE GAME
0311          JSR SOUND1        ;PLAY END OF GAME SOUNDS
0312          JSR SOUND1
0313          JSR SOUND1
0314 READ     LDA KEYS          ;IS THE BUTTON BEING PUSHED
0315          BPL READ
0316          CMP #$9B
0317          BNE SPACE
0318          LDA STROBE
0319          LDA TEXT          ;END
0320          RTS
0321 SPACE    CMP #$A0
0322          BNE READ
0323          LDA STROBE
0324          JMP START         ;RESTART GAME
0325 HIT      PHA
0326          CLC               ;ADD 10 POINTS TO SCORE
0327          SED               ;FOR HITTING PADDLE
0328          LDA #$10
0329          ADC SCORE+1
0330          STA SCORE+1
```

```
0331          BCC CLEAR
0332          LDA #00
0333          ADC SCORE
0334          STA SCORE
0335 CLEAR    CLD
0336          PLA
0337          CMP #06          ;LEFT, MIDDLE, OR RIGHT
0338          BCS STRAIT       ;HIT MIDDLE OF SPRING
0339          LDA #254
0340          STA EGGINC,Y     ;MAKE EGG TRAVEL IN
0341          LDA #07          ;NEGATIVE X DIRECTION
0342          STA COMPRS       ;COMPRESS THE SPRING
0343          JMP NOEGGS
0344 HIT1     PHA
0345          CLC
0346          SED
0347          LDA #$10         ;ADD TEN POINTS TO SCORE
0348          ADC SCORE+1      ;FOR HITTING THE SPRING
0349          STA SCORE+1
0350          BCC CLEAR1
0351          LDA #00
0352          ADC SCORE
0353          STA SCORE
0354 CLEAR1   CLD
0355          PLA
0356          CMP #250
0357          BCS STRAIT
0358          LDA #02          ;MAKE EGG TRAVEL IN
0359          STA EGGINC,Y     ;POSITIVE X DIRECTION
0360          LDA #07
0361          STA COMPRS       ;COMPRESS THE SPRING
0362          JMP NOEGGS
0363 STRAIT   LDA #00           ;MAKE X INCREMENT ZERO
0364          STA EGGINC,Y
0365          LDA #07
0366          STA COMPRS       ;COMPRESS THE SPRING
0367 NOEGGS   DEY
0368          BMI CALC
0369          JMP CHECK
0370
0371 CALC     LDA INPUT
0372          BMI NOKEY
0373          LDA $C000
0374          CMP #$95         ;RIGHT ARROW
0375          BNE RETRY
0376          INC SPRING
0377          INC SPRING
0378          INC SPRING
0379          INC SPRING
0380          LDA SPRING
0381          BPL MOVING
0382          LDA #$00
0383          STA SPRING
0384          JMP MOVING
0385 RETRY    CMP #$88          ;LEFT ARROW
0386          BNE MOVING
0387          DEC SPRING
0388          DEC SPRING
0389          DEC SPRING
0390          DEC SPRING
0391          LDA SPRING
0392          BPL MOVING
0393          LDA #125
0394          STA SPRING
0395          JMP MOVING
0396 NOKEY    LDX #00          ;READ PADDLE
0397          JSR PADDLE
0398          TYA
0399          LSR A            ;DIVIDE BY TWO
0400          CMP #125
0401          BCC INSIDE       ;IGNORE #251-#255
0402          LDA #125
0403 INSIDE   STA SPRING
0404
0405 MOVING   LDX #07          ;ADVANCE EGGS, BIRDS
0406 MAIN     LDA EGGS,X
0407          BEQ DOWN
0408          LDY YPART,X      ;INDEX FOR TABLE STORED
0409          INC YPART,X      ;IN YPART
0410          LDA TABLE,Y      ;Y VALUE FOR EGGY
0411 ABOVE    BNE MIDDLE       ;END OF TABLE
0412          STA YPART,X      ;START OVER
0413          TAY
0414          LDA TABLE,Y
0415 MIDDLE   STA EGGY,X
0416          LDA EGGX,X       ;ADD X INCR TO EGGX
0417          CLC
0418          ADC EGGINC,X
0419          CMP #250         ;OFF THE SCREEN?
0420          BCC PLUS
0421          LDA #130         ;MOVE EGG TO OTHER SIDE
0422          JMP OKAY
0423 PLUS     CMP #131         ;OFF THE SCREEN?
0424          BCC OKAY
0425          LDA #00          ;MOVE EGG TO OTHER SIDE
0426 OKAY     STA EGGX,X
0427          LDA EGGS,X
0428          CMP #03          ;IS IT AN EGG SHELL?
0429          BNE DOWN
0430          LDA EGGY,X
0431          CMP #52
0432          BNE NODROP       ;EGG SHELL AT BOTTOM?
0433          LDA #66          ;YES, LEAVE IT THERE
0434          STA EGGY,X
0435          LDA #09
0436          STA YPART,X
0437          LDA #00
0438          STA EGGINC,X
0439 NODROP   DEC BIRDX,X      ;MOVE BIRD
0440          DEC BIRDX,X
0441          BPL CHECKY         ;REMOVE EGG SHELL AND
0442          LDA #00            ;BIRD IF OFF THE SCREEN
0443          STA EGGS,X
0444          JMP DOWN
0445 CHECKY   DEC BIRDY,X
0446          DEC BIRDY,X
0447          BPL DOWN           ;REMOVE EGG SHELL AND
0448          LDA #00            ;BIRD IF OFF THE SCREEN
0449          STA EGGS,X
0450 DOWN     DEX
0451          BMI DRAWIT
0452          JMP MAIN
0453
0454 DRAWIT   LDA SPRING       ;DRAW EVERYTHING
0455          STA DRWDAT
0456          LDA #58
0457          STA DRWDAT+1
0458          LDA #GREEN
0459          STA DRWDAT+2
0460          LDA #GREEN
0461          STA DRWDAT+3
0462          LDA COMPRS
0463          STA DRWDAT+4
0464
0465          LDX #05
0466          LDY #07
0467 DO       LDA EGGS,Y       ;IF EGG STATE IS ZERO
0468          BEQ DONE           ;DON'T DRAW ANYTHING
0469          LDA EGGX,Y         ;DRAW EGG OR EGG SHELL
0470          STA DRWDAT,X
0471          INX
0472          LDA EGGY,Y
0473          STA DRWDAT,X
0474          INX
0475          LDA #WHITE1
0476          STA DRWDAT,X
0477          INX
0478          LDA #WHITE1
0479          STA DRWDAT,X
0480          INX
0481          LDA EGGS,Y
0482          STA DRWDAT,X
0483          INX
0484          LDA EGGS,Y
0485          CMP #03            ;DRAW BIRD
0486          BNE DONE
0487          LDA BIRDX,Y
0488          STA DRWDAT,X
0489          INX
0490          LDA BIRDY,Y
0491          STA DRWDAT,X
0492          INX
0493          LDA #BLUE
0494          STA DRWDAT,X
0495          INX
0496          LDA #BLUE
0497          STA DRWDAT,X
0498          INX
0499          LDA #02
0500          STA DRWDAT,X
0501          INX
0502 DONE     DEY
0503          BPL DO
0504          INX
0505          INX
0506          INX
0507          INX
0508          LDA #255
0509          STA DRWDAT,X
0510          JSR ERASE        ;ERASE EVERYTHING
0511          JSR DRAW         ;DRAW EVERY THING
0512          LDA COMPRS       ;IF SPRING IS COMPRESSED
0513          CMP #07
0514          BNE NONOIS
0515          JSR SOUND        ;PLAY SPRING SOUND
0516 NONOIS   INC GENRAT        ;INCREMENT RANDOM NUMBER
0517          JMP TOP
0518
0519 TABLE       ;Y VALUES THAT EGGY MAY
0520          ;ASSUME
0521
0522          .BYT 10,12,15,18,22,28,35,42,50,52
0523          .BYT 50,42,35,28,22,18,15,10,00
0524
0525          .END
```

END OF LISTING 7

Listing 8: SHAPER.OBJ.S Source Code

```
0003          ;MID RES SHAPE GENERATOR
0004          ;BY JENNY SCHMIDT
0005          ;COPYRIGHT (C) 1991, MINDCRAFT PUBL. CORP.
0006 YCOORD=$800
0007 XCOORD=$801
0008 TEMPSH=$802
0009 COLOR=$803
0010 PAGE=$804
0011 DIVISR=$805
0012 ANSWER=$806
0013 REMAIN=$807
0014 COORDS=$06
0015 ADDRSS=$F9
0016 SHAPAD=$FB
0017 SHAPET=$FD
0018 SHAPE=$1111              ;ANY INITIAL VALUE
0019 XTEMP=$808
0020 XTEMP1=$809
0021 LENGTH=$80A
0022 WIDTH=$80B
0023 COLOR1=$80C
0024 COLOR2=$80D
0025
0026          *=$80E
0027
0028                          ;MAIN LOOP
0029
0030          LDY #$00        ;GET X,Y,COLOR, AND SHAPE #
0031 INFO     LDA (SHAPET),Y
0032          STA XTEMP
0033          STA XTEMP1
0034          INY
0035          LDA (SHAPET),Y
0036          ASL A
0037          STA YCOORD
0038          INY
0039          LDA (SHAPET),Y
0040          STA COLOR1
0041          INY
0042          LDA (SHAPET),Y
0043          STA COLOR2
0044          INY
0045          TYA
0046          PHA
0047          LDA (SHAPET),Y
0048          BMI FINISH      ;DONE IF HIGH BIT SET
0049          ASL A
0050          TAY
0051          LDA (SHAPAD),Y ;STORE ADDRESS OF SHAPE
0052          STA GETBYT+1    ;TO BE WORKED ON
0053          STA DATA+1
0054          INY
0055          LDA (SHAPAD),Y
0056          STA GETBYT+2
0057          STA DATA+2
0058          JSR PLOT
0059          PLA
0060          TAY
0061          INY
0062          JMP INFO
0063 FINISH   PLA
0064          RTS
0065
0066 PLOT     LDX #$00        ;GET BYTE OF SHAPE DATA
0067 GETBYT   LDA SHAPE,X
0068          BPL NOLINE      ;HIGH BIT SET IF END OF LINE
0069          ASL A
0070          BMI RTRN        ;END OF SHAPE?
0071          INX
0072          INC YCOORD      ;MOVE DOWN TO NEXT LINE
0073          INC YCOORD
0074          LDA XTEMP
0075          STA XTEMP1
0076          JMP GETBYT
0077 RTRN     RTS
0078 NOLINE   STA TEMPSH
0079          LDA XTEMP1
0080          STA XCOORD
0081          LDA COLOR1
0082          STA COLOR
0083          JSR BITS
0084          INC YCOORD
0085 AGAIN    LDA XTEMP1      ;REPEAT SEG JUST PLOTTED
0086          STA XCOORD
0087 DATA     LDA SHAPE,X
0088          STA TEMPSH
0089          LDA COLOR2
0090          STA COLOR
0091          JSR BITS
0092          LDA XCOORD
0093          STA XTEMP1
0094          INX
0095          DEC YCOORD
0096          JMP GETBYT
0097
0098 BITS     ASL TEMPSH
0099          LDY #$07
0100 NEXT     ASL TEMPSH      ;POINT ON OR OFF
0101          BCC BLANK
0102          TYA
0103          PHA
0104          JSR DRAW        ;PLOT IT
0105          PLA
0106          TAY
0107 BLANK    INC XCOORD      ;NEXT POINT
0108          DEY
0109          BNE NEXT
0110          RTS
0111
0112 DRAW     JSR FNDCRD      ;GET ADDRESS FROM COORDINATES
0113          LDA COLOR
0114          LDY REMAIN
0115          BEQ DONE
0116          CPY #$06        ;IS DOT PAIR BETWEEN 2 BYTES
0117          BEQ BETWN
0118 BITPOS   ASL A           ;SHIFT COLOR OVR REMAIN TIMES
0119          DEY
0120          BNE BITPOS
0121 DONE     ORA (ADDRSS),Y  ;PLOT IT
0122          STA (ADDRSS),Y
0123          LDA COLOR       ;ORANGE AND BLUE OR GREEN AND
0124          AND #$80        ;PURPLE
0125          ORA (ADDRSS),Y
0126          STA (ADDRSS),Y
0127          RTS
0128 BETWN    LDA COLOR       ;DOT PAIR IS BETWEEN BYTES
0129          AND #%00000001 ;GET FIRST DOT
0130          ASL A           ;SHIFT IT TO LEFT MOST POSN
0131          ASL A
0132          ASL A
0133          ASL A
0134          ASL A
0135          ASL A
0136          LDY #$00        ;PLOT THE DOT
0137          ORA (ADDRSS),Y
0138          STA (ADDRSS),Y
0139          LDA COLOR
0140          AND #$80
0141          ORA (ADDRSS),Y
0142          STA (ADDRSS),Y
0143          INC ADDRSS
0144          BNE NOVER
0145          INC ADDRSS+1
0146 NOVER    LDA COLOR       ;GET SECOND DOT
0147          AND #%00000010
0148          LSR A           ;SHIFT IT TO RIGHT MOST POSN
0149          ORA (ADDRSS),Y
0150          STA (ADDRSS),Y
0151          RTS.
0152
0153 FNDCRD   LDA YCOORD      ;CALCULATE ADDRESS FROM
0154          LSR A           ;COORDINATES
0155          LSR A           ;DIVIDE BY FOUR
0156          LSR A
0157          ASL A           ;AND ROUND OFF TO NEAREST 2
0158          TAY
0159          LDA TABLE1,Y    ;GET ADDRESS
0160          STA ADDRSS
0161          INY
0162          LDA TABLE1,Y
0163          STA ADDRSS+1
0164          LDA YCOORD
0165          AND #%00000111
0166          TAY
0167          LDA TABLE2,Y
0168          CLC
0169          ADC ADDRSS+1
0170          CLC
0171          ADC PAGE        ;PAGE IS ZERO FOR PRIMARY
0172          STA ADDRSS+1    ;AND $20 FOR SECONDARY PAGE
0173          LDA #$00        ;DIVIDE XCOORDINATE BY 3.5
0174          STA ANSWER      ;BY MULTIPLYING TWO
0175          LDA #%01110000 ;AND DIVIDING BY SEVEN
0176          STA DIVISR
0177          LDA XCOORD
0178          CMP DIVISR
0179          BCC DOUBLE
0180          SEC
0181          SBC DIVISR
0182          ROL ANSWER
0183 DOUBLE   ASL A
0184          LDY #$05
0185 DIVIDE   CMP DIVISR
```

Listing 8: SHAPER.OBJ.S Source Code *continued*

```
0186          BCC ROTATE
0187          SEC
0188          SBC DIVISR
0189 ROTATE   ROL ANSWER
0190          LSR DIVISR
0191          DEY
0192          BNE DIVIDE
0193          STA REMAIN      ;REMAINDER CONTAINS BIT POSN
0194          LDA ANSWER      ;WITHIN THE BYTE OF THE X
0195          CLC             ;COORDINATE
0196          ADC ADDRSS
0197          STA ADDRSS      ;FINAL ADDRESS CALCULATED
0198          LDA #$00
0199          ADC ADDRSS+1
0200          STA ADDRSS+1
0201          RTS
0202
0203 TABLE1
0204          .BYT $00,$20,$80,$20,$00,$21,$80,$21,$00
0205          .BYT $22,$80,$22,$00,$23,$80,$23,$28,$20
0206          .BYT $A8,$20,$28,$21,$A8,$21,$28,$22,$A8
0207          .BYT $22,$28,$23,$A8,$23,$50,$20,$D0,$20
0208          .BYT $50,$21,$D0,$21,$50,$22,$D0,$22,$50
0209          .BYT $23,$D0,$23
0210
0211 TABLE2
0212          .BYT $00,$04,$08,$0C,$10,$14,$18,$1C
0213
0214                          ;ERASES SCRN BLOCK W STARTING
0215                          ;COORD, WIDTH, & LNGTH OF BLOCK
0216
0217 ERASE    LDY #$00         ;GET X, Y, WIDTH, AND LENGTH
0218 MORE     LDA (COORDS),Y
0219          CMP #$FF
0220          BNE CONTNU       ;HIGH BIT SET SIGNALS END OF
0221          JMP STOP         ;DATA
0222 CONTNU   STA XCOORD
0223          INY
0224          LDA (COORDS),Y
0225          ASL A
0226          STA YCOORD
0227          INY
0228          LDA (COORDS),Y
0229          ASL A
0230          STA LENGTH
0231          INY
0232          LDA (COORDS),Y
0233          ASL A
0234          STA WIDTH
0235          TYA
0236          PHA
0237          LDX WIDTH
0238 GTADRS   JSR FNDCRD       ;CALCULATE ADDRESS
0239          INC YCOORD
0240          LDY LENGTH
0241          LDA #$00
0242 CLEAR    STA (ADDRSS),Y   ;ZERO THE BYTE
0243          DEY
0244          BNE CLEAR
0245          STA (ADDRSS),Y
0246          DEX
0247          BNE GTADRS
0248          PLA
0249          TAY
0250          INY
0251          JMP MORE
0252 STOP     RTS
0253
0254          .END
```

END OF LISTING 8

Listing 9: SOUND GENERATOR

```
0003                          ;SOUND GENERATOR
0004                          ;BY JENNY SCHMIDT
0005
0006 CLICK=$C030
0007
0008          *=$9E0
0009          ;FIRST SOUND
0010
0011          TXA             ;SAVE X,Y
```

```
0012          PHA
0013          TYA
0014          PHA
0015          LDX #$20        ;LENGTH OF SOUND
0016 AGAIN    STA CLICK        ;CLICK SPEAKER
0017          LDY #$80        ;PITCH
0018 PAUSE    DEY
0019          BNE PAUSE
0020          DEX
0021          BNE AGAIN
0022          PLA             ;RESTORE X,Y
0023          TAY
0024          PLA
0025          TAX
0026          RTS
0027
0028          ;SECOND SOUND
0029
0030          TXA             ;SAVE X,Y
0031          PHA
0032          TYA
0033          PHA
0034          LDX #$FF        ;LENGTH OF SOUND
0035 LOOP     ROL A            ;CLICK SPEAKER WHEN
0036          BCC SILENT      ;THE CARRY IS SET
0037          STA CLICK
0038 SILENT   PHA
0039          TXA
0040          TAY
0041          PLA
0042 GAP      DEY              ;PITCH DEPENDS ON X
0043          BNE GAP
0044          DEX             ;THEREFORE IT CONTINUES
0045          BNE LOOP        ;TO RISE
0046
0047          LDX #$01
0048 LOOP1    ROL A            ;CLICK SPEAKER WHEN
0049          BCC SILNT1      ;THE CARRY IS SET
0050          STA CLICK
0051 SILNT1   PHA
0052          TXA
0053          TAY
0054          PLA
0055 GAP1     DEY              ;PICTH DEPENDS ON X
0056          BNE GAP1
0057          INX             ;THEREFORE IT DROPS
0058          BNE LOOP1
0059          PLA             ;RESTORE X,Y
0060          TAY
0061          PLA
0062          TAX
0063          RTS
0064
0065          ;SOUND 3
0066          TXA             ;SAVE X,Y
0067          PHA
0068          TYA
0069          PHA
0070          LDX #$FF        ;LENGTH OF SOUND
0071 MOVEIT   ROL A            ;CLICK SPEAKER IF
0072          BCC NONOIS      ;CARRY IS SET
0073          STA CLICK
0074 NONOIS   LDY #$10         ;PITCH
0075 REDUCE   DEY
0076          BNE REDUCE
0077          DEX
0078          BNE MOVEIT
0079          PLA             ;RESTORE X,Y
0080          TAY
0081          PLA
0082          TAX
0083          RTS
0084
0085          .END
```

END OF LISTING 9

Ask Nibble

By Roger Wagner

continued from page 4

Q What is the "alternate display mode" on my Apple IIGS, and how can I switch alternate display mode on without going to the Control Panel?

Joe Hass
Albuquerque, NM

A Joe, the alternate display mode is actually not a new display mode, per sé, but the name of a program that is built in to your computer. (Who says the Apple IIGS doesn't come with useful software right out of the box?)

Your computer has a chip in it called the Mega II, which allows the machine to run Apple IIe software. The display screens on an Apple IIGS reside in banks $E0 and $E1, while on a IIe, they're in the main bank (bank $00) and the auxiliary bank (bank $01). Screens include the main text page (40 and 80 columns are covered up there), the Hi-Res pages, and the double Hi-Res screen.

The screen that we're interested in, however, is the second text page, which normally resides on the Apple IIGS in bank $E0 from location $800 to $A00. In order to allow Apple IIe software to function properly, the Mega II is supposed to copy all the data for the screen being currently displayed from bank $00 up into bank $E0 when in Apple II emulation mode, so that Apple IIe programs will show the correct display. (An Apple IIe program doesn't know to change the display in bank $E0, so it changes that one that it does know, down in bank $00.) The Mega II does the job fine for all of the screens except for the second text page, which doesn't get copied.

This wasn't discovered until it was too late to change the chip (but cut the designers some slack; they were busy adding all of the other neat features to the Apple IIGS, like Super Hi-Res). Luckily, not many Apple IIe programs actually use the second text page. When you're running one that does, however, you may be presented with a screen full of "2"s instead of the text you expect. (For some reason, the ASCII value for a "2" character seems to be what the second text page in bank $E0 is filled with after the machine is turned on.)

As a result, the Apple IIGS has the Alternate Display Mode CDA, which installs a small program which runs in the background and continually copies the text page two from bank $00 up to bank $E0.

If all of this sounds a little fishy to you, try this short experiment. Enter Applesoft BASIC and type the following commands

```
CALL -151
C055
```

The first command will take you into the Monitor, and the second will toggle page 2 on. On the display now is the second text page, as it exists in bank $E0. Make note of the screen, go to the Control Panel (Control-Apple-ESC) and turn Alternate Display Mode on. The screen will be different when you exit the Control Panel. (Turning Alternate Display Mode back off will not change the screen again; the image in bank $00 has already been copied to bank $E0.)

I don't know of any way to turn Alternate Display Mode on without using the Control Panel; however, this shouldn't be a problem, as it is so rarely used and it's not particularly useful except for the purpose that it was designed.