ProDOS DOS 3.3

LIFT OFF

Convert sections of the Hi-Res screen to shape tables with this combination Applesoft and machine language program. Save the tables to disk for later use in your programs.

by Steven Wong
1427 67th St.
Brooklyn, NY 11219

We have all seen shapes in game programs that we would like to use in our own programs. Until now, the only way to accomplish this was to re-create them. Depending on the complexity of the shapes, re-creation could take hours. LIFT.OFF is a program that will perform this tedious task for you quite easily.

LIFT.OFF is a graphics utility which enables you to convert anything on the Hi-Res screen into a usable vector shape. The entire process usually takes a couple of minutes from start to finish.

Using LIFT OFF

Upon running this program, a short menu will be displayed. There are four options available:

1. LOAD SCREEN — This is used to load a Hi-Res screen from disk. The program will ask you for the file name. If you press the RETURN key without typing a file name, the disk Catalog will be displayed. After the screen has been loaded into memory, you will immediately be sent to the editor mode.

2. EDITOR — This command places you in the editor mode. In this mode, you will be able to edit or create a shape. Use the familiar I, J, K, and M keys to move the blinking cursor around the screen.

   Pressing A will plot a point and pressing S will erase a point. Pressing W locks plotting and pressing E locks erasing. Press Q to unlock either one. Pressing the ESC key will switch between full and mixed screen graphics. You can return to the menu at any time by pressing the "at" (@) key.

   Note: When you enter the editor mode, the CLEAR HIBIT routine will be CALLed. While this routine does not physically affect the Hi-Res screen, it does alter its colors.

> **"This option permits you to isolate the part of the Hi-Res screen to be converted."**

3. BOX IN SHAPE — this option permits you to isolate the part of the Hi-Res screen to be converted into a shape. If you are not familiar with the layout of the Hi-Res screen, the simple chart (**Figure 1**) should be of help.

   There are four values which can be changed: the top, bottom, left, and right. To change the value of a certain edge, press the **space bar**. When the flashing cursor reaches the name of the edge you wish to modify, press the **RETURN** key and then make the change.

   You can keep changing the values until you are satisfied. The Hi-Res rectangle will be altered upon the entry of each new value.

   Once you are satisfied, press the CONTROL key while typing S (**CTRL-S**) to start the converting process. You will be asked if you would like a negative shape. Entering "no" will convert the shape exactly the way it is, while entering "yes" will convert the shape into a vector shape containing its opposite colors.

   At this point, the machine language program takes over and does the actual conversion. Within a second or two (the advantage of machine language), you will see the result of your efforts.

   After you are done admiring your creation, press <**RETURN**>. You will be asked if you want to save the shape. If you are satisfied with the shape, enter "yes" and then the name you wish to save it under. Once the shape is saved, the disk Catalog will be displayed before you are sent back to the menu.

   Note: The CLEAR HIBIT routine is also CALLed in the "box in shape" mode.

4. QUIT — This option allows you to exit gracefully from the program.

Rules

There are a few rules you should be aware of when using LIFT OFF:

1. The difference between the top and bottom values must be at least two.

2. The difference between the left and right values must be at least three.

3. The top value cannot be greater than the bottom value.

4. The left value cannot be greater than the right value.

5. You cannot convert a shape which is greater than 8000 pixels in size. The formula **(bottom-top) * (right-left)** will compute the number of pixels in the shape.

Don't worry too much about breaking these rules. The program detects all transgressions and will not allow you to proceed to the conversion stage until you have rectified them. You can always return to the menu by pressing the '@' key.

IMPORTANT: The shape to be converted must be set flush against each side of the Hi-Res rectangle. It should not touch or exceed any of these boundaries.

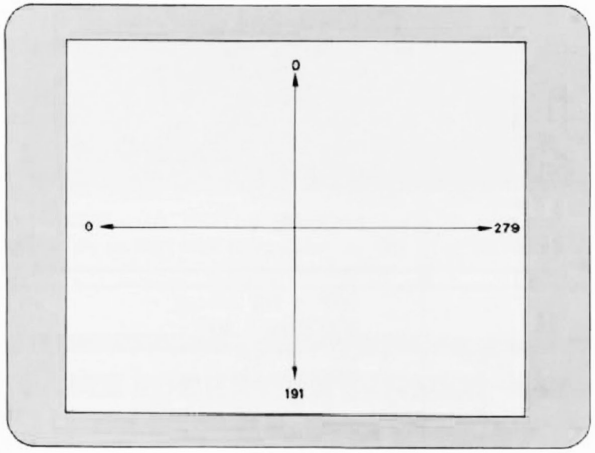

FIGURE 1

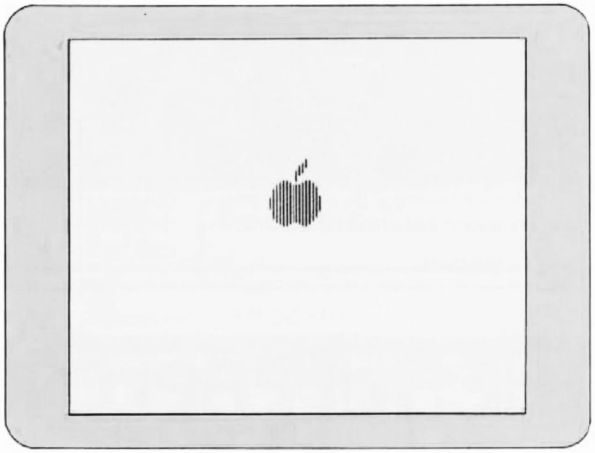

FIGURE 2

Typing in the Program

LIFT OFF is composed of two parts, a BASIC program and a machine language program, both of which are relatively short. The program requires at least 48K of RAM and Applesoft in ROM or in a RAM card.

First, type in the BASIC program (**Listing 1**) and save it under the name **LIFT.OFF**.

Next, enter in the machine language program. (See **Listing 2**, the source listing.) If you do not own an assembler or if you do not understand assembly language, refer to "A Welcome to New *Nibble* Readers" for instructions on entering machine code.

Once you have entered the program, save it by typing **BSAVE LIFT.OFF.ML, A$1EDC,L$DF**.

Capturing A Shape: A Case Study

I will show you how simple the procedure of capturing a shape really is.

1. Type **NEW** < **RETURN** > to clear memory, and type in the short program from **Listing 3** named "APPLE.CREATOR."
2. **RUN** it. You should see a green apple appear in the middle of the screen.
3. **RUN LIFT.OFF.**
4. When the menu options appear, enter **3** < **RETURN** > for option 3.
5. You should see the apple surrounded by a large rectangle. (See **Figure 2**.) Press < **RETURN** >, and the names of the rectangle sides along with their values will be displayed.
6. Start to enclose the apple shape, moving the BOTTOM side first. Press the **space bar** until it is aligned with the name of the side you are modifying; in this case, BOTTOM. Then press < **RETURN** > and enter in **97** < **RETURN** > when the "ENTER?" question appears. You should see the Hi-Res rectangle re-justifying itself.
7. Now move the LEFT side. Press < **RETURN** > and follow the same procedure outlined in step 6, but apply it to the LEFT side. Enter **122** < **RETURN** > for the LEFT value.
8. At this point, your screen should look like **Figure 3**. (If it does not, you have made a mistake somewhere along the way. I suggest you restart from the beginning.)
9. Now move the TOP side. Again follow the procedure outlined in step 6. Enter **61** < **RETURN** > for the TOP value.
10. Finally, move the RIGHT side. Enter **154** < **RETURN** > for the RIGHT value.
11. Your screen should look like **Figure 4**. Notice that the sides of the rectangle are completely flush against the apple shape without touching it.
12. Press < **RETURN** >. The correct values for each side are:

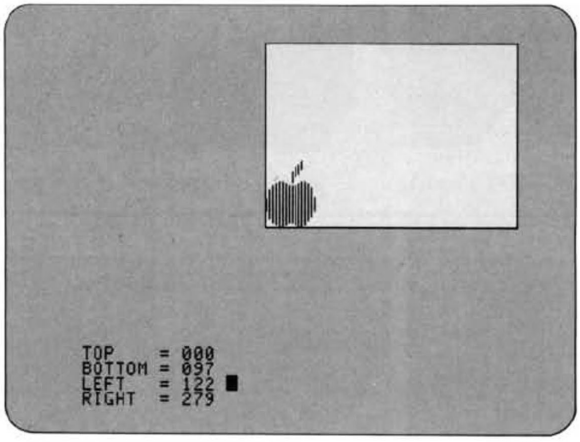

FIGURE 3

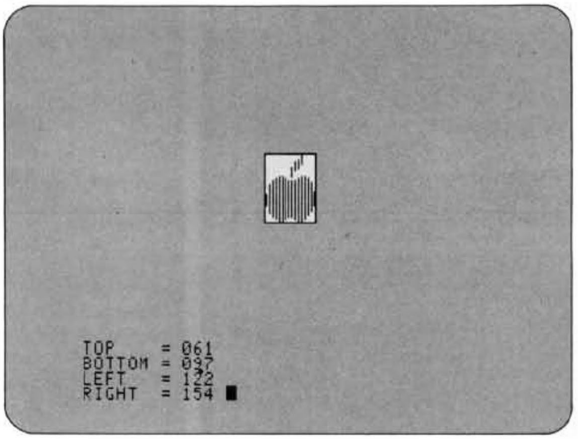

FIGURE 4

TOP = 061
BOTTOM = 097
LEFT = 122
RIGHT =154

If any of your values are different, correct them to match the ones above.

13. Now press **CTRL-S** and type **NO** when the "NEGATIVE SHAPE?" question appears.

14. Within a few seconds, the captured shape should appear.

15. Press **<RETURN>** and enter **YES** when the question "SAVE SHAPE?" appears. Then enter **APPLE** as the file name.

16. After the disk drive stops, press **<RESET>** and type **NEW <RETURN>**.

17. Now, enter the short program from **Listing 4** named "SHAPE.DEMO" and **RUN** it.

18. If you have correctly followed all the instructions outlined, you should see four colorful apples and an interesting animated demonstration.

I hope you now realize how simple capturing shapes really is. Take some time to learn the "ins and outs" of this program — and don't be afraid to experiment!

How LIFT OFF Works

The Applesoft Program LIFT.OFF

Lines 110-160 contain various short subroutines. The subroutine at **line 110** may be of interest to you. It is a simple method I have devised to simulate the Print Using function. This useful function is available in the BASICs of many computers, but unfortunately, is not available in Applesoft.

Lines 180-240 contain the input subroutine. I, like many programmers, hate to use the INPUT statement, but I will normally write a small INPUT routine when it is necessary for the users of my programs to enter any substantial amount of information.

Basically, this subroutine builds the string I$ with the characters obtained via the GET function. Special characters such as backspaces are checked for and handled accordingly. When a <RETURN> is detected, this subroutine returns back to its caller. At that point, I$ will contain the string obtained from the user, J$ will contain string I$'s first character, and the variable I will contain the value of I$, if it is numerical.

Lines 250-390 contain the editor routine. **Lines 250-280** set the Hi-Res and text screens and CALL the CLEAR HIBIT routine. All the editing commands are checked for and processed in **lines 290-390**.

> **"...notice the heavy use of Boolean expressions."**

As you scan through this section, you will notice the heavy use of Boolean expressions. This was done mainly to decrease the size of the program. While I did have 64K of RAM to work with, my goal was to write this program in less than 4K. This would not have been possible if the program had been written with the scores of IF...THEN statements that were replaced by the Boolean expressions.

Lines 400-450 contain the menu. This section of the program displays the program title, author's name and the copyright notice. The four options available in the menu are printed in **line 430**. **Lines 440-450** wait for the user's choice and process it accordingly.

Line 460 allows you to quit the program.

Lines 470-500 load the screen and get the file name from the user. If the user enters no name, then the disk Catalog is displayed.

Lines 510-730 contain the "box in shape" routine. The variables used by this part of the program are initialized at **line 510**. The CLEAR HIBIT routine is CALLed. **Lines 520-550** may be the most complex part of the program. (The algorithm alone took me over a day to write.) It is a very small routine which simulates the nonexistent Applesoft command, XHPLOT. XHPLOT, if there were such a command in Applesoft, would operate exactly like HPLOT, except that it would provide a nondestructive way of line plotting. (HPLOT destroys everything that it plots over.)

> **"It is a very small routine which simulates the nonexistent Applesoft command, XHPLOT."**

By properly manipulating the SCALE, ROT and XDRAW commands, this routine has given me a satisfactory substitute for an XHPLOT command. And while this routine is in no way complete, it does meet the needs of this program.

Line 560 blinks the Hi-Res rectangle. **Lines 630-640** contain the routine which allows you to manipulate values of the Hi-Res rectangle. The input data is processed and checked for errors in **lines 650-730**.

Lines 740-780 contain the conversion routine. **Line 740** checks the size of the shape. If the shape is too large, it informs you and loops back to the "box in shape" routine. **Lines 760-770** POKE in the values of each side of the Hi-Res rectangle (so that the machine language program can find the shape).

The captured shape directory is set up in **line 780**. **Line 780** also CALLs the machine language program which actually does the conversion. **Line 790** draws the captured shape onto Hi-Res page 2. The variable J holds the length of the captured shape (used later to save the captured shape).

Lines 800-860 save the captured shape. This routine saves the captured shape onto disk at address $6000 (24576 dec.). After the shape is saved, the relevant information regarding the shape will be displayed.

Lines 870-920 perform initialization. **Line 860** loads the machine language program from disk into memory. **Line 880** POKEs in a Shape Table which consists of a single dot and the short machine language routine CLEAR HIBIT. This routine strips the high bits out of each byte in the address range of $2000-$3FFF (8192-16383 dec.). The variables and strings used by LIFT.OFF are initialized at **lines 890-910**.

The Machine Language Program

Essentially, the machine language program LIFT.OFF.ML scans a rectangular area (previously defined by the user) on the Hi-Res screen and converts the data it finds into a Vector Shape Table. (See pp. 92-99 of the *BASIC Programming Manual*, or pp. 150-166 of the *Applesoft BASIC Programmer's Reference Manual* for more information regarding Vector Shape Tables.)

Lines 030-040 initialize the pointers used by this program.

Lines 044-062 scan the screen from left to right and call the FIND routine to check the pixel at HLOC,VLOC. Depending on the condition of the pixel, a vector value of 5 or 1 will be placed into memory. (See **Figure 5**.) HLOC is incremented and checked to see if it has reached the right side of the Hi-Res rectangle.

Lines 066-085 perform functions similar to those performed by **lines 44-62**, except: they scan the screen from right to left, the vector codes used are 7 and 3 instead of 5 and 1, and HLOC is decremented and checked to determine whether it has reached the left side of the Hi-Res rectangle.

Lines 089-105 contain the FIND routine which checks the status (on/off) of the pixel at HLOC,VLOC (a sort of Hi-Res SCRN function). First, the value of HLOC is divided by 7 to locate the screen byte. The remainder is used to locate the bit within the byte. (Pixels are actually bits in a byte.)

The accumulator is then loaded with the value of the screen byte and logically ANDed to filter out the bits it is not presently testing. If the accumulator contains 0, then the pixel at HLOC,VLOC is off. If it contains any other value, then the pixel is on.

Lines 109-115 contain a routine that, after each screen line is completed, places a vector code of 6 or 2 into memory — again, depending on the condition of the pixel at HLOC,VLOC.

Lines 119-124 contain the PUI routine. This routine stores the vector codes into memory starting at $4000 (16384 dec.).

Lines 128-132 contain a routine that increments the variable VLOC by one and checks to see if it has reached the bottom of the Hi-Res rectangle. If it has not, then it CALLs the Monitor's HPOSN routine and loops back to scan another screen line.

HPOSN, among other things, calculates the Hi-Res screen's vertical base addresses. The addresses can be found at $26,$27 (38,39 dec.). When the bottom of the Hi-Res rectangle is reached, the scanning process is done and the conversion of the vector codes into Vector Shape Table data begins.

Figure 6 shows how the program scans the Hi-Res screen and stores proper vector codes into memory.

Lines 136-159 contain a routine that takes the vector codes from memory and converts them into a Vector Shape Table. Two vector codes are combined to form a single Vector Shape Table value. This value is found by adding the first vector code to the product of the second vector code and 8. (See **Figure 7**.) The Vector Shape Table values are stored beginning at $6004 (24580 dec.).

This process continues until a Vector Shape Table value of zero is computed. (The zero value will also be stored in memory to signal the end of the Vector Shape Table.) At that point, the conversion is over and control returns to the BASIC program.

VECTOR	CODE	FUNCTION
▲	0	MOVE UP, NO PLOT
►	1	MOVE RIGHT, NO PLOT
▼	2	MOVE DOWN, NO PLOT
◄	3	MOVE LEFT, NO PLOT
▲●	4	MOVE UP, PLOT
●►	5	MOVE RIGHT, PLOT
●▼	6	MOVE DOWN, PLOT
◄●	7	MOVE LEFT, PLOT

FIGURE 5

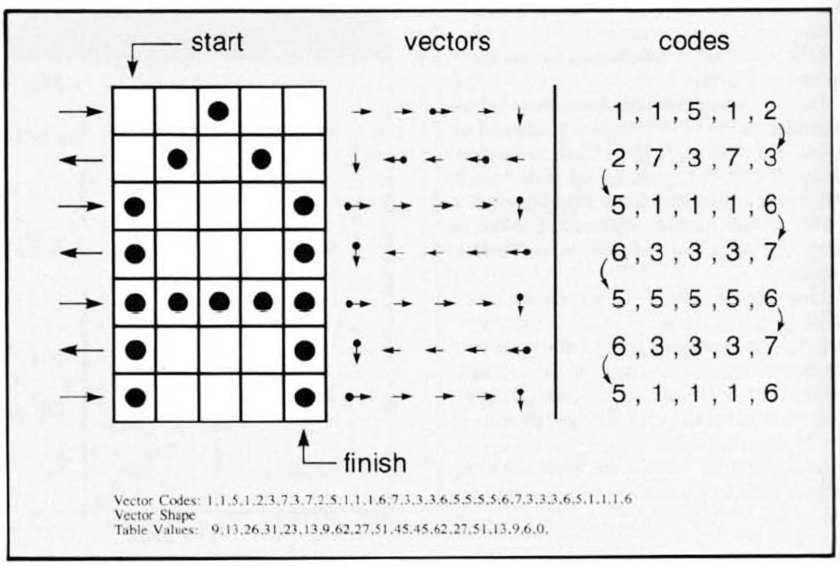

FIGURE 6

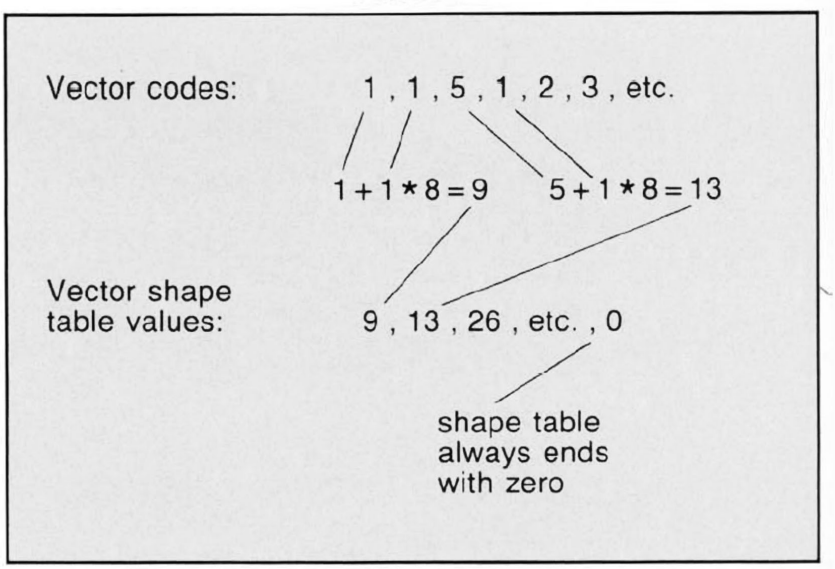

FIGURE 7

A Final Thought

If you would like to compile some of your captured shapes into a Shape Table, I suggest you obtain a copy of *Nibble* Vol. 2/No. 5. There you will find a program called HI-RES SHAPE COMBINER by Chris Carroll which performs this task for you. Otherwise, you might have to do it the hard way, by hand.

LISTING 1 — LIFT.OFF

```
20  REM ************************
30  REM *       LIFT.OFF       *
40  REM *    BY STEVEN WONG    *
50  REM *                      *
60  REM *  COPYRIGHT (C) 1984  *
70  REM *  BY MICROSPARC, INC  *
80  REM *  LINCOLN, MA. 01773  *
90  REM ************************
100  LOMEM: 32768: TEXT : HOME : GOTO 870
110  PRINT  RIGHT$ ("00" +  STR$ (J),3);: RETURN
120  POKE  - 16304,0: POKE  - 16297,0: RETURN
130  HOME : PRINT D$"CATALOG": PRINT : PRINT : GOTO
     150
140  CALL BL: HTAB 1: PRINT "DISK ERROR #" PEEK (222
     );
150  PRINT ". . . PRESS A KEY": CALL BL
160  POKE KR,0: WAIT KB,128: POKE KR,0: RETURN
170 I =  FRE (0):I$ = "":K =  POS (0) + 1
180  GET A$: IF A$ < S$ AND A$ <  > R$ AND A$ <  > B
     $ THEN 180
190  PRINT A$;: IF A$ = R$ THEN 240
200 I =  PEEK (SP) -  PEEK (SP): IF A$ <  > B$ THEN
     I$ = I$ + A$: GOTO 230
210  IF  LEN (I$) < 2 THEN  HTAB K: PRINT S$;: HTAB
     K: GOTO 170
220  PRINT S$B$;:I$ =  LEFT$ (I$, LEN (I$) - 1): GOTO
     180
230  IF  LEN (I$) < 16 THEN 180
240 I =  VAL (I$):J$ =  LEFT$ (I$,1): RETURN
250  POKE MS,0:K = 0:MC = 0: CALL 774: HOME : VTAB 2
     2: FOR I = 1 TO 40: PRINT "=";: NEXT
260  VTAB 23: PRINT M4$;: VTAB 23: HTAB 3: FLASH : PRINT
     M1$: POKE 50,63: VTAB 24
270  PRINT "A=ON  S=OFF  I-J-K-M  ?=MENU  ESC=SCREE"
     ;: POKE 2039,14: SCALE= 1: ROT= 0: GOSUB 120: NORMAL

280  VTAB 21: HTAB 14: PRINT "X=";:J = X: GOSUB 110:
      HTAB 23: PRINT "Y=";:J = Y: GOSUB 110
290 I =  PEEK (KB): XDRAW 1 AT X,Y:I = I - 128: XDRAW
     1: IF I < 0 THEN 290
300  POKE KR,0: IF I <  > 69 AND I <  > 81 AND I <  >
     87 THEN 350
310  VTAB 23: HTAB 1: PRINT M4$;: VTAB 23: FLASH : IF
     I = 81 THEN MC = 0: HTAB 3: PRINT M1$: GOTO 340

320  IF I = 87 THEN MC = 1: HTAB 15: PRINT M2$: GOTO
     340
330  IF I = 69 THEN MC = 2: HTAB 30: PRINT M3$;
340  NORMAL : GOTO 380
350 Z =  PEEK (SP) -  PEEK (SP):X = X + (I = 75) - (
     I = 74):X = X + ((X < 0) - (X > 279)) * 280
360 Y = Y + (I = 77) - (I = 73):Y = Y + ((Y < 0) - (
     Y > 191)) * 192: IF I = 27 THEN K =  NOT (K): POKE
     MS - K,0
370  IF  NOT MC THEN  IF I = 65 OR I = 83 THEN  HCOLOR=
     3 * (I = 65): HPLOT X,Y
380  IF MC THEN  HCOLOR= 3 * (MC = 1): HPLOT X,Y
390  IF I <  > 64 THEN 280
400 I =  FRE (0): POKE 216,0: POKE 230,32: POKE 233,
     3: TEXT : HOME : FOR J = 1 TO 3 STEP 2: VTAB J
410  FOR I = 1 TO 40: PRINT "-";: NEXT I,J: PRINT "
     COPYRIGHT (C) 1984 BY MICROSPARC, INC."
420  INVERSE : VTAB 2: PRINT  SPC( 7)"LIFT OFF     BY
      STEVEN WONG" SPC( 7): NORMAL
430  VTAB 10: PRINT "(1) LOAD SCREEN": PRINT "(2) ED
     ITOR": PRINT "(3) BOX IN SHAPE": PRINT "(4) QUI
     T"
440  CALL BL: VTAB 15: CALL BH: PRINT : HTAB 5: PRINT
     "YOUR CHOICE --> ";: GOSUB 170: IF  NOT I OR I >
     4 THEN 440
450  ON I GOTO 470,250,510
460  HOME : END
470  VTAB 20: PRINT "FILE NAME? ";: GOSUB 170: IF I$
      = "" THEN  GOSUB 130: GOTO 480
480  ONERR  GOTO 500
490  PRINT D$"BLOAD "I$",A$2000": POKE 216,0: GOTO 2
     50
500  VTAB 20: GOSUB 140: GOTO 440
510  CALL 774:A = 0:B = 0:C = 279:D = 191: GOSUB 120
     :L2 = 1630:L = 0: GOTO 560
520 J = C - A: ROT= 0: IF C < 256 THEN  FOR Z = 1 TO
     100: NEXT
530  IF J > 254 THEN  SCALE= C - 255 + 1: XDRAW 1 AT
     255,B: XDRAW 1 AT 255,D:J = 254 - A
540  SCALE= J + 1: XDRAW 1 AT A,B: XDRAW 1 AT A,D: ROT=
     16: SCALE= D - B - 1
550  XDRAW 1 AT A,B + 1: XDRAW 1 AT C,B + 1: ROT= 0:
      RETURN
560  POKE FS,0: FOR K = 1 TO 7:Z =  PEEK (SP): GOSUB
     520: NEXT : GOSUB 160: POKE MS,0
570  HOME : VTAB 21: PRINT " TOP    = ";:J = B: GOSUB
     110: PRINT : PRINT " BOTTOM = ";:J = D: GOSUB 1
     10
580  PRINT : PRINT " LEFT   = ";:J = A: GOSUB 110: PRINT
     : PRINT " RIGHT  = ";:J = C: GOSUB 110: POKE L2
     ,96
590  GOSUB 160:K =  PEEK (KB): IF K = 32 THEN L = (L
      + 1) * (L < 3):L1 = 1630 + L * 128: POKE L1,96
     : POKE L2,160:L2 = L1
600  IF K = 64 THEN  GOSUB 520: GOTO 400
610  IF K = 19 THEN 740
620  IF K <  > 13 THEN 590
630  POKE L2,160: VTAB 21 + L: HTAB 18: CALL BL: PRINT
     "ENTER? ";: GOSUB 170: IF J$ = " " OR I$ = "" THEN
     570
640  HOME : GOSUB 520: ON L + 1 GOTO 650,670,690,710
650  IF I < 0 OR I > 189 OR I + 2 > D THEN 730
660 B = I: GOTO 560
670  IF I < 2 OR I > 191 OR I - 2 < B THEN 730
680 D = I: GOTO 560
690  IF I < 0 OR I > 276 OR I + 3 > C THEN 730
700 A = I: GOTO 560
710  IF I < 3 OR I > 279 OR I - 3 < (A) THEN 730
720 C = I: GOTO 560
730  HOME : VTAB 22: PRINT "ERROR";: GOSUB 150: GOSUB
     520: GOTO 570
740  HOME : VTAB 22: IF (D - B) * (C - A) > 8000 THEN
      PRINT "SHAPE IS TOO LARGE";: GOSUB 150: GOTO 5
     70
750  PRINT "NEGATIVE SHAPE? ";: GOSUB 170: POKE 239,
     4 * (J$ = "Y"): GOSUB 520: POKE 249,B: POKE 250
     ,D: POKE 237,B
760 A = A + 1:J =  INT (A / 256):I = A - J * 256: POKE
     251,I: POKE 252,J: POKE 235,I: POKE 236,J
770 C = C - 1:J =  INT (C / 256):I = C - J * 256: POKE
     253,I: POKE 254,J
780  HGR2 : POKE 230,96: CALL 62450: POKE 24576,1: POKE
     24578,4: POKE 230,32: CALL 7900: POKE 233,96: SCALE=
     1: ROT= 0
790  HOME : HGR2 : XDRAW 1 AT A,B + 1: GOSUB 160:J =
      PEEK (8103) +  PEEK (8104) * 256 - 24575
800  TEXT : HOME : VTAB 8: PRINT "SAVE SHAPE? ";: GOSUB
     170: IF J$ <  > "Y" THEN 400
810  VTAB 11: CALL BH: PRINT "NAME? ";: GOSUB 170: IF
     I$ = "" THEN 800
820  ONERR  GOTO 860
830  PRINT D$"BSAVE SHAPE."I$",A24576,L"J: HOME
840  VTAB 8: PRINT "NAME: SHAPE."I$: VTAB 11: PRINT
     "ADDRESS SAVED: 24576"
850  VTAB 14: PRINT "LENGTH OF SHAPE: "J" BYTES": VTAB
     17: GOSUB 150: PRINT : GOSUB 130: GOTO 400
860  VTAB 14: GOSUB 140: GOTO 800
870 D$ =  CHR$ (4): IF  PEEK (7900) <  > 169 OR  PEEK
     (8000) <  > 208 THEN  PRINT D$"BLOAD LIFT.OFF.M
     L"
880  POKE 232,0: IF  PEEK (768) <  > 1 OR  PEEK (790
     ) <  > 200 THEN  FOR I = 768 TO 798: READ J: POKE
     I,J: NEXT
890 R$ =  CHR$ (13):B$ =  CHR$ (8):S$ =  CHR$ (32):X
     = 139:Y = 95:BH =  - 958:BL =  - 198
900 KB =  - 16384:KR =  - 16368:SP =  - 16336:FS =  -
     16302:MS =  - 16301:SP =  - 16336
910 M1$ = "REGULAR":M2$ = "LOCKS PLOT":M3$ = "LOCKS
     ERASE":M4$ = "Q=" + M1$ + "   W=" + M2$ + "   E
     =" + M3$: GOTO 400
920  DATA 1,8,4,8,29,8,169,8,133,6,168,169,32,133,7,
     170,177,6,41,127,145,6,200,208,247,230,7,202,20
     8,242,96
```

```
        KEY PERFECT 4.0
            RUN ON
          LIFT.OFF
=============================================
      CODE      LINE# - LINE#
     ------     -------------
      5AF3         20 -  110
      834E        120 -  210
      B45E        220 -  310
      AD04        320 -  410
      9C95        420 -  510
      BC33        520 -  610
      6E9C        620 -  710
      E06F        720 -  810
      E072        820 -  910
      30FC        920 -  920
TOTAL PROGRAM CHECK IS : 0CA8
```

```
CHECK CODE 3.0


ON: LIFT.OFF
TYPE: A

LENGTH: 0B0C
CHECKSUM: FC
```

LISTING 3 — APPLE.CREATOR

```
1  REM   ************************
2  REM   *    APPLE.CREATOR     *
3  REM   *   BY  STEVEN WONG   *
4  REM   * COPYRIGHT (C)  1984 *
5  REM   * BY MICROSPARC, INC. *
6  REM   * LINCOLN, MA.  01773 *
7  REM   ************************
10  REM APPLE.CREATOR
20  HOME : HGR : HCOLOR= 1
30  FOR I = 123 TO 153 STEP 2: READ
     J,K
40  HPLOT I,J TO I,K: NEXT
50  FOR I = 139 TO 145 STEP 2: READ
     J,K
60  HPLOT I,J TO I,K: NEXT
70  DATA 82,87,77,92,75,94,74,95,
     73,96,73,96,74,95,75,94,75,9
     4,75,95
80  DATA 74,96,73,96,73,95,74,93,
     76,91,81,88,68,73,65,70,64,6
     9,62,66
90  VTAB 21: PRINT "** COPYRIGHT
     1984 BY MICROSPARC, INC. **"
```

LISTING 4 — SHAPE.DEMO

```
10  REM   ***********************
11  REM   *     SHAPE.DEMO      *
12  REM   *   BY STEVEN WONG    *
13  REM   * COPYRIGHT (C) 1984 *
14  REM   * BY MICROSPARC, INC *
15  REM   * LINCOLN, MA. 01773 *
16  REM   ***********************
20  HOME : PRINT  CHR$ (4)"BLOAD
     SHAPE.APPLE,A24576"
30  HGR2 : SCALE= 1: ROT= 0: POKE
     232,0: POKE 233,96: GOTO 50
40  XDRAW 1 AT X,110: XDRAW 1 AT
     245 - X,110: RETURN
50  HCOLOR= 7: FOR J = 1 TO 2: POKE
     230,32 * J: CALL 62450
60  XDRAW 1 AT 51,46: XDRAW 1 AT
     100,46: DRAW 1 AT 149,46
70  DRAW 1 AT 198,46:X = 2 + 4 *
     (J = 1): GOSUB 40: NEXT
80 X = 6: FOR I = 1 TO 30:J =  NOT
     J: POKE 230,64 - 32 * J
90  POKE 49236 + J,0:X = X - 4: GOSUB
     40:X = X + 8: GOSUB 40: NEXT

100  FOR I = 1 TO 3000: NEXT : TEXT
```

LISTING 2 — LIFT.OFF.ML

```
:ASM
                        1   ******************************
                        2   *                            *
                        3   *         Lift Off ML        *
                        4   *       By  Steven Wong      *
                        5   *                            *
                        6   *     Copyright (c)  1984    *
                        7   *     By Micro-Sparc, Inc    *
                        8   *     Lincoln, MA.  01773    *
                        9   *                            *
                        10  *     Assembler:   MERLIN    *
                        11  *                            *
                        12  ******************************
                        13
                        14            ORG  $1EDC
                        15
                        16  BUFF      =    $08
                        17  HBASL     =    $26
                        18  HLOC      =    $EB
                        19  VLOC      =    $ED
                        20  TEMP      =    $EE
                        21  HINVFLG   =    $EF
                        22  TOP       =    $F9
                        23  BOTTOM    =    $FA
                        24  LEFT      =    $FB
                        25  RIGHT     =    $FD
                        26  HPOSN     =    $F411
                        27  *-------------------------------
                        28  * Initialize the pointers
                        29  *-------------------------------
1EDC: A9 00             30            LDA  #$00
1EDE: 8D 80 1F          31            STA  PUSH+1
1EE1: 85 08             32            STA  BUFF
1EE3: A9 40             33            LDA  #$40
1EE5: 8D 81 1F          34            STA  PUSH+2
1EE8: 85 09             35            STA  BUFF+1
1EEA: A9 04             36            LDA  #$04
1EEC: 8D A7 1F          37            STA  S2+1
1EEF: A9 60             38            LDA  #$60
1EF1: 8D A8 1F          39            STA  S2+2
1EF4: 20 8B 1F          40            JSR  TEST
                        41  *-------------------------------
                        42  * Scan left to right
                        43  *-------------------------------
1EF7: 20 48 1F          44  GORIGHT  JSR  FIND
1EFA: F0 04             45            BEQ  R3
1EFC: A9 05             46            LDA  #$05
1EFE: D0 02             47            BNE  R4
1F00: A9 01             48  R3        LDA  #$01
1F02: 20 7D 1F          49  R4        JSR  PU1
1F05: 18                50            CLC
1F06: A5 EB             51            LDA  HLOC
1F08: 69 01             52            ADC  #1
1F0A: 85 EB             53            STA  HLOC
1F0C: AA                54            TAX
1F0D: A5 EC             55            LDA  HLOC+1
1F0F: 69 00             56            ADC  #0
1F11: 85 EC             57            STA  HLOC+1
1F13: C5 FE             58            CMP  RIGHT+1
1F15: D0 E0             59            BNE  GORIGHT
1F17: E4 FD             60            CPX  RIGHT
1F19: D0 DC             61            BNE  GORIGHT
1F1B: 20 6C 1F          62            JSR  OVER
                        63  *-------------------------------
                        64  * Scan right to left
                        65  *-------------------------------
1F1E: 20 48 1F          66  GOLEFT   JSR  FIND
1F21: F0 04             67            BEQ  L1
1F23: A9 07             68            LDA  #$07
1F25: D0 02             69            BNE  L2
1F27: A9 03             70  L1        LDA  #$03
1F29: 20 7D 1F          71  L2        JSR  PU1
1F2C: 38                72  L3        SEC
1F2D: A5 EB             73            LDA  HLOC
1F2F: E9 01             74            SBC  #1
1F31: 85 EB             75            STA  HLOC
1F33: AA                76            TAX
1F34: A5 EC             77            LDA  HLOC+1
1F36: E9 00             78            SBC  #0
1F38: 85 EC             79            STA  HLOC+1
1F3A: C5 FC             80            CMP  LEFT+1
1F3C: D0 E0             81            BNE  GOLEFT
1F3E: E4 FB             82            CPX  LEFT
1F40: D0 DC             83            BNE  GOLEFT
1F42: 20 6C 1F          84            JSR  OVER
1F45: 4C F7 1E          85            JMP  GORIGHT
                        86  *-------------------------------
                        87  * Status of pixel at HLOC,VLOC
                        88  *-------------------------------
1F48: A0 00             89  FIND      LDY  #$00
1F4A: A5 EB             90            LDA  HLOC
1F4C: A6 EC             91            LDX  HLOC+1
1F4E: F0 05             92            BEQ  F1
1F50: A0 24             93            LDY  #$24
1F52: 18                94            CLC
1F53: 69 04             95            ADC  #$04
1F55: C9 07             96  F1        CMP  #$07
1F57: 90 05             97            BCC  F2
1F59: E9 07             98            SBC  #$07
1F5B: C8                99            INY
1F5C: D0 F7             100           BNE  F1
1F5E: AA                101 F2        TAX
1F5F: B1 26             102           LDA  (HBASL),Y
1F61: 3D 65 1F          103           AND  MASK,X
1F64: 60                104           RTS
1F65: 01 02 04          105 MASK      HEX  01020408102040
1F68: 08 10 20 40
                        106 *-------------------------------
                        107 * Store 'MOVE DOWN' vector code
                        108 *-------------------------------
1F6C: 20 48 1F          109 OVER      JSR  FIND
1F6F: F0 04             110           BEQ  O1
1F71: A9 06             111           LDA  #$06
1F73: D0 02             112           BNE  O2
1F75: A9 02             113 O1        LDA  #$02
1F77: 20 7D 1F          114 O2        JSR  PU1
1F7A: 4C 8B 1F          115           JMP  TEST
                        116 *-------------------------------
                        117 * Store vector codes
                        118 *-------------------------------
1F7D: 45 EF             119 PU1       EOR  HINVFLG
1F7F: 8D 00 40          120 PUSH      STA  $4000
1F82: EE 80 1F          121           INC  PUSH+1
1F85: D0 03             122           BNE  P1
1F87: EE 81 1F          123           INC  PUSH+2
1F8A: 60                124 P1        RTS
                        125 *-------------------------------
                        126 * Finish?
                        127 *-------------------------------
1F8B: E6 ED             128 TEST      INC  VLOC
1F8D: A5 ED             129           LDA  VLOC
1F8F: C5 FA             130           CMP  BOTTOM
1F91: B0 03             131           BCS  COMPILE
1F93: 4C 11 F4          132           JMP  HPOSN
                        133 *-------------------------------
                        134 * Convert vector codes
                        135 *-------------------------------
1F96: 68                136 COMPILE   PLA
1F97: 68                137           PLA
1F98: A0 00             138           LDY  #$00
1F9A: B1 08             139 S1        LDA  (BUFF),Y
1F9C: 85 EE             140           STA  TEMP
1F9E: C8                141           INY
1F9F: B1 08             142           LDA  (BUFF),Y
1FA1: 0A                143           ASL
1FA2: 0A                144           ASL
1FA3: 0A                145           ASL
1FA4: 65 EE             146           ADC  TEMP
                        147 *-------------------------------
                        148 * Store vector shape table data
                        149 *-------------------------------
1FA6: 8D 00 60          150 S2        STA  $6000
1FA9: F0 0F             151           BEQ  FINISH
1FAB: EE A7 1F          152           INC  S2+1
1FAE: D0 03             153           BNE  S3
1FB0: EE A8 1F          154           INC  S2+2
1FB3: C8                155 S3        INY
1FB4: D0 E4             156           BNE  S1
1FB6: E6 09             157           INC  BUFF+1
1FB8: D0 E0             158           BNE  S1
1FBA: 60                159 FINISH    RTS


--End assembly--

223 bytes

Errors: 0
```