SHADOW PLAY

This Hi-Res utility clears to the current HCOLOR below any line you can draw with the HPLOT command.

Drawing three-dimensional graphics causes some problems for the programmer. It's important that lines that are supposed to be hidden stay that way. The standard approach is called the painter's algorithm: you start with the background, then draw successively closer objects. When you're making a 3-D contour drawing, the method is the same. You just start at the back of the scene and take vertical slices perpendicular to the line of view. With each slice, the contour is drawn and the entire screen beneath the line is erased. It's the erasing process that causes the problem, since the most obvious method is slow — even in machine language. Here's a fast algorithm that figures out when entire bytes, as opposed to individual bits, can be cleared.

USING SHADOW PLAY

After you install the machine language program (**Listing 1**), the utility is available with the same syntax as the HPLOT command, except that it's preceded with an ampersand (&). For instance:

& HPLOT 0,50 TO 100,65 TO 180,40

would change all the pixels below this *contour line* to the current HCOLOR. The practical way to use it is to start with the contour line, using a normal HPLOT command. Then change the HCOLOR to whatever you want to clear with (usually black), and follow up with an & HPLOT command using using the same points as the original HPLOT command.

Listing 2 is a simple demonstration of SHADOW.PLAY. It creates a simple scene with two contours for the hills.

ENTERING THE PROGRAMS

If you have an assembler, enter the source code from **Listing 1**. Use the name SHADOW.PLAY for the object file. If you don't have an assembler, enter the Monitor with CALL −151 and key in the hex code. Save the resulting program with the command:

BSAVE SHADOW.PLAY,A$7000,L$195

Then enter the Applesoft program from **Listing 2** and save it with the command:

SAVE SHADOW.DEMO

For help with entering *Nibble* listings, see the Typing Tips section.

HOW IT WORKS

Remember that in Applesoft Hi-Res graphics, the horizontal X-coordinate may range from 0 to 279, the vertical Y-coordinate may range from 0 to 191, and the point (0,0) is defined as the upper-leftmost point of the screen. This means that increasing the Y-coordinate corresponds to moving down the screen.

Let's first define a few terms and briefly review the arrangement of the Hi-Res screen. The points that lie on a particular *horizon line* are *horizon points*, and the points that lie below the horizon points are *shadow points*. Your Hi-Res display is determined by the values of the 8,192 (8K) bytes in a section of memory called *Hi-Res screen memory*, which begins at decimal 8192 (hex 2000) for page 1. Since the Hi-Res screen is a grid with 280 x 192 = 53,760 intersections, it's obvious that the points are not controlled on a byte-by-byte basis. Each horizontal line on the screen is

FIGURE 1: Order of Pixel Clearing

	Byte 1							Byte 2							Byte 3							
screen top ↑																		J	K	L		
													G	H	I	16	21	26			
									D	E	F	10	12	14		17	22	27			
				A	B	C	7	8	9	11	13	15				18	23	28			
				1	3	5	5	5	5	5	5	5				19	24	29			
screen bottom ↓				2	4	6	6	6	6	6	6	6				20	25	30			

instead controlled by the values of 40 contiguous bytes. In each byte, seven bits correspond to seven bits on the screen, and the remaining bit has a color-shifting effect. In any byte, the lowest plotting bit controls the rightmost of the points controlled by that byte.

As you may have noticed, the number 280 (the number of horizontal points on a line) is divisible by 7 (the number of plotting bits in a byte). We may therefore think of the screen as divided into 40 columns, and of each screen horizontal line as divided into 40 sections corresponding to the controlling bytes. When all the points controlled by a particular byte are shadow points, we'll call that byte a *shadow byte*.

Contrary to what you might expect, bytes 41-80 do not control the second horizontal line on the screen, but rather the ninth. The complex mapping of the screen memory into the hex grid is shown in the *Applesoft BASIC Reference Manual*. Because your Applesoft interpreter is so complex, it uses two sets of coordinates. One set, which we'll call the external cursor, is stored in $E0-$E2. The X-coordinate, since it can exceed 255, must

In each byte, seven bits correspond to seven bits on the screen, and the remaining bit has a color-shifting effect.

be represented by two bytes ($E0,$E1), and the Y-coordinate is represented by one byte ($E2). The second set is the internal cursor. This consists primarily of a two-byte pointer ($26,$27) to the byte controlling the leftmost section of the proper row, and an offset value ($E5) indicating which of the 40 bytes must be examined. Then there is a bit position indicator at $30 to indicate which bit in that byte is to be considered.

The Simplest Way

One simple algorithm for clearing the screen beneath a horizon line is as follows:

1. Calculate the next horizon point.
2. Save the value of the left-edge pointer ($26,$27).
3. Use the INCRY subroutine ($F504) to get the memory location for each shadow point below the calculated horizon point, and turn off the indicated pixels. (When INCRY is called, it resets the internal cursor to correspond to incrementing the vertical screen coordinate by one — a higher Y-coordinate means moving downscreen.)
4. When you reach the floor of the screen (Y = 191, or, equivalently, when the pointer $26,$27 is set to $23D0), restore the value of the left-edge pointer to the same row as the horizon point.
5. If you have just processed the last horizon point, stop. If not, go back to step 1.

Calculating the horizon points is easy, since the Applesoft HLINE routine already knows how to calculate the points on a line. We want to modify that routine so that after calculating each horizon point, it clears the shadow points below it, as described in steps 2-4 above.

When this algorithm is used to clear the points below the line 0,0 to 279,0 (virtually the entire screen), the process takes just under four seconds. This is much slower than the HGR routine, which turns the entire screen black in under a second. Let's examine why.

Here's a fast algorithm that figures out when entire bytes, as opposed to individual bits, can be cleared.

The answer is that the HGR routine turns off the pixels on the screen one byte (seven pixels) at a time, while this particular algorithm works strictly on a pixel-by-pixel basis. A better algorithm would determine in advance when all the pixels in a byte were to be shadowed and, if so, shadow them all at once, rather than separately.

A Faster Way

Now, here's a more sophisticated algorithm (Note: It assumes a horizon line similar to that shown in Figure 1):

1. Set FLOOR (a zero-page pointer used by the algorithm and not by Applesoft) to the bottom row of the screen ($23D0).
2. Get the Y-offset (column 0-39) for the right end of the horizon line, and store it in another (otherwise unused) zero page location, which we'll label LASTE5.
3. Calculate the next horizon point, and save the left edge address stored in the $26,$27 pointer as part of the internal cursor.
4. If the horizon point is in the same column as the right endpoint (condition A), or if it is the leftmost bit controlled by a screen byte (condition B), then set FLOOR to $23D0.
5. If condition B is met, then use the INCRY subroutine to get the location for each shadow BYTE below the horizon point and not below the FLOOR, and change the value of that byte to that of the shadow color.
6. Get the address value saved in LASTE5 and put it back into the left-edge pointer ($26,$27).
7. If condition B is met, then set FLOOR to the address saved in LASTE5; that is, to the address of the left edge of the row of the current horizon point.
8. Stop if the end of the line has been reached. If not, go to step 3.

Figure 1 illustrates the workings of this improved algorithm in detail. Circled letters A - P are the horizon points. The numbered points below them are shadow points. The points are divided into three groups, labeled *first byte*, *second byte*, and *third byte*. These are the seven-pixel-wide columns described at the beginning of this section.

The algorithm begins with horizon point A. The FLOOR is set to $23D0. Neither condition A nor B is met, so proceed bit by bit, clearing pixels 1 and 2. Then move on to horizon point B, and similarly clear pixels 3 and 4. When you reach horizon point C you'll find that it's the leftmost bit controlled by a screen byte, which is condition B. FLOOR is set to $23D0. Then all the shadow points marked 5 are cleared at once, since they're in the same shadow byte. Similarly, the pixels marked 6 are cleared.

Now we restore the left edge pointer to horizon point D and, in accordance with step 8 of the algorithm, store the value of that pointer in FLOOR. FLOOR now corresponds to the row Y = 188.

As we proceed from horizon point E to horizon point I, in accordance with step 6, we clear pixels 8-15. Coming to horizon point J, you'll find that it's in the same seven-pixel-wide column as endpoint L. Thus, condition A is met. The value $23D0 (Y-191) is placed in FLOOR and you'll clear, bit by bit, pixels 16-20 under J, 21-25 under K, and 26-30 under L.

If the line were sloped the other way (running from top left to bottom right), you would change the algorithm somewhat:

1. Start first from the bottom right and move toward the top left.
2. Condition B would be that the horizon point was the rightmost bit of a screen byte.
3. The Y-offset (column number) of the left end of the line would be placed in LASTE5. In order to traverse the horizon line in the correct direction, it may be necessary to temporarily switch the endpoints from the order in which they were given.

The first part of SHADOW parses the command, which is in the form:

```
&HPLOT A,B TO C,D
```

and sets the graphics cursors and Apple registers so that the line will run in the proper direction. The second part calculates the horizon points and shadows them point-by-point or byte-by-byte, as appropriate.

LISTING 1: SHADOW.PLAY

```
                        1   *
                        2   * SHADOW.PLAY
                        3   * BY IVER COOPER
                        4   * COPYRIGHT (C) 1987
                        5   * BY MICROSPARC, INC.
                        6   * CONCORD, MA  01742
                        7   *
                        8   * MERLIN ASSEMBLER
                        9   *
                        10  INTX      EQU   $F465
                        11  INTY      EQU   $F4D3
                        12  RTS1      EQU   $F600
                        13  HOLD      EQU   $FF
                        14  COUNT     EQU   $FE
                        15  HOLD26    EQU   $FC
                        16  HOLD27    EQU   $FD
                        17  INCRY     EQU   $F504
                        18  HFNS      EQU   $F6B9
                        19  SAVE      EQU   $FF4A
                        20  ACC       EQU   $45
                        21  XREG      EQU   $46
                        22  YREG      EQU   $47
                        23  CHRGET    EQU   $B1
                        24  FLOOR     EQU   $FA
                        25  DIR       EQU   $F9
                        26  LASTE5    EQU   $1F
                        27  HPOSN     EQU   $F411
                        28  FLAG      EQU   $1E
                        29  HPLOT     EQU   $F457
                        30  CHRGOT    EQU   $B7
                        31  *
                        32  * SET UP AMPERSAND ENTRY WITH
                        33  * JMP $7000 AT $3F5
                        34  *
                        35  *
                        36  * GET DESTINATION
                        37  *
                        38          ORG   $7000
                        39
7000: C9 93             40          CMP   #$93        "HPLOT"
7002: F0 0E             41          BEQ   PLOT
7004: C9 C1             42  PARSE   CMP   #$C1        "TO"
7006: D0 09             43          BNE   OUT
7008: 20 21 70          44          JSR   DESGET
700B: 20 B7 00          45          JSR   CHRGOT
700E: 4C 04 70          46          JMP   PARSE
7011: 60                47  OUT     RTS
7012: 20 B1 00          48  PLOT    JSR   CHRGET
7015: 20 B9 F6          49          JSR   HFNS
7018: 20 57 F4          50          JSR   HPLOT
701B: 20 B7 00          51          JSR   CHRGOT
701E: 4C 04 70          52          JMP   PARSE
7021: A9 D0             53  DESGET  LDA   #$D0
7023: 85 FA             54          STA   FLOOR
7025: A9 23             55          LDA   #$23
7027: 85 FB             56          STA   FLOOR+1
7029: A9 00             57          LDA   #0
702B: 85 1E             58          STA   FLAG
702D: A5 E0             59          LDA   $E0
702F: 48                60          PHA
7030: A5 E1             61          LDA   $E1
7032: 48                62          PHA
7033: A5 E2             63          LDA   $E2
7035: 48                64          PHA
7036: 48                65          PHA
7037: A5 E5             66          LDA   $E5
7039: 85 FF             67          STA   HOLD
703B: 48                68          PHA
703C: 20 B1 00          69          JSR   CHRGET
703F: 20 B9 F6          70          JSR   HFNS
7042: 20 4A FF          71          JSR   SAVE
7045: A5 45             72          LDA   ACC
7047: A6 46             73          LDX   XREG
7049: 20 11 F4          74          JSR   HPOSN
704C: A5 E5             75          LDA   $E5
704E: 85 1F             76          STA   LASTE5
7050: 68                77          PLA
7051: C5 1F             78          CMP   LASTE5
7053: F0 06             79          BEQ   GET2
7055: 90 04             80          BCC   GET2
7057: A9 80             81          LDA   #$80
7059: 85 1E             82          STA   FLAG
705B: 68                83  GET2    PLA
705C: C5 E2             84          CMP   $E2
705E: 90 08             85          BCC   GET3
7060: F0 06             86          BEQ   GET3
7062: A5 1E             87          LDA   FLAG
7064: 49 FF             88          EOR   #$FF
7066: 85 1E             89          STA   FLAG
7068: A9 81             90  GET3    LDA   #$81
706A: 85 F9             91          STA   DIR
706C: A5 1E             92          LDA   FLAG
706E: 29 80             93          AND   #$80
7070: D0 04             94          BNE   GET4
7072: A9 C0             95          LDA   #$C0
7074: 85 F9             96          STA   DIR
7076: A5 1E             97  GET4    LDA   FLAG
7078: 29 01             98          AND   #$1
707A: F0 17             99          BEQ   HLINB
707C: D0 02             100         BNE   HLINA
707E: 90 13             101         BCC   HLINB
7080: 68                102 HLINA   PLA
7081: 85 E2             103         STA   $E2
7083: 68                104         PLA
7084: 85 E1             105         STA   $E1
7086: A8                106         TAY
7087: 68                107         PLA
7088: 85 E0             108         STA   $E0
708A: AA                109         TAX
708B: A5 E2             110         LDA   $E2
708D: 20 11 F4          111         JSR   HPOSN
7090: 4C 9F 70          112         JMP   HLINC
7093: 68                113 HLINB   PLA
7094: A8                114         TAY
7095: 68                115         PLA
7096: AA                116         TAX
7097: A5 FF             117         LDA   HOLD
7099: 85 1F             118         STA   LASTE5
709B: 68                119         PLA
709C: 4C A5 70          120         JMP   HLIND
709F: A4 45             121 HLINC   LDY   ACC
70A1: A6 47             122         LDX   YREG
70A3: A5 46             123         LDA   XREG
                        124 *
                        125 * THIS IS COPY OF HLINE ROUTINE
                        126 * IN ROM AT $F53A-F58C
                        127 *
70A5: 48                128 HLIND   PHA
70A6: 38                129         SEC
70A7: E5 E0             130         SBC   $E0
70A9: 48                131         PHA
70AA: 8A                132         TXA
70AB: E5 E1             133         SBC   $E1
70AD: 85 D3             134         STA   $D3
70AF: B0 0A             135         BCS   HLIN2
70B1: 68                136         PLA
70B2: 49 FF             137         EOR   #$FF
70B4: 69 01             138         ADC   #$01
70B6: 48                139         PHA
70B7: A9 00             140         LDA   #$00
70B9: E5 D3             141         SBC   $D3
70BB: 85 D1             142 HLIN2   STA   $D1
70BD: 85 D5             143         STA   $D5
70BF: 68                144         PLA
70C0: 85 D0             145         STA   $D0
70C2: 85 D4             146         STA   $D4
70C4: 68                147         PLA
70C5: 85 E0             148         STA   $E0
70C7: 86 E1             149         STX   $E1
70C9: 98                150         TYA
70CA: 18                151         CLC
70CB: E5 E2             152         SBC   $E2
70CD: 90 04             153         BCC   HLIN3
70CF: 49 FF             154         EOR   #$FF
70D1: 69 FE             155         ADC   #$FE
70D3: 85 D2             156 HLIN3   STA   $D2
70D5: 84 E2             157         STY   $E2
70D7: 66 D3             158         ROR   $D3
70D9: 38                159         SEC
70DA: E5 D0             160         SBC   $D0
70DC: AA                161         TAX
70DD: A9 FF             162         LDA   #$FF
70DF: E5 D1             163         SBC   $D1
70E1: 85 1D             164         STA   $1D
70E3: A4 E5             165         LDY   $E5
70E5: B0 05             166         BCS   HLIN5
70E7: 0A                167 HLIN4   ASL
70E8: 20 65 F4          168         JSR   INTX
70EB: 38                169         SEC
70EC: A5 D4             170 HLIN5   LDA   $D4
70EE: 65 D2             171         ADC   $D2
70F0: 85 D4             172         STA   $D4
70F2: A5 D5             173         LDA   $D5
70F4: E9 00             174         SBC   #$00
70F6: 85 D5             175 HLIN6   STA   $D5
                        176 *
                        177 * HERE IS SUBSTITUTED, IN PLACE OF ROM $F580-$F596
                        178 * THE ROUTINE DESCRIBED IN THE ARTICLE
                        179 *IN THE ARTICLE "NEW HORIZONS"
                        180 *
                        181 *
70F8: 08                182 SHADE1  PHP
70F9: 86 FF             183         STX   HOLD
70FB: A5 26             184         LDA   $26
70FD: 85 FC             185         STA   HOLD26
70FF: A5 27             186         LDA   $27
7101: 85 FD             187         STA   HOLD27
7103: A5 30             188         LDA   $30
7105: C5 F9             189         CMP   DIR
7107: F0 06             190         BEQ   SHADEX
7109: A5 E5             191         LDA   $E5
710B: C5 1F             192         CMP   LASTE5
710D: D0 08             193         BNE   SHADE2
710F: A9 D0             194 SHADEX  LDA   #$D0
7111: 85 FA             195         STA   FLOOR
7113: A9 23             196         LDA   #$23
7115: 85 FB             197         STA   FLOOR+1
7117: A5 26             198 SHADE2  LDA   $26
7119: C5 FA             199         CMP   FLOOR
711B: D0 09             200         BNE   SHADEY
711D: A5 27             201         LDA   $27
711F: C5 FB             202         CMP   FLOOR+1
7121: D0 03             203         BNE   SHADEY
7123: 4C 4D 71          204         JMP   SHADE3
7126: 20 04 F5          205 SHADEY  JSR   INCRY
```

```
7129: A5 30     206           LDA   $30
712B: C5 F9     207           CMP   DIR
712D: D0 0F     208           BNE   SHADEP
712F: A5 E5     209           LDA   $E5
7131: C5 1F     210           CMP   LASTE5
7133: F0 09     211           BEQ   SHADEP
7135: A5 1C     212 LFTBIT    LDA   $1C
7137: A4 E5     213           LDY   $E5
7139: 91 26     214           STA   ($26),Y
713B: 4C 17 71  215           JMP   SHADE2
713E: A5 1C     216 SHADEP    LDA   $1C
7140: A4 E5     217           LDY   $E5
7142: 51 26     218           EOR   ($26),Y
7144: 25 30     219           AND   $30
7146: 51 26     220           EOR   ($26),Y
7148: 91 26     221           STA   ($26),Y
714A: 4C 17 71  222           JMP   SHADE2
714D: A5 FC     223 SHADE3    LDA   HOLD26
714F: 85 26     224           STA   $26
7151: A5 FD     225           LDA   HOLD27
7153: 85 27     226           STA   $27
7155: A5 30     227           LDA   $30
7157: C5 F9     228           CMP   DIR
7159: D0 08     229           BNE   SHADE4
715B: A5 FC     230           LDA   HOLD26
715D: 85 FA     231           STA   FLOOR
715F: A5 FD     232           LDA   HOLD27
7161: 85 FB     233           STA   FLOOR+1
7163: A6 FF     234 SHADE4    LDX   HOLD
7165: 28        235           PLP
                236           *
                237           * HERE IS A COPY OF ROM $F597-F5B1
                238           * THE REMAINDER OF THE HLIN ROUTINE
                239           *
7166: E8        240 HLIN7     INX
7167: D0 14     241           BNE   HLIN8
7169: E6 1D     242           INC   $1D
716B: D0 10     243           BNE   HLIN8
716D: A5 1E     244           LDA   FLAG
716F: 29 01     245           AND   #$1
7171: D0 09     246           BNE   EXIT
7173: A5 45     247           LDA   ACC
7175: A4 47     248           LDY   YREG
7177: A6 46     249           LDX   XREG
7179: 20 11 F4  250           JSR   HPOSN
717C: 60        251 EXIT      RTS
717D: A5 D3     252 HLIN8     LDA   $D3
717F: 90 03     253           BCC   HLIN9
7181: 4C E7 70  254           JMP   HLIN4
7184: 20 D3 F4  255 HLIN9     JSR   INTY
7187: 18        256           CLC
7188: A5 D4     257           LDA   $D4
718A: 65 D0     258           ADC   $D0
718C: 85 D4     259           STA   $D4
718E: A5 D5     260           LDA   $D5
7190: 65 D1     261           ADC   $D1
7192: 4C F6 70  262           JMP   HLIN6

--End assembly, 405 bytes, Errors: 0
```

END OF LISTING 1

```
KEY PERFECT 5.0
RUN ON
SHADOW.PLAY
```

CODE-5.0	ADDR# - ADDR#	CODE-4.0
F939A915	7000 - 704F	29D7
14BB7206	7050 - 709F	2358
A344E93F	70A0 - 70EF	2A30
B1B99948	70F0 - 713F	2E48
A2F6F02D	7140 - 718F	2744
806D849D	7190 - 7194	027C
B56F06C1 = PROGRAM TOTAL =		0195

LISTING 2: SHADOW.DEMO

```
10  REM  ************************
20  REM  * SHADOW.DEMO          *
30  REM  * BY IVER COOPER       *
40  REM  * COPYRIGHT (C) 1987 *
50  REM  * BY MICROSPARC, INC *
60  REM  * CONCORD, MA  01742 *
70  REM  ************************
80  TEXT : HOME : VTAB 10: PRINT "SHADOW.DEMO
    BY IVER COOPER": PRINT "COPYRIGHT 1987
    BY MICROSPARC, INC.": ONERR  GOTO 340
90  PRINT  CHR$ (4);"BLOAD SHADOW.PLAY"
100  POKE 1013,76: POKE 1014,0: POKE 1015,112
110  VTAB 21: PRINT "PRESS RETURN TO CONTINUE
     ";: GET A$: PRINT : HOME
120  REM  DEMO
130  HGR
140  HCOLOR= 6: REM  COLOR BLUE
150  REM  PLOT POINT AND CALL BKGD COLOR ROUT
     INE
160  HPLOT 0,0: CALL 62454
170  REM  SET COLOR TO WHITE2 AND DRAW FIRST
     HORIZON. WHITE2 MUST BE USED WHEN DRAWIN
     G OVER A BLUE (HI BIT SET) BKGD
180  HCOLOR= 7: HPLOT 0,120 TO 50,80 TO 100,1
     00 TO 150,40 TO 200,70 TO 279,100
190  REM  SET COLOR TO ORANGE AND RETRACEFIRS
     T HORIZON WITH &HPLOT, THUS CREATING AN
     ORANGE MOUNTAIN.
200  HCOLOR= 5: &  HPLOT 0,120 TO 50,80 TO 10
     0,100 TO 150,40 TO 200,70 TO 279,100
210  REM  SET COLOR TO WHITE2 AND DRAW SECOND
      (NEARER) MOUNTAIN'S PROFILE
220  HCOLOR= 7: HPLOT 0,80 TO 35,100 TO 70,11
     0 TO 105,60 TO 140,30 TO 175,20 TO 210,2
     0 TO 245,80 TO 279,100
230  REM  SET COLOR TO ORANGE AND RETRACE WIT
     H &HPLOT
240  HCOLOR= 5: &  HPLOT 0,80 TO 35,100 TO 70
     ,110 TO 105,60 TO 140,30 TO 175,20 TO 21
     0,20 TO 245,80 TO 279,100
250  REM  SET COLOR TO GREEN AND &HPLOT AHORI
     ZONTAL LINE. SCREEN. THIS TIME WE DON'T
     BOTHER TO DRAW A WHITE BORDER.
260  HCOLOR= 1: &  HPLOT 0,140 TO 279,140
270  REM  SET COLOR TO BLACK2 AND &HPLOT ROOF
     . THEN HPLOT ROOF WITH OVERHANG. SET COL
     OR TO BLACK1 AND &HPLOT PART OF HOUSE BL
     OCKING GREEN (HI BIT CLR).
280  HCOLOR= 4: &  HPLOT 140,120 TO 160,110 TO
     180,120: HPLOT 136,122 TO 160,110 TO 184
     ,122: HCOLOR= 0: &  HPLOT 140,140 TO 180
     ,140
290  REM  NOW DRAW A FLAG FOR FUN
300  HCOLOR= 4: HPLOT 200,20 TO 200,2 TO 196,
     2 TO 196,3 TO 200,3 TO 200,4 TO 196,4 TO
     196,5 TO 200,5
310  REM  AND A DOOR IN WHITE1, USING &HPLOT
320  HCOLOR= 3: &  HPLOT 155,142 TO 165,142
330  HOME : VTAB 21: PRINT "RETURN TO REPEAT,
      ESCAPE TO QUIT";: GET Z$: PRINT : ON Z$
     <  >  CHR$ (27) GOTO 130: TEXT : HOME :
     GOTO 410
340 E =  PEEK (222):EL =  PEEK (218) + 256 *
     PEEK (219): CALL  - 3288: POKE 216,0
350  TEXT : HOME : VTAB 12
360  IF E = 8 THEN  PRINT "I/O ERROR--CHECK D
     RIVE DOOR": GOTO 390
370  IF E = 6 THEN  PRINT "SHADOW.PLAY NOT ON
      THIS DISK"
380  IF E <  > 6 AND E <  > 8 GOTO 400
390  PRINT "RETURN TO TRY AGAIN, ESCAPE TO QU
     IT";: GET Z$: PRINT : ON Z$ =  CHR$ (27)
      GOTO 410: GOTO 80
400  PRINT "ERROR "E" IN LINE "EL
410  END
```

END OF LISTING 2

```
KEY PERFECT 5.0
RUN ON
SHADOW.DEMO
```

CODE-5.0	LINE# - LINE#	CODE-4.0
223C8748	10 - 100	8A79
A1EB3277	110 - 200	C72C
F98FBE1D	210 - 300	013773
C0EF7DEB	310 - 400	B1AC
AAA99904	410 - 410	D7
216FA15A = PROGRAM TOTAL =		06A5