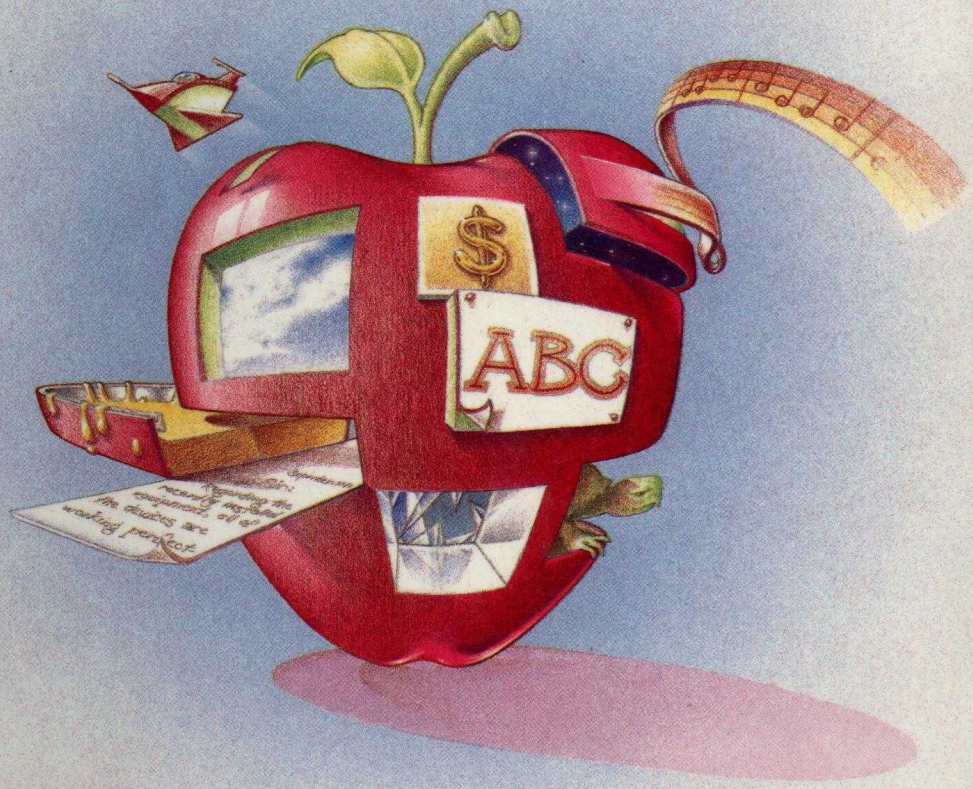
# COMPUTE!'s First Book of Apple




Challenging games, exciting educational tools, useful home applications, and sophisticated graphics and sound programs for Apple II, II+, IIe, and IIc computers.

A **COMPUTE! Books** Publication $12.95

# COMPUTE!'s First Book of Apple

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

The following articles were originally published in *COMPUTE!* magazine, copyright 1981, Small System Services, Inc.: "Commas, Colons, and Quote Marks Too" (May); "Oscilloscope" (July); "Undeletable Lines" (October).

The following articles were originally published in *COMPUTE!* magazine, copyright 1982, Small System Services, Inc.: "IRA Planner" (July); "Chemistry Lab" (August).

The following articles were originally published in *COMPUTE!* magazine, copyright 1983, Small System Services, Inc.: "Home Energy Calculator" (January); "Custom Catalog" (March); "Letter and Number Play" (March) "Typing Teacher" (April); "Apple Fast Sort" (May); "Crosswords" (May); "One on One" (May); "The Apple Hi Res Painter" (May); "Apple Shape Generator (June); "Astrostorm" (June); "Memory Trainer" (June).

The following articles were originally published in *COMPUTE!* magazine, copyright 1983, COMPUTE! Publications, Inc.: "First Math" (August); "Weather Forecaster" (August); "Caves of Ice" (September); "Apple Sounds—from Beeps to Music, Part 1" (October); "Spiralizer" (October); "Apple Sounds—from Beeps to Music, Part 2" (November); "Calorie Cop" (December); "Utility Bill Audit" (December).

The following articles were originally published in *COMPUTE!* magazine, copyright 1984, COMPUTE! Publications, Inc.: "Quatrainment" (February); "Barrier Battle" (March); "Roader" (March); "Snertle" (May); "3-D Plotting" (May); "Apple Input and Menu Screens" (June); "Devastator" (August); "ML Tracer" (August); "Apple Editing Hints" (September); "Canyon Runner" (October); "All About the Status Register" (October/November).



Printed in the United States of America

ISBN 0-942386-69-8

10 9 8 7 6 5 4

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. Apple is a trademark of Apple Computer, Inc.

# Contents

# Foreword

Since its introduction in the late 1970s, the Apple II computer series has become the mainstay of tens of thousands of home computer users. From games and simulations to educational, financial, and programming applications, users of the II, the II+, the IIe, and the IIc have delighted in the power of Apple computers.

Now, with *COMPUTE!'s First Book of Apple*, these adaptable computers become more versatile than ever before.

There's something here for almost every Apple user. Do you like exciting games? Try "Caves of Ice," and escape from an icy cavern before the chills set in. Or take to the highway in "Roader," a game in which you maneuver a high-performance racer down a winding, hazard-strewn highway. "Devastator" puts the Earth's fate in your hands (and a 3-D scrolling spacescape on your screen), while "Barrier Battle" brings new meaning to the phrase "Don't fence me in."

Perhaps you use your Apple in education. "Letter and Number Play" introduces preschoolers to numbers and to the alphabet. "First Math" and "Snertle" make entertaining games out of elementary math. "Chemistry Lab" adds visual excitement to simple chemistry experiments, while "Typing Teacher" and "Memory Trainer" are valuable to students of all ages.

Your Apple can handle many practical assignments too. "Weather Forecaster" turns your computer into an impressive meteorological predictor. "Home Energy Calculator" and "Utility Bill Audit" help you monitor heating and cooling costs.

There's even more. Programming aids like "Custom Catalog" and "Undeletable Lines" let you customize your programs with that professional touch. The graphics techniques illustrated by "Spiralizer" and "Oscilloscope" show you what your Apple's graphics can do. There's even a comprehensive sound editor, "Apple Sounds," that can be used to create custom sound effects or to compose and play musical tunes.

Whether you've had an Apple for years or for just a few months, you'll find that *COMPUTE!'s First Book of Apple* has plenty to offer. It's a book that helps you achieve more with your Apple than you might have thought possible.

# Chapter 1

# Games

# Introduction

Every Apple owner knows that the Apple is a real workhorse, but the programs in this chapter will show you its more playful side.

If arcade games are your favorite, you can play tag with hurtling asteroids in Peter Lear's "Astrostorm." Another fast-paced arcade adventure—"Roader," by Brian Foley—puts you at the wheel of a high-performance race car. Just watch out for the curves.

Then there's "Devastator," by David R. Arnold. It puts the fate of the Earth in your hands—and a 3-D scrolling spacescape on your screen.

For a change of pace, try "Caves of Ice" by Martin Bunker and Robert Tsuk. You're imprisoned inside an immense ice cavern, and there's only one way out. But where could it be? With its remarkable 3-D maze graphics, this one is sure to keep your interest from one chilling episode to the next.

After escaping from the ice cave, you can limber up your frozen muscles with Heath Lawrence's "Barrier Battle." Who would have thought that fences could be so much fun? Equally challenging is Chris York's "One on One," a game that may be the ultimate variation on *Pong*.

Your Apple can play intellectual games, too. "Quatrainment," by Sean Puckett, innocently challenges you to match a simple geometric pattern. It's hard to beat, in more ways than one. And Tim Smith's "Mind Reader" will have you wondering whether your Apple is even smarter than you thought. Load it up and run it the next time someone says that computers are nothing but dumb machines.

# Astrostorm

Peter Lear
Apple Translation by Karen McCollough

*Try to guide your spaceship, carrying emergency medical supplies, through a dangerous asteroid storm. But be careful. Success may depend on your ability to make split-second decisions.*

You are Captain Bosdiger of the interstellar tug *The Viccard.* While orbiting the fifth planet in the Benard system, you receive a distress call from the Solarian system. They are in desperate need of vital medical supplies, and you are the only one who can deliver them in time.

But your calculations indicate that you'll have to cross *six* different asteroid fields to reach the planet in safety. Can you do it?

## Looking at Astrostorm

There is no time limit. The game loop (lines 120–250) will execute until a crash is detected in line 230.

Asteroids scroll vertically, from the bottom to the top of the screen, but only horizontal movement of the spaceship is allowed. Direction is controlled with the left and right arrow keys. Movement is initiated by pressing the space bar; every time you change direction, your ship will stop until you press the space bar again.

There are several skill levels, and you can make the game more difficult by specifying a higher level. That will place the spaceship closer to the bottom of the screen and require a quicker reaction to avoid collisions.

Scoring is based on the level of difficulty, on how far you have traveled, and on the direction in which you are moving. More points are given on the higher levels. In addition, points are awarded for movement of the spaceship to the right and deducted for movement to the left.

## Astrostorm

```
20  FOR I = 770 TO 795: READ M: POKE I,M: NEXT I
25  DATA  172,01,03,174,01,03,169,04,32,168,252,17
    3,48,192,232,208,253,136,208,239,206,0,03,208
    ,231,96
30  GOTO 1000
```

```
100  TEXT : HOME : VTAB 1: PRINT "SCORE: ": POKE 3
     4,2
110  FOR I = 1 TO 15:SP =  INT ( RND (1) * 39) + 2
     : VTAB 24: HTAB (SP): PRINT "*": NEXT I
115  FOR I = 10 TO 50 STEP 40: POKE 768,10: POKE 7
     69,I: CALL 770: NEXT I
120  POKE OP,160: VTAB 24:SP =  INT ( RND (1) * 39
     ) + 2
130  HTAB (SP): PRINT "*"
140  IF  PEEK (CP) = 170 THEN  GOTO 800
145  POKE CP,CC:OP = CP
146  FOR I = 1 TO 25: NEXT
149  VTAB 1: HTAB 7: CALL  - 868: VTAB 1: HTAB 7: PRINT
     PT
170 M =  PEEK ( - 16384): ON M < 128 GOTO 120
180  IF M - 128 = 8 THEN CC = 188:MV =  - 1: GOTO
     120
190  IF M - 128 = 21 THEN CC = 190:MV =  + 1: GOTO
     120
200  IF M - 128 <  > 32 THEN  GOTO 120
210  IF CP <  > BP OR MV > 0 THEN  GOTO 215
212  ON SF GOTO 120
213 CP = BP + 39:SF = SF - 1: GOTO 230
215 CP = CP + MV
220  IF CP <  > BP + 39 THEN  GOTO 230
222  ON SF = 6 GOTO 600
225 CP = BP:SF = SF + 1
230  IF  PEEK (CP) = 170 THEN  GOTO 800
240  IF  NOT MV THEN PT = PT +  INT ((MV * ( INT (
     DL * .2) * ((CP - BP) * SF))) / 2): GOTO 250
245 PT = PT + (MV * ( INT (DL * .3) * ((CP - BP) *
     SF)))
250  GOTO 120
600  REM  WIN
610  TEXT : HOME : VTAB 2: HTAB 15: PRINT "SCORE:
     ";PT
620  VTAB 8: HTAB 13: FLASH : PRINT "CONGRATULATIO
     NS": NORMAL : VTAB 12: HTAB 14: PRINT "YOU MA
     DE IT!"
640  POKE 768,15: POKE 769,50: CALL 770: POKE 768,
     10: POKE 769,10: CALL 770: POKE 768,15: POKE
     769,50: CALL 770
650  FOR I = 1 TO 500: NEXT I
660  POKE  - 16368,0: GOTO 910
800  REM  CRASH
810 V = DL + 2:H = CP - BP
820  IF H <  = 1 THEN  VTAB V - 1: HTAB H: PRINT "
      /": VTAB V: HTAB H: PRINT " -": VTAB V + 1: HTAB
     H: PRINT " "; CHR$ (92): GOTO 850
```

```
830  IF H >  = 39 THEN  VTAB V - 1: HTAB H: PRINT
      CHR$ (92);" ": VTAB V: HTAB H: PRINT "- ": VTAB
     V + 1: HTAB H: PRINT "/ ": GOTO 850
840  VTAB V - 1: HTAB H: PRINT  CHR$ (92);" /": VTAB
     V: HTAB H: PRINT "- -";: VTAB V + 1: HTAB H: PRINT
     "/ "; CHR$ (92);
850  FOR I = 1 TO 200:X =  PEEK ( - 16336): NEXT I

890  POKE  - 16368,0
900  TEXT : HOME : VTAB 2: HTAB 15: PRINT "SCORE:
     ";PT
905  VTAB 10: HTAB 11: PRINT "YOU LOST YOUR SHIP!"

910  VTAB 21: HTAB 6: PRINT "SAME GAME ? Y)ES, N)O
     , E)ND ";: GET A$
915  IF A$ = "E" THEN  END
920  IF A$ = "Y" OR A$ =  CHR$ (13) THEN  GOTO 113
     0
1000  TEXT : HOME
1010  VTAB 5: HTAB 15: PRINT "ASTROFIELD"
1050  VTAB 10: HTAB 1: INPUT "ENTER DIFFICULTY LEV
     EL (5-18) ";DL$
1055  IF  LEN (DL$) < 1 OR  LEN (DL$) > 2 THEN  GOTO
     1050
1060 E = 0: FOR I = 1 TO  LEN (DL$):CH$ =  MID$ (D
     L$,I,1): IF  ASC (CH$) < 48 OR  ASC (CH$) > 5
     7 THEN E = 1
1070  NEXT I: ON E GOTO 1050
1080 DL =  VAL (DL$)
1090  IF DL < 5 THEN  VTAB 18: PRINT "TOO EASY": FOR
     I = 1 TO 1000: NEXT I: VTAB 18: CALL  - 868: GOTO
     1050
1100  IF DL > 18 THEN  VTAB 18: PRINT "TOO HARD": FOR
     I = 1 TO 1000: NEXT I: VTAB 18: CALL  - 868: GOTO
     1050
1110  VTAB 15: HTAB 10: INPUT "INSTRUCTIONS (Y/N)
     ";A$: IF A$ = "Y" THEN  GOTO 1200
1130 J = 0: FOR I = 1024 TO 1920 STEP 128
1140 J = J + 1
1150  IF J = DL + 2 THEN CP = I:I = 1921: GOTO 118
     0
1160  IF J + 8 = DL + 2 THEN CP = I + 40:I = 1921:
      GOTO 1180
1170  IF J + 16 = DL + 2 THEN CP = I + 80:I = 1921
```

```
1180  NEXT I
1190 BP = CP:SF = 1:CC = 190:MV =  + 1:PT = 0: GOTO
     100
1200  TEXT : HOME
1210  VTAB 3: HTAB 14: PRINT "INSTRUCTIONS"
1220  VTAB 6: PRINT "THE LEFT AND RIGHT ARROW KEYS
      CONTROL ": PRINT "THE DIRECTION OF MOVEMENT
     OF THE ROCKET.";: PRINT "PRESS THE SPACE BAR
     TO MOVE THE ROCKET."
1230  PRINT : PRINT : HTAB 4: PRINT "PRESS SPACE B
     AR TO BEGIN THE GAME"
1240  VTAB 20: HTAB 8: PRINT "PRESS ANY KEY WHEN R
     EADY": WAIT  - 16384,128
1250  GOTO 1130
```

# One on One

Chris York
Apple Version by Patrick Parrish

*Can you defend your wall against fate, your opponent, and a bouncy but determined brick? "One on One" will let you find out. Paddles required to play.*

In "One on One," two players go head-to-head in an attempt to knock down the wall that the opponent is protecting. Player 1 tries to protect the wall at the top of the screen, while player 2 defends the wall at the bottom.

Each player's "swatter" (the horizontal line closest to the middle section of the screen) is used to intercept the ball before it hits his wall and destroys a section. When the ball hits either player's swatter, it bounces toward the opponent's wall. En route, the flight of the ball may be changed or impeded by barriers or additional sections of wall—a feature that serves to make the game faster and more exciting.

Eventually, the ball will break through. The first player to get the ball past his opponent's wall wins the game and receives an appropriate victory message.

## Two Skill Levels

One on One is played with paddles and offers two skill levels. On level 1, all ball movement is at a 45-degree angle to the swatter and walls. However, on level 2, players can alter the ball's flight angle by moving the swatter just before the ball strikes it. If this is successful (as detected in lines 18–28), then the X component of the ball's velocity is doubled and the ball moves twice as fast horizontally. Vertical ball movement remains the same. To return to normal ball motion, the ball must strike a stationary swatter.

A pleasing feature of this game is the random choice of wall colors each time a new game is played. This is accomplished by the short subroutine at line 30.

For an interesting variation, change the rules so that the object is to *break through* the wall behind you rather than defend it.

## One on One

```
10  GOSUB 2000: GOTO 50
12  IF  SCRN( X,Y + DY) = 1 THEN DX =  - DX:DY =  -
    DY: RETURN
13  IF  SCRN( X + DX,Y) = 15 THEN DX =  - DX: RETURN

14 DY =  - DY: RETURN
18  IF DX =  - 2 THEN DX =  - 1
19  IF DX = 2 THEN DX = 1
22  IF Y + DY = R1 THEN 26
23 X0 =  INT ( PDL (0) / M6) + 2: IF X0 <  > L0 THEN
    DX = 2 * DX
25  RETURN
26 X1 =  INT ( PDL (1) / M6) + 2: IF X1 <  > L1 THEN
    DX = DX * 2
28  RETURN
30 D =  INT ( RND (1) * 13) + 2: IF D = DL OR D =
    13 THEN 30
40  RETURN
50 M6 = 7.73:X0 = 2:X1 = 34:R0 = 7:R1 = 32
110  GOTO 1000
112  REM  PADDLE 0 SUBROUTINE
115 X0 =  INT ( PDL (0) / M6) + 2: IF X0 = L0 THEN
     RETURN
120  COLOR= 0: HLIN L0,L0 + 3 AT R0
130  COLOR= 1: HLIN X0,X0 + 3 AT R0
140 L0 = X0: RETURN
145  REM  PADDLE 1 SUBROUTINE
150 X1 =  INT ( PDL (1) / M6) + 2: IF X1 = L1 THEN
     RETURN
155  COLOR= 0: HLIN L1,L1 + 3 AT R1
160  COLOR= 1: HLIN X1,X1 + 3 AT R1
170 L1 = X1: RETURN
250  FOR I = 1 TO 5:A =  PEEK ( - 16336): NEXT I: RETURN
260  RETURN
280  POKE 768,1: POKE 769,10: CALL 770: RETURN
1000  TEXT : HOME : VTAB 11: HTAB 10: FLASH : PRINT
    "O N E    O N    O N E!": NORMAL
1010  VTAB 17: PRINT  SPC( 13);"LEVEL 1 OR 2 ";: INPUT
    LV: IF LV > 2 OR LV < 1 THEN 1010
1020  HOME : GR : PRINT : PRINT : PRINT : PRINT : FOR
    Z = 1 TO 35 STEP 34: FOR Y = Z TO Z + 3
1030  GOSUB 30
1035  COLOR= D:DL = D
1040  HLIN 2,37 AT Y: NEXT Y: NEXT Z
1043  FOR Z = 8 TO 28 STEP 10: FOR Y = 19 TO 21: GOSUB
    30: COLOR= D:DL = D
1045  HLIN Z,Z + 4 AT Y: NEXT Y: NEXT Z: COLOR= 15
```

```
1048  FOR I = Ø TO 38 STEP 38: VLIN 1,38 AT I: VLIN
      1,38 AT I + 1: NEXT I: IF LV = 1 THEN 1056
1049  FOR I = 7 TO 32 STEP 25: VLIN 17,23 AT I: VLIN
      17,23 AT I + 1: NEXT I
1050  FOR X = 13 TO 26 STEP 13: FOR Y = 11 TO 23 STEP
      12: VLIN Y,Y + 5 AT X: VLIN Y,Y + 5 AT X + 1:
      NEXT Y: NEXT X: GOTO 1059
1056  FOR I = 5 TO 35 STEP 30: VLIN 17,23 AT I: NEXT
      I
1057  FOR X = 14 TO 26 STEP 12: FOR Y = 11 TO 24 STEP
      13: VLIN Y,Y + 5 AT X: NEXT Y: NEXT X
1059  COLOR= 1: GOSUB 130: GOSUB 160
1060  PRINT  SPC( 8);"PRESS THE FIRE BUTTON ON": PRINT
      SPC( 4);"PADDLE Ø OR 1 TO START THE GAME"
1070 PØ =  PEEK ( - 16287):P1 =  PEEK ( - 16286): IF
      PØ > 127 OR P1 > 127 THEN 1090
1080  GOSUB 115: GOSUB 150: GOTO 1070
1090  PRINT : PRINT : PRINT : PRINT : REM  CLEAR T
      EXT WINDOW
1100  REM  GAME ROUTINE
1110 X =  INT ( RND (1) * 9) + 17:Y = 23:DX = 1:DY
      = 1
1120  IF  RND (1) < .5 THEN DX =  - 1
1130  IF  RND (1) < .5 THEN DY =  - 1:Y = 17
1135  GOTO 1180
1140  COLOR= Ø: PLOT X,Y: IF  ABS (DX) = 2 AND ( SCRN(
      X + DX / 2,Y + DY) <  > 15 AND  SCRN( X + DX /
      2,Y + DY) <  > 1) THEN  PLOT X + DX / 2,Y + D
      Y
1150 X = X + DX:Y = Y + DY: COLOR= 13: PLOT X,Y: IF
      Y > 4 AND Y < 35 THEN FL = Ø
1160  IF (L < 15 AND L > 1 AND OLDL < 15 AND OLDL >
      1) OR (L < 15 AND L > 1 AND FL = 1) THEN  GOSUB
      250: GOTO 1180
1170  IF L < 15 AND L > 1 THEN  GOSUB 250:DY =  -
      DY: IF Y < 5 OR Y > 34 THEN FL = 1
1180  GOSUB 115: GOSUB 150: IF Y = Ø OR Y = 39 THEN
      1250
1190 OLDL = L
1200 L =  SCRN( X + DX,Y + DY)
1210  IF L = Ø THEN 1140
1220  IF L = 15 THEN  GOSUB 280: GOSUB 12: GOTO 12
      ØØ
1230  IF L = 1 AND LV = 1 THEN  GOSUB 280:DY =  -
      DY
1235  IF L = 1 AND LV = 2 THEN  GOSUB 280: GOSUB 1
      8:DY =  - DY: GOTO 1200
1240  GOTO 1140
1250  REM  WINNER
```

```
1270  IF Y = 39 THEN  PRINT  SPC( 5);"!!!VICTORY G
      OES TO PLAYER 1!!!"
1280  IF Y = 0 THEN  PRINT  SPC( 5);"!!!VICTORY GO
      ES TO PLAYER 2!!!"
1290  FOR I = 1 TO 1000: NEXT I
1300  PRINT : PRINT  SPC( 5);"PRESS A PADDLE BUTTO
      N TO PLAY": PRINT  SPC( 5);"AGAIN, Q TO QUIT"
      ;
1310  POKE  - 16368,0:P0 =  PEEK ( - 16287):P1 =  PEEK
      ( - 16286): IF P0 > 127 OR P1 > 127 THEN 1000

1320  IF  PEEK ( - 16384) =  ASC ("Q") + 128 THEN
      1400
1330  GOTO 1310
1400  POKE  - 16368,0: TEXT : HOME : END
2000  REM  SOUND ROUTINE
2010  FOR I = 770 TO 795: READ M: POKE I,M: NEXT
2020  DATA  172,01,03,174,01,03,169,04,32,168,252,
      173,48,192,232,208,253,136,208,239,206,0,03,2
      08,231,96
2030  RETURN
```

# Roader

Brian Foley
Apple Translation by Chris Poer

*Your driving skills and endurance are put to the test as you careen around curves and dodge highway obstacles in "Roader." Paddles required.*

The object of "Roader" is to control a car on a winding road while dodging obstacles. The farther you drive, the more dangerous the road becomes—but the longer you stay on the pavement, the higher your score. If you hit the side of the road or crash into an obstacle, you'll hear appropriate sounds and the run will end. Your score will appear on the screen.

When you run the program, the computer will wait for you to select a level (1–4). Level 1 is for beginners. On level 2 there are more curves and you have to take them at higher speed. Level 3 gives you slower speeds and a less curvy road, but it puts obstacles in your path. Finally, when you're ready for the big time, level 4 challenges you with a curvy road, high speeds, and obstacles too. With all of that to choose from, Roader should be challenging for everyone.

The car is steered with paddle 0.

## Roader

```
100 N$ = " REDAOR":D = 0:A = 0:B = 0
110  HOME
120  FOR I = 1 TO 7:N$(I) =  MID$ (N$,I,1): NEXT I

130  FOR I = 1 TO 7:A = A + .4:N =  INT ( COS (A) *
     8)
140  VTAB 24 - D - I: HTAB 20 + N: PRINT N$(I)
150  NEXT I:B = B + .4:A = B: IF D = 16 THEN 170
160 D = D + 1: GOTO 130
170  VTAB 12: PRINT " WHAT SKILL LEVEL DO YOU WISH
      TO PLAY?"
180  PRINT : PRINT "1) EASY";: HTAB 26: PRINT "2)
     INTERMEDIATE"
190  PRINT "3) DIFFICULT";: HTAB 26: PRINT "4) EXP
     ERT"
200  PRINT : PRINT "  USE PADDLE 0 TO CONTROL YOUR
      CAR.": PRINT
210  GET LV$:LV =  VAL (LV$)
220  IF LV < 1 OR LV > 4 THEN 210
230 C = 0: IF LV = 2 OR LV = 4 THEN C = .05
```

```
240 C = C + .05:C1 = 14:C2 = 25:A = 0:SC = 8
250 N$ = "                                        "
260  HOME
270 A = A + C + LV / 16:Y =  INT ( COS (A) * 10)
280  POKE YLOC,160: INVERSE
290  PRINT  LEFT$ (N$,C1 + Y);: PRINT "^";: HTAB C
     2 + Y: PRINT "^";: PRINT  LEFT$ (N$,C1 - Y +
     1);
300 N =  INT ( PDL (0) / 7):XLOC = N + 1360
310  IF  PEEK (XLOC) = 30 OR  PEEK (XLOC) = 42 OR
      PEEK (XLOC) = 32 THEN 380
320  NORMAL : POKE XLOC,200:YLOC = XLOC
330  IF LV = 1 OR LV = 2 THEN 350
340 G =  INT ( RND (1) * 120): IF G = 1 THEN G =  INT
     ( RND (1) * 39): POKE 1872 + G,42
350 D = D + 1: IF D / 120 =  INT (D / 120) AND D <
     480 THEN C1 = C1 + 1:SC = SC - 1
360  IF C < .25 THEN C = C + .001
370  GOTO 270
380  FOR I = 1 TO 20
390  FOR C = 1 TO 15:W =  PEEK ( - 16336): NEXT C
400  POKE XLOC - 128,220: POKE XLOC - 128,225: POKE
     XLOC - 128,239: POKE XLOC - 128,223
410  NEXT I: NORMAL
420  HOME : VTAB 5: HTAB 10: PRINT " YOUR SCORE IS
      ";: INVERSE : PRINT  INT (10000 / SC): NORMAL

430  VTAB 10: PRINT "HIT THE PADDLE BUTTON TO PLAY
      AGAIN AT": HTAB 10: PRINT "THE SAME LEVEL, O
     R:": PRINT
440  PRINT "TYPE (S) TO START OVER, (E) TO END."
450  IF  PEEK ( - 16384) = 197 THEN  POKE  - 16287
     ,0: END
460  IF  PEEK ( - 16384) = 211 THEN  POKE  - 16287
     ,0: GOTO 100
470  IF  PEEK ( - 16287) > 127 THEN 230
480  GOTO 450
```

# Caves of Ice

Marvin Bunker and Robert Tsuk

*Here's a game that will send chills down your spine. "Caves of Ice" puts you inside a three-dimensional ice cavern and challenges you to find your way out. And you thought it was chilly in the basement.*

Ever wondered what it would be like to be trapped inside a giant ice cube? "Caves of Ice" will give you a chance to find out. Imagine yourself imprisoned somewhere inside a five-story structure made entirely of ice. Each floor has 25 rooms in a five-by-five array. Carved into the walls of each room are one or more openings; there may be exits to the north, south, east, or west, as well as trap doors leading up or down.

Unfortunately, only *one* door opens to the outside. You may find it in any of the exterior rooms—in a wall, the ceiling, or the floor.

Your goal is to escape as quickly as possible. It's getting chilly, and you left your mittens at home.

## Exploring the Caves

As you stand in the maze, you can see straight ahead, up, down, left, and right. To see behind you, you'll have to turn (by pressing the F key) and face a different direction.

Navigation is simple. Move through the maze by typing the N, S, E, W, U, and D keys to specify the direction of movement. However, if you type F to change the direction you are facing, you'll need to enter a number instead of a letter to specify the new direction. Remember that N=1, S=2, E=3, and W=4.

To escape from the caves, it helps to be methodical. One proven strategy is to travel in one direction as far as you can go. At that point, assume that you've reached an outside wall, and explore it carefully for an exit. Be careful, though. Those icy rooms look very much alike, and if you're careless you could wander around inside the caves forever.

Once you do find your way out, you'll have the option of trying the same maze again to improve your time. Alternately, you may decide to play a new random maze.

If you decide to take a break, press Q (to *Quit* that round) and you'll have the opportunity to save the current maze for

future exploration. Following the prompts, select a filename and press RETURN. The maze (and your location in it) will be saved under the name you selected. To return to that maze, answer YES at the prompt RESTART OLD MAZE and then type in the name of the maze you want to explore. Be sure to use the correct name, or you'll get an END OF DATA error.

## Cold Clues

If you find the game too challenging, you can type an asterisk (*) to learn your location in the maze. You will be given X and Y coordinates (0–4) on the current level, as well as a value for A (0–4) that indicates which level you are on.

If you get hopelessly lost, the program will even show you the coordinates of the exit. Press the question mark (?) key (remember to press SHIFT too) to display the coordinates of the exit. It's nice to know you have that as a last resort. But remember: True adventurers frown on using the ?—unless the hot coffee is running low.

## Caves of Ice

```
1  DATA  201,84,208,15,32,177,0,32,248,230,138,72,
     32,183,0,201,44,240,3,76,201,222,32,177,0,32,
     248,230
2  FOR I = 768 TO 833: READ P: POKE I,P: NEXT I
3  DATA  104,134,3,134,1,133,0,170,160,1,132,2,173
     ,48,192,136,208,4,198
4  DATA  1,240,7,202,208,246,166,0,208,239,165,3,1
     33,1,198,2,208,241,96
5  POKE 1013,76: POKE 1014,0: POKE 1015,3
10  TEXT : HOME
90  GOSUB 2000
100  DIM FC(5,7): DIM FC$(5)
105 FC$(1) = "NORTH":FC$(2) = "SOUTH":FC$(3) = "EA
     ST":FC$(4) = "WEST"
110  FOR B = 1 TO 4: FOR I = 1 TO 6: READ FC(B,I):
      NEXT : NEXT
115  GOTO 155
120  HPLOT 0,0 TO 279,0 TO 279,159 TO 0,159 TO 0,0
      TO 69,29 TO 209,29 TO 209,129 TO 69,129 TO 6
     9,29: HPLOT 209,29 TO 279,0: HPLOT 209,129 TO
     279,159: HPLOT 69,129 TO 0,159: RETURN
125  RETURN
```

```
130  HPLOT 109,9 TO 169,9 TO 159,19 TO 119,19 TO 1
     09,9: HPLOT 119,19 TO 119,9: HPLOT 159,19 TO
     159,9: RETURN
135  HPLOT 119,139 TO 159,139 TO 169,149 TO 109,14
     9 TO 119,139: HPLOT 119,139 TO 119,149: HPLOT
     159,139 TO 159,149: RETURN
140  HPLOT 19,39 TO 49,49 TO 49,139: HPLOT 19,149 TO
     19,39: HPLOT 19,139 TO 49,139: HPLOT 19,49 TO
     49,49: RETURN
145  HPLOT 119,59 TO 159,59 TO 159,129 TO 119,129 TO
     119,59 TO 129,69 TO 149,69 TO 149,119 TO 129,
     119 TO 129,69: HPLOT 149,69 TO 159,59: HPLOT
     149,119 TO 159,129: HPLOT 129,119 TO 119,129:
      RETURN
150  HPLOT 229,49 TO 259,39 TO 259,149: HPLOT 229,
     139 TO 229,49: HPLOT 229,49 TO 259,49: HPLOT
     229,139 TO 259,139: RETURN
155  DIM S$(6,6)
160  INPUT "RESTART OLD MAZE ";Y$: IF  LEFT$ (Y$,1
     ) = "Y" THEN 1360
165  FOR A = 1 TO 5: FOR X = 1 TO 5: FOR Y = 1 TO
     5
167  & T10 * A + 10 * X + 10 * Y,10
170  IF A <  > 5 AND  RND (1) < .80 THEN S$(X,A) =
     S$(X,A) + "0": GOTO 180
175 S$(X,A) = S$(X,A) + "X"
180  IF  MID$ (S$(X,A - 1),(Y - 1) * 6 + 1,1) = "0
     " THEN S$(X,A) = S$(X,A) + "0": GOTO 190
185 S$(X,A) = S$(X,A) + "X"
190  IF Y - 2 < 0 THEN 200
195  IF  MID$ (S$(X,A),(Y - 2) * 6 + 4,1) = "0" THEN
     S$(X,A) = S$(X,A) + "0": GOTO 205
200 S$(X,A) = S$(X,A) + "X"
205  IF Y <  > 5 AND  RND (1) < .8 THEN S$(X,A) =
     S$(X,A) + "0": GOTO 215
210 S$(X,A) = S$(X,A) + "X"
215  IF X <  > 5 AND  RND (1) < .8 THEN S$(X,A) =
     S$(X,A) + "0": GOTO 225
220 S$(X,A) = S$(X,A) + "X"
225  IF  MID$ (S$(X - 1,A),(Y - 1) * 6 + 5,1) = "0
     " THEN S$(X,A) = S$(X,A) + "0": GOTO 235
230 S$(X,A) = S$(X,A) + "X"
235  NEXT : NEXT : NEXT
240 X =  INT ( RND (1) * 3) + 2:Y =  INT ( RND (1)
      * 3) + 2:A =  INT ( RND (1) * 3) + 2
245 RD =  INT ( RND (1) * 6) + 1: ON RD GOTO 250,2
     55,260,265,270,275
250 A = 5:P1$ =  LEFT$ (S$(X,A),(Y - 1) * 6):L = 2
     9 -  LEN (P1$):P2$ =  RIGHT$ (S$(X,A),L):S$(X
     ,A) = P1$ + "0" + P2$: GOTO 280
```

```
255 A = 1:P1$ =  LEFT$ (S$(X,A),(Y - 1) * 6 + 1):L
     = 29 -  LEN (P1$):P2$ =  RIGHT$ (S$(X,A),L):
    S$(X,A) = P1$ + "0" + P2$: GOTO 280
260 Y = 5:P1$ =  LEFT$ (S$(X,A),(Y - 1) * 6 + 3):L
     = 29 -  LEN (P1$):P2$ =  RIGHT$ (S$(X,A),L):
    S$(X,A) = P1$ + "0" + P2$: GOTO 280
265 Y = 1:P1$ =  LEFT$ (S$(X,A),(Y - 1) * 6 + 2):L
     = 29 -  LEN (P1$):P2$ =  RIGHT$ (S$(X,A),L):
    S$(X,A) = P1$ + "0" + P2$: GOTO 280
270 X = 5:P1$ =  LEFT$ (S$(X,A),(Y - 1) * 6 + 4):L
     = 29 -  LEN (P1$):P2$ =  RIGHT$ (S$(X,A),L):
    S$(X,A) = P1$ + "0" + P2$: GOTO 280
275 X = 1:P1$ =  LEFT$ (S$(X,A),(Y - 1) * 6 + 5):L
     = 29 -  LEN (P1$):P2$ =  RIGHT$ (S$(X,A),L):
    S$(X,A) = P1$ + "0" + P2$: GOTO 280
280 SX = X:SY = Y:SA = A
290  VTAB 23: PRINT "HIT ANY KEY TO START"
300  IF  PEEK ( - 16384) < 127 THEN 300
310  POKE  - 16368,00
1000 X =  INT ( RND (1) * 5) + 1:Y =  INT ( RND (1
    ) * 5) + 1:A =  INT ( RND (1) * 5) + 1:FC = 1
    : GOTO 1220
1010  HOME : VTAB 22: HTAB 18: PRINT FC$(FC):A$ =
    "":D = 0: IF LS = 1 THEN  PRINT X,Y,A
1020  VTAB 22: PRINT "TIME :";T: FOR TIME = 1 TO 8
    0
1025  IF  PEEK ( - 16384) > 127 THEN 1030
1027  NEXT :T = T + 1: VTAB 22: PRINT "TIME :";T: GOTO
    1020
1030  GET A$
1035  IF A$ = "*" THEN LS = 1
1040  IF A$ = "Q" THEN 1300
1050  IF A$ = "U" THEN D = 1
1060  IF A$ = "D" THEN D = 2
1070  IF A$ = "N" THEN D = 3
1080  IF A$ = "S" THEN D = 4
1090  IF A$ = "E" THEN D = 5
1100  IF A$ = "?" THEN 1290
1110  IF A$ = "W" THEN D = 6
1120  IF A$ = "F" THEN  GOTO 1280
1130  IF D = 0 THEN 1010
1135 T = T + 1
1140  IF  MID$ (S$(X,A),(Y - 1) * 6 + D,1) <  > "0
    " THEN  PRINT  CHR$ (7): GOTO 1010
1150  ON D GOTO 1160,1170,1180,1190,1200,1210
1160 A = A + 1: GOTO 1220
1170 A = A - 1: GOTO 1220
1180 Y = Y - 1: GOTO 1220
1190 Y = Y + 1: GOTO 1220
```

```
1200 X = X + 1: GOTO 1220
1210 X = X - 1: GOTO 1220
1220  IF X > 5 OR X < 1 OR Y > 5 OR Y < 1 OR A > 5
      OR A < 1 THEN  PRINT "YOU WIN": & T100,100: &
     T100,50: & T100,50: & T75,66: & T100,66: & T7
     5,66: & T60,255: GOTO 3000
1230  HGR : HCOLOR= 3: HPLOT 0,0: CALL 62454: HCOLOR=
     0: GOSUB 120
1240  FOR I = 1 TO 6: IF  MID$ (S$(X,A),(Y - 1) *
     6 + I,1) = "X" THEN  NEXT : GOTO 1010
1250 R = FC(FC,I) + 1
1260  HCOLOR= 0: ON R GOSUB 125,130,135,140,145,15
     0
1270  NEXT : GOTO 1010
1280  INPUT "WHAT FACING 1-N 2-S 3-E 4-W";FC: IF F
     C < 1 OR FC > 4 THEN 1280
1285  GOTO 1220
1290  INVERSE : HTAB 18: PRINT SX;"  ";SY;"  ";SA:
      NORMAL : GOTO 1220
1300  PRINT "DO YOU WANT TO SAVE THIS MAZE": INPUT
     Y$: IF  LEFT$ (Y$,1) <  > "Y" THEN  GOTO 3000

1310  INPUT "WHAT DO YOU WANT TO CALL IT ";N$
1320 D$ =  CHR$ (4)
1330  PRINT D$;"OPEN OLD MAZE/";N$: PRINT D$;"WRIT
     E OLD MAZE/";N$
1340  FOR A1 = 1 TO 5: FOR X1 = 1 TO 5: PRINT S$(X
     1,A1): NEXT : NEXT : PRINT X: PRINT Y: PRINT
     A: PRINT T: PRINT FC
1350  PRINT D$;"CLOSE OLD MAZE/";N$: GOTO 3000
1360  INPUT "WHAT IS ITS NAME ";N$
1370 D$ =  CHR$ (4)
1380  PRINT D$;"OPEN OLD MAZE/";N$: PRINT D$;"READ
      OLD MAZE/";N$
1390  FOR A1 = 1 TO 5: FOR X1 = 1 TO 5: INPUT S$(X
     1,A1): NEXT : NEXT : INPUT X: INPUT Y: INPUT
     A: INPUT T: INPUT FC
1400  PRINT D$;"CLOSE OLD MAZE/";N$: GOTO 1220
2000  VTAB 10: HTAB 14: INVERSE : PRINT "CAVES OF
     ICE": NORMAL : VTAB 22: INPUT "DO YOU WANT IN
     STRUCTIONS ";Y$: IF  LEFT$ (Y$,1) <  > "Y" THEN
      RETURN
2010  HOME : PRINT "THE OBJECT OF MAZE IS TO FIND
     YOUR WAY": PRINT : PRINT "OUT OF A 5X5X5 CUBI
     C MAZE. IN ONE OF THE": PRINT "ROOMS THERE IS
      AN EXIT OUT OF THE MAZE."
```

```
2020  PRINT : PRINT "YOU MUST TRY TO FIND IT IN AS
      FEW TURNS ": PRINT "AS POSSIBLE. THE COMMAND
     S ARE  :"
2030  PRINT : HTAB 6: INVERSE : PRINT "U";: NORMAL
     : PRINT "-UP";: HTAB 17: INVERSE : PRINT "S";
     : NORMAL : PRINT "-SOUTH"
2040  PRINT : HTAB 6: INVERSE : PRINT "D";: NORMAL
     : PRINT "-DOWN";: HTAB 17: INVERSE : PRINT "E
     ";: NORMAL : PRINT "-EAST"
2050  PRINT : HTAB 6: INVERSE : PRINT "N";: NORMAL
     : PRINT "-NORTH";: HTAB 17: INVERSE : PRINT "
     W";: NORMAL : PRINT "-WEST"
2060  PRINT : HTAB 6: INVERSE : PRINT "Q";: NORMAL
     : PRINT "-QUIT";: HTAB 17: INVERSE : PRINT "F
     ";: NORMAL : PRINT "-CHANGE FACING"
2070  VTAB 23: PRINT "HIT ";: INVERSE : PRINT "SPA
     CE";: NORMAL : PRINT " FOR MORE"
2080  IF  PEEK ( - 16384) < 127 THEN 2080
2090  POKE  - 16368,0: HOME : INVERSE : PRINT "F";
     : NORMAL : PRINT " WILL COME BACK WITH A QUES
     TION AS TO": PRINT : PRINT "WHICH FACING YOU
     WISH.HIT ONLY ONE KEY": PRINT : PRINT "AND ";
     : INVERSE : PRINT "RETURN": NORMAL
2100  PRINT : PRINT "PLEASE WAIT WHILE IT SETS UP
     THE MAZE": PRINT : PRINT : RETURN
3000  TEXT : HOME : VTAB 5: HTAB 12: PRINT "CONGRA
     TULATIONS !"
3010  PRINT : PRINT  TAB( 7)"YOU HAVE FINISHED THE
      MAZE IN ": PRINT  TAB( 7)T;" SECONDS"
3030  INPUT "DO YOU WANT TO PLAY AGAIN ? ";Y$
3040  IF  LEFT$ (Y$,1) = "Y" THEN  RUN
9999  NORMAL
10000  DATA     1,2,4,0,5,3,1,2,0,4,3,5,1,2,3,5,4,
     0,1,2,5,3,0,4
```

# Barrier Battle

Heath Lawrence
Apple Translation by Chris Poer

*Barriers, barriers everywhere—but wait! Is that a hole? Its four levels will leave you breathless, if you don't get all boxed in. Requires paddles.*

The object of "Barrier Battle" is to build barriers to cut off your opponent so that he runs out of room and collides with a wall. You create barriers by guiding a barrier builder with your paddle. Be careful, though. You'll lose the game if you hit one of the screen boundaries or one of the player-built barriers.

Your barrier builder automatically leaves a solid trail as you move it around the screen. However, by pressing the trigger, you can create up to five holes in your barrier. Strategically placed, those holes can spell the difference between victory and defeat—particularly when escape routes become scarce. A legend at the top of the screen shows how many holes each player has left. At the end of each round, the winner is identified (particularly helpful in the case of close calls). The game is over when you or your opponent wins four rounds.

At the beginning of the game, it's a good idea to secure yourself a large part of the playfield. In the long run, it's usually the player with the most real estate who is victorious. But if you should find yourself out of room, try pressing the trigger and moving back and forth. This will only delay the inevitable, but it may stall long enough for the other player to smash into a barrier.

## About the Program

The barriers are drawn on the low-resolution graphics screen. Paddles were chosen to control each player's movement, and direction is based on the change of the values in functions PDL(0) and PDL(1). A positive change (of a preset magnitude) will move you to the right. A negative change will turn you to the left. If you find that the paddles are too sensitive (or not sensitive enough), increase or decrease the number in lines 340 and 400.

## Barrier Battle

```
10  TEXT : HOME :PI = 3.1415927 / 180
20 A = 0:B = 0
30  GOSUB 660
40  REM  INITIALIZATION
50 FIR = 0:SEC = 0
60 XLOC = 20:YLOC = 26:ALOC = 20:BLAC = 25:AVAR =
     0:BVAR =  - 1
70 T1 = 0:T2 = 180:XVAR = 0:YVAR = 1
80 S = 0:T = 0
90  REM  PADDLE SETTING
100  PRINT : PRINT "NOW SET YOUR PADDLE ON THE CEN
     TER VALUE OF 125": PRINT : PRINT : PRINT
130  GOSUB 1000
140  PRINT " HIT A PADDLE BUTTON TO CONTINUE"
150  IF  PEEK ( - 16287) < 128 AND  PEEK ( - 16286
     ) < 128 THEN 150
160 N1 =  PDL (1):N2 =  PDL (0)
170  HOME
180  REM  SET SCREEN
190  GR : HOME : POKE  - 16302,0: CALL  - 1998
200  COLOR= 1: HLIN 0,39 AT 4: HLIN 0,39 AT 47: VLIN
     47,4 AT 0: VLIN 47,4 AT 39
210  COLOR= 13: FOR I = 1 TO 10 STEP 2: PLOT I,2: NEXT

220  COLOR= 4: FOR I = 20 TO 29 STEP 2: PLOT I,2: NEXT

230  COLOR= 13: PLOT ALOC,BLOC: COLOR= 4: PLOT XLO
     C,YLOC
240  FOR I = 1 TO 300: NEXT
250  GOTO 330
260  REM  MOVE PLAYERS
270  COLOR= 13
280  IF S < 5 AND  PEEK ( - 16286) > 127 THEN  COLOR=
     0:S = S + 1: PLOT S * 2 - 1,2
290  PLOT ALOC,BLOC
300  COLOR= 4
310  IF T < 5 AND  PEEK ( - 16287) > 127 THEN T =
     T + 1: COLOR= 0: PLOT 18 + (T * 2),2
320  PLOT XLOC,YLOC
330 O1 = N1:N1 =  PDL (0):O2 = N2:N2 =  PDL (1)
340  IF  ABS (O1 - N1) < 8 THEN 390
350 S1 =  SGN (O1 - N1)
360  IF S1 = 1 THEN T1 = T1 + 90: GOTO 380
370 T1 = T1 - 90
380 XVAR =  INT ( SIN (T1 * PI) + .1):YVAR =  INT
     ( COS (T1 * PI) + .1)
390 XLOC = XLOC + XVAR:YLOC = YLOC + YVAR
400  IF  ABS (O2 - N2) < 8 THEN 450
```

```
410 S2 =  SGN (O2 - N2)
420  IF S2 = 1 THEN T2 = T2 + 90: GOTO 440
430 T2 = T2 - 90
440 AVAR =  INT ( SIN (T2 * PI) + .1):BVAR =  INT
    ( COS (T2 * PI) + .1)
450 ALOC = ALOC + AVAR:BLOC = BLOC + BVAR
460 PNT =  SCRN( XLOC,YLOC):POT =  SCRN( ALOC,BLOC
    )
470  IF PNT = 0 AND POT = 0 THEN  FOR I = 1 TO LEV
    : GOTO 270
480  IF PNT = 4 OR PNT = 1 OR PNT = 13 THEN FIR =
    1
490  IF POT = 4 OR POT = 13 OR POT = 1 THEN SEC =
    1
500  FOR I = 1 TO 1000: NEXT
510  REM  DETERMINING WINNER
520  GOSUB 790: TEXT : HOME
530  IF FIR = 1 AND SEC = 1 THEN  PRINT "IT WAS A
    TIE": GOTO 560
540  IF FIR = 1 THEN B = B + 1: PRINT B$;" WON THI
    S ROUND": GOTO 560
550 A = A + 1: PRINT A$;" WON THIS ROUND"
560  PRINT "THE SCORE IS ": PRINT B;" VICTORIES FO
    R ";B$
570  PRINT A;" VICTORIES FOR ";A$
580  IF B = 4 THEN C$ = B$: GOTO 630
590  IF A = 4 THEN C$ = A$: GOTO 630
600  PRINT "HIT YOUR PADDLE BUTTON TO CONTINUE"
610  IF  PEEK ( - 16287) > 127 OR  PEEK ( - 16286)
     > 127 THEN 40: GOTO 620
620  GOTO 610
630  PRINT : PRINT : PRINT C$;" IS THE WINNER"
640  END
650  IF  PEEK ( - 16287) > 127 OR  PEEK ( - 16286)
     > 127 THEN 40: GOTO 650
660  INVERSE : HTAB 15: PRINT "BARRIER BATTLE"
670  NORMAL : PRINT : PRINT : PRINT "THE OBJECT OF
     THE GAME IS TO FORCE YOUR OPPONENT INTO A WA
    LL."
680  PRINT : PRINT "YOU CANNOT RUN INTO YOUR OWN W
    ALL OR THE";: PRINT "BOUNDARY."
690  PRINT : PRINT "YOU CAN MAKE FIVE HOLES IN THE
     WALL PER ROUND BY PRESSING THE BUTTON ON YOU
    R": PRINT "PADDLE.": PRINT : PRINT "THE NUMBE
    R OF HOLES YOU HAVE LEFT IS": PRINT "SHOWN AT
     THE TOP OF THE SCREEN."
700  PRINT : PRINT "THE FIRST ONE TO WIN FOUR ROUN
    DS WINS": PRINT "THE GAME."
```

```
710  PRINT : PRINT "WHAT SPEED DO YOU WANT (1-4) ?
      <4 IS THE";: PRINT "FASTEST>": INPUT LEV
720 LEV = (4 / LEV - 1) * 40
730  PRINT "WHO IS PLAYER ONE": INPUT B$
735 B$ =  LEFT$ (B$,8)
740  PRINT "WHO IS PLAYER TWO": INPUT A$
745 A$ =  LEFT$ (A$,8)
750  HOME
760  PRINT : PRINT B$;" IS ON TOP AND USES PADDLE
     1": PRINT A$;" IS UNDERNEATH AND USES PADDLE
     0"
765  PRINT "GET READY!!!!!": FOR D = 1 TO 2000: NEXT
     D
770  RETURN
780  REM  NOISE
790  FOR I = 1 TO 40
800 F =  PEEK ( - 16336)
810  NEXT
820  RETURN
1000  VTAB 22: PRINT B$;" IS AT": VTAB 22: HTAB 15
     : PRINT "          ": VTAB 22: HTAB 15: PRINT
     , PDL (1)
1010  VTAB 23: PRINT A$;" IS AT": VTAB 23: HTAB 15
     : PRINT "          ": VTAB 23: HTAB 15: PRINT ,
      PDL (0)
1020  IF  PDL (1) < 122 OR  PDL (1) > 128 OR  PDL
     (0) < 122 OR  PDL (0) > 128 THEN 1000
1030  RETURN
```

# Devastator

David R. Arnold
Apple Version by Todd Koumarian

*You and your comrades approach the hostile Devastator—a powerful mothership ready to destroy Earth. Out of nowhere, guardian ships attack. You have 30 seconds to destroy all of them or the Earth will be lost. Requires a joystick (for one player) or paddles (for two players).*

"Devastator" is an action game where you must save Earth from aliens.

You and your comrades are in one-person spaceships skimming the surface of a huge alien craft known as *Devastator.* You're being attacked by alien ships, and you have 30 seconds to destroy the attackers before *Devastator* annihilates Earth.

## How It Works

Devastator is written in Applesoft, with several machine language (ML) subroutines. A single player can aim with the joystick. Alternately, two players can use paddles and work together. When two are playing, one controls movement from left to right while the other controls movement up and down.

When playing Devastator, there is no need to hold down the fire button. Merely placing the crosshairs on the moving alien interceptor will insure its destruction. However, if you take too long to aim, your foe will destroy the earth.

The crosshairs and alien interceptors are drawn using shape tables. The Applesoft SCALE and ROT commands are used to create the approach (and explosion) of the interceptors. The shape table is POKEd in at line 8020 and sits at $300.

Earth and its subsequent destruction are handled by short ML routines. The world drawing routine resides at $9100 and is CALLed once every loop through the main program or whenever the image is garbled. The routine stores the bit image in screen memory from a data table at $1980–$1A6F. The world drawing routine ORs the image with what is on the screen and then stores it so that it does not erase what is already there.

The destruction of Earth at the end of the game is handled by an ML routine at $1A70. It stores random garbage in a randomly selected line and byte in screen memory; the routine momentarily confines the garbage to the area around Earth and then expands it to the edges of the screen. The effect is that of a rapidly expanding explosion.

The ML random number generator used at $1AFF is a common one that generates random nybbles and masks them together for random byte values. A short lookup table is used by both the world drawing and world exploding routines to find the addresses of the first 40 lines on the screen. The table lies between $1930 and $197F; its use has been well documented.

When you're typing in Devastator, it's important that the data be typed in correctly. If the data for the shape tables or the world image has errors, the images will look malformed. If there are errors in the data for the ML routines, the computer will most likely crash or write all over your program. If you have a printer, use it to check the data. Finally, remember to save your program before you run it.

## Devastator

```
5  TEXT : HOME : VTAB 10: HTAB 15: PRINT "PLEASE W
     AIT"
10  GOSUB 8000
15  HGR : POKE  - 16302,0:EX = 140:EY = 90:Q = 1:D
     L = 10
20  SCALE= 1: ROT= 0
25  CALL 6400
30  HCOLOR= 7: HPLOT 0,100 TO 91,100 TO 91,130 TO
     189,130 TO 189,100 TO 279,100
35  HPLOT 91,100 TO 0,191: HPLOT 189,100 TO 279,19
     1
40  HPLOT 91,130 TO 30,191: HPLOT 189,130 TO 249,1
     91
45  GOTO 3999
50 I = I + 1: IF I > 3 THEN I = 1
55  ON I GOTO 100,200,300
100  HCOLOR= 7: GOSUB 1000: HCOLOR= 4: GOSUB 3000:
      RETURN
200  HCOLOR= 7: GOSUB 2000: HCOLOR= 4: GOSUB 1000:
      RETURN
300  HCOLOR= 7: GOSUB 3000: HCOLOR= 4: GOSUB 2000:
      RETURN
```

```
1000  HPLOT 0,105 TO 84,105: HPLOT 86,107 TO 86,13
      2: HPLOT 88,134 TO 190,134: HPLOT 192,132 TO
      192,106: HPLOT 195,105 TO 279,105
1010  RETURN
2000  HPLOT 0,125 TO 63,125: HPLOT 65,127 TO 65,15
      3: HPLOT 69,155 TO 210,155: HPLOT 212,152 TO
      212,127: HPLOT 216,125 TO 279,125
2010  RETURN
3000  HPLOT 0,155 TO 33,155: HPLOT 35,157 TO 35,18
      3: HPLOT 38,185 TO 241,185: HPLOT 243,182 TO
      243,157: HPLOT 245,155 TO 279,155
3010  RETURN
3999 X = 140:Y = 90
4000  HCOLOR= 0: SCALE= 1: DRAW 1 AT X,Y:PX = X:PY
      = Y
4010 X =  PDL (0)
4020 Y =  PDL (1): IF Y > 124 THEN Y = 124
4030  IF Y < 6 THEN Y = 6
4040  IF X > 95 AND X < 165 THEN 4060
4050  IF Y > 94 THEN  HCOLOR= 7:X = PX:Y = PY: DRAW
      1 AT PX,PY: GOSUB 50
4060  HCOLOR= 7: DRAW 1 AT X,Y
4070  GOSUB 50
4090  IF  ABS (EY - Y) > 9 THEN 4120
4100  IF T = 3 AND EX - X > 3 AND EX - X < 13 AND
      ABS (EY - Y) < 6 THEN 5000
4110  IF T = 4 AND EX - X >  - 9 AND EX - X < 13 THEN
      5000
4120  IF F = 0 THEN 4140
4130  HCOLOR= 0: SCALE= SC: DRAW SS AT EX,EY
4140 W =  INT ( RND (1) * 2) + 1: IF W = 2 THEN W =
      - 1
4150 EX = EX + W *  INT ( RND (1) * 30):EY = EY +
      W *  INT ( RND (1) * 20)
4160  IF EX < 0 THEN EX = 0
4170  IF EX > 260 THEN EX = 260
4180  IF EY < 8 THEN EY = 8
4190  IF EY > 121 THEN EY = 121
4200  IF EX > 95 AND EX < 165 THEN 4220
4210  IF EY > 90 THEN EY = 90
4220 DI = DI + Q *  INT ( RND (1) * 20): IF DI > 1
      00 THEN DI = 100: IF  INT ( RND (1) * 2) = 0 THEN
      Q =  - 1
4230  IF DI < 0 THEN DI = 0: IF  INT ( RND (1) * 2
      ) = 0 THEN Q = 1
4240  IF DI < 30 THEN SC = 1:SS = 2
4250  IF DI > 30 AND DI < 70 THEN SC = 2:SS = 2
4260  IF DI > 71 THEN SC = 1:SS = 3
4270  HCOLOR= 7: SCALE= SC: DRAW SS AT EX,EY
4280 T = SS + SC
```

```
4290 F = 1
4300 TI = TI + 1
4310  IF TI > DL THEN 10000
4320  CALL 6400
4330  GOTO 4000
5000  HCOLOR= 0: DRAW 1 AT X,Y
5010  HCOLOR= 7: FOR I = SC TO SC + 15: SCALE= I: DRAW
     SS AT EX,EY: POKE 6952,15 + I: POKE 6953,3: CALL
     6954: NEXT
5020  HCOLOR= 0: FOR I = SC TO SC + 15: SCALE= I: DRAW
     SS AT EX,EY: POKE 6952,30 + I: POKE 6953,3: CALL
     6954: NEXT
5030 SR = SR + 10 * (101 - DI)
5040  CALL 6400
5050 DI = 0
5060 EX =  INT ( RND (1) * 60) + 95:EY =  INT ( RND
     (1) * 80): HCOLOR= 7
5070  FOR I = 20 TO 1 STEP  - 1: ROT= 1.05 * I - 1
     : SCALE= I: DRAW 2 AT EX,EY: POKE 6952,I + 40
     : POKE 6953,3: CALL 6954: NEXT
5080  HCOLOR= 0: FOR I = 20 TO 1 STEP  - 1: ROT= 1
     .05 * I - 1: SCALE= I: DRAW 2 AT EX,EY: POKE
     6952,20 + I: POKE 6953,3: CALL 6954: NEXT
5090 DD = DD + 1
5100  IF (DD / 4) =  INT (DD / 4) THEN DL = DL - 2

5110  IF DL < 2 THEN DL = 2
5120 TI = 0
5130  GOTO 20
8000 I = 768
8010  POKE 232,0: POKE 233,3
8020  READ A: IF A =  - 1 THEN 9030
8030  POKE I,A:I = I + 1: GOTO 8020
9000  DATA  3,0,8,0,31,0,43,0,45,45,45,45,45,64,3
     6,164,146,82,41,45,45,45,45,221,219,219,219,2
     10,54,54,0
9010  DATA  36,37,45,45,46,54,54,55,63,63,60,36,0
     ,36,36,45,36,45,45,36,45,45,45,54,45,45,54,45
     ,54,54,54,54,63
9020  DATA  54,63,63,54,63,63,63,36,63,63,36,63,3
     6,36,0,-1
9030 AD = 6448
9040  FOR I = 0 TO 1: FOR J = 0 TO 1: FOR K = 0 TO
     7: POKE AD + (I * 16 + J * 8) + K,32 + (4 * K
     ) + I: NEXT : NEXT : NEXT
9050  FOR K = 0 TO 7: POKE AD + (I * 16) + K,32 +
     (4 * K) + I: NEXT
9060  FOR Q = 0 TO 4: FOR J = 0 TO 7: IF (Q / 2) =
      INT (Q / 2) THEN W = 0: GOTO 9080
9070 W = 1
```

```
9080  POKE AD + (I * 15) + 10 + J + (8 * Q),128 *
      W
9090  NEXT : NEXT
9299  FOR I = 6400 TO 6447: READ A: POKE I,A: NEXT
      : GOTO 9399
9300  DATA     32,74,255,169,0,168,170,133,0,164,0,
      185,48,25,133,4,185,88,25,133,3,160,17,189
9310  DATA      128,25,17,3,145,3,232,200,192,23,20
      8,243,230,0,165,0,201,40,208,221,32,63,255,96

9399  FOR I = 6528 TO 6974: READ A: POKE I,A: NEXT
      : RETURN
9400  DATA  0,0,124,15,0,0,0,64,15,124,0,0,0,112,1
      ,96,3,0,0,60,14,0,15,0,0,14,31,56,28,0,0,7,59
      ,124,56,0,64,3,119,111,112,0,64,1,6,96,96,0,9
      6,1,6,96,96,1,112,0,7,96,64,3
9410  DATA  56,0,3,48,0,7,24,0,3,48,0,6,24,0,3,24,
      0,6,28,0,7,24,0,14,12,0,6,24,0,12,14,0,6,48,0
      ,28,6,0,102,55,0,24,6,0,110,60,0,24,6,0,124,1
      24,0,24,6,0,56,64,1,24
9420  DATA  6,0,112,0,3,24,6,0,96,1,0,24,6,0,64,1,
      0,24,6,0,96,7,0,24,14,0,112,12,0,28,12,0,48,1
      2,0,12,28,0,24,24,0,14,24,0,24,24,0,6,24,0,24
      ,28,0,6,56,0,56,12,0,7
9430  DATA     112,0,48,14,64,3,96,1,112,6,96,1,64,
      1,96,7,96,0,64,3,96,3,112,0,0,7,96,3,56,0,0,1
      4,96,3,28,0,0,60,96,1,15,0,0,112,1,96,3,0,0,6
      4,15,124,0,0,0,0,124,15,0,0
9500  DATA    32,74,255,169,0,133,1,133,5,162,5,181
      ,78,149,6,202,208,249,169,0,133,4,32,180,26,2
      30,4,165,4,201,127,208,245,230,1,165,1,201,3,
      208,233,169,0,133,4
9510  DATA    32,219,26,230,4,165,4,201,127,208,245
      ,230,5,165,5,201,5,208,233,32,63,255,96,32,25
      5,26,41,63,201,39,16,247,170,189,48,25,133,3,
      189,88,25,133,2,32,255
9520  DATA    26,41,7,201,7,240,247,24,105,17,168,3
      2,255,26,145,2,96,32,255,26,41,63,201,39,16,2
      47,170,189,48,25,133,3,189,88,25,133,2,32,255
      ,26,41,63,201,39,16
9530  DATA    247,168,32,255,26,145,2,96,32,14,27,1
      33,12,32,14,27,10,10,10,10,5,12,96,56,165,7,1
      01,10,101,11,133,6,162,4,181,6,149,7,202,16,?
      49,165,6,41,15,141,48,192,96
9600  DATA  0,0,173,48,192,136,208,5,206,41,27,240
      ,9,202,208,245,174,40,27,76,42,27,96
```

```
10000  HCOLOR= 7: SCALE= 1: FOR I = 127 TO 20 STEP
     - 5: ROT= I: DRAW 2 AT 135,I: HCOLOR= 0: ROT=
    I + 5: DRAW 2 AT 135,I + 5: HCOLOR= 7: NEXT
10010  FOR I = 1 TO 7 STEP 2: HPLOT 135 + I,0 TO 1
    35 + I,130: HPLOT 135 - I,0 TO 135 - I,130: NEXT

10020  CALL 6768: HOME : VTAB 21: HTAB 7: PRINT "Y
    OU MADE "SR" POINTS BEFORE": VTAB 22: HTAB 9:
     PRINT "PLANETARY DESTRUCTION"
10030  VTAB 23: PRINT "PRESS BUTTON (0) FOR ANOTHE
    R CHANCE TO";
10040  VTAB 24: HTAB 15: PRINT "SAVE EARTH";
10050  POKE  - 16301,0
10060  IF  PEEK ( - 16287) > 127 THEN 10060
10070  IF  PEEK ( - 16287) < 128 THEN 10070
10080  CLEAR : GOTO 15
```

# Quatrainment

Sean Puckett
Apple Translation by Chris Poer

*Fast thinking and careful logic are required to win "Quatrainment," a game in which you race the clock and plan your moves to match a master pattern. A joystick is required.*

The object of "Quatrainment" is to match a pattern generated by the program, using the fewest moves possible and finishing in the shortest amount of time. As the game begins, your game board is drawn at the left of the screen, and the master pattern is displayed at the right. A timer and move counter are also displayed.

A cursor appears in one of the squares on the game board. To change your pattern, use the joystick to move the cursor onto the square you want. Part of your pattern will toggle from on to off, or from off to on, depending on whether you are in the middle, in a corner, or at an edge of the board. The different ways the pattern can change are shown in examples displayed on the screen.

When you match the pattern, your weighted score will be displayed, based on elapsed time and the number of moves you made. The lower your score, the better.

## Quatrainment

```
5  TEXT : HOME : FLASH : VTAB 7: HTAB 17: PRINT "Q
    UATRAINMENT"
10  INVERSE : VTAB 12: HTAB 12: PRINT "PRESS ANY K
    EY TO BEGIN": VTAB 7: HTAB 19: GET XX$: NORMAL
20  GOSUB 1000
30  GOSUB 1100
35  GOSUB 1300
50 P1 =  INT ( PDL (0) / 64):P2 =  INT ( PDL (1) /
    64):X = P1 * 4 + 2:Y = P2 * 6 + 3
60  COLOR= 1: HLIN P1 * 4 + 1,P1 * 4 + 3 AT P2 * 6
     + 1: HLIN P1 * 4 + 1,P1 * 4 + 3 AT P2 * 6 +
    5
70  COLOR= 0: HLIN P1 * 4 + 1,P1 * 4 + 3 AT P2 * 6
     + 1: HLIN P1 * 4 + 1,P1 * 4 + 3 AT P2 * 6 +
    5
```

```
80  IF ( PEEK ( - 16286) > 127 OR  PEEK ( - 16287)
     > 127) THEN  GOSUB 1200: GOSUB 2100:MOV = MO
    V + 1: VTAB 22: HTAB 17: PRINT MOV
90 TC = TC + 1: IF TC > 10 THEN TIME = TIME + 1:TC
     = 0: VTAB 23: HTAB 17: PRINT TIME
100  GOTO 50
1000  GR : COLOR= 6: HLIN 0,16 AT 0: HLIN 24,39 AT
    0: HLIN 0,16 AT 6: HLIN 24,39 AT 6: HLIN 0,16
     AT 12: HLIN 24,39 AT 12
1010  HLIN 0,16 AT 18: HLIN 24,39 AT 18: HLIN 0,16
     AT 24: HLIN 24,39 AT 24
1020  VLIN 0,24 AT 0: VLIN 0,24 AT 23: VLIN 0,24 AT
    4: VLIN 0,24 AT 27: VLIN 0,24 AT 8: VLIN 0,24
     AT 31
1030  VLIN 0,24 AT 12: VLIN 0,24 AT 35: VLIN 0,24 AT
    16: VLIN 0,24 AT 39
1040  RETURN
1100  FOR R = 0 TO 3: FOR RR = 0 TO 3:B(R,RR) =  INT
    ( RND (1) * 1 + .5): NEXT : NEXT
1110  COLOR= 12: FOR RR = 0 TO 3: FOR R = 0 TO 3: IF
    B(R,RR) = 1 THEN  PLOT RR * 4 + 2,R * 6 + 3
1120  NEXT : NEXT
1130  FOR I = 1 TO 10: FOR C = 0 TO 3: FOR R = 0 TO
    3: READ Z1:E(I,C,R) = Z1: NEXT : NEXT : NEXT

1140 S =  INT ( RND (1) * 10 + 1): FOR C = 0 TO 3:
     FOR R = 0 TO 3:D(C,R) = E(S,C,R): NEXT : NEXT

1150  COLOR= 9: FOR RR = 0 TO 3: FOR R = 0 TO 3: IF
    D(R,RR) = 1 THEN  PLOT RR * 4 + 25,R * 6 + 3
1160  NEXT : NEXT
1170  FOR A = 1 TO 9: FOR I = 1 TO 6
1180  READ Z1,Z2:X1(A,I) = Z1:Y1(A,I) = Z2
1190  NEXT : NEXT : RETURN
1200  IF (P1 = 1 OR P1 = 2) AND (P2 = 1 OR P2 = 2)
     THEN A = 1: GOSUB 2000: RETURN
1210  IF (P1 = 1 OR P1 = 2) AND P2 = 0 THEN A = 2:
     GOSUB 2000: RETURN
1220  IF (P1 = 1 OR P1 = 2) AND P2 = 3 THEN A = 3:
     GOSUB 2000: RETURN
1230  IF (P2 = 1 OR P2 = 2) AND P1 = 0 THEN A = 4:
     GOSUB 2000: RETURN
1240  IF (P2 = 1 OR P2 = 2) AND P1 = 3 THEN A = 5:
     GOSUB 2000: RETURN
1250  IF P1 = 0 AND P2 = 0 THEN A = 6: GOSUB 2000:
     RETURN
1260  IF P1 = 3 AND P2 = 0 THEN A = 7: GOSUB 2000:
     RETURN
```

```
1270  IF P1 = 3 AND P2 = 3 THEN A = 8: GOSUB 2000:
      RETURN
1280 A = 9: GOSUB 2000: RETURN
1300  COLOR= 2: VLIN 26,31 AT 4: PLOT 5,27: PLOT 6
     ,28: PLOT 7,27: VLIN 31,26 AT 8
1310  VLIN 33,39 AT 6: HLIN 4,8 AT 36
1320  VLIN 26,31 AT 18: HLIN 18,20 AT 26: HLIN 18,
     20 AT 31
1330  VLIN 33,38 AT 18: VLIN 35,38 AT 19: VLIN 37,
     38 AT 20
1340  VLIN 26,31 AT 31: HLIN 31,34 AT 26: HLIN 31,
     34 AT 29: HLIN 31,34 AT 32
1350  VLIN 37,39 AT 30: VLIN 37,39 AT 31: HLIN 32,
     33 AT 36: HLIN 32,33 AT 35: HLIN 32,33 AT 34:
      VLIN 37,39 AT 34: VLIN 37,39 AT 35
1355  PRINT : PRINT
1360  VTAB 23: PRINT "MOVES";: PRINT ,MOV: PRINT "
     TIME";: PRINT ,TIME: RETURN
2000  FOR I = 1 TO 6
2005  IF X1(A,I) = 1 THEN 2030
2010  IF  SCRN( X + X1(A,I),Y + Y1(A,I)) > 0 THEN
      COLOR= 0: PLOT X + X1(A,I),Y + Y1(A,I):B((Y -
     3 + Y1(A,I)) / 6,(X - 2 + X1(A,I)) / 4) = 0: GOTO
     2030
2020  COLOR= 12: PLOT X + X1(A,I),Y + Y1(A,I):B(((
     Y - 3 + Y1(A,I)) / 6),((X - 2 + X1(A,I)) / 4)
     ) = 1
2030  NEXT : RETURN
2100  FOR R = 0 TO 3: FOR RR = 0 TO 3: IF B(R,RR) <
      > D(R,RR) THEN  RETURN
2110  NEXT : NEXT
2120 SC =  INT (TIME / 10) *  INT (MOV / 5)
2130  TEXT : HOME : VTAB 10: FLASH : HTAB 16: PRINT
     "YOU HAVE WON": VTAB 13: HTAB 11: PRINT "YOUR
      SCORE IS ";SC;" POINTS"
2140  NORMAL : END
20000  DATA  1,1,1,1,1,0,0,1,1,0,0,1,1,1,1,1
20010  DATA  0,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0
20020  DATA  0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0
20030  DATA  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
20040  DATA  1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1
20050  DATA  1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1
20060  DATA  0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1
20070  DATA  1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1
20080  DATA  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20090  DATA  0,0,0,0,1,0,0,1,1,0,0,1,0,0,0,0
```

```
20110  DATA  0,0,-4,0,4,0,0,-6,0,6,1,1
20120  DATA  4,0,-4,0,0,6,1,1,1,1,1,1
20130  DATA   4,0,-4,0,0,-6,1,1,1,1,1,1
20140  DATA  4,0,0,-6,0,6,1,1,1,1,1,1
20150  DATA  -4,0,0,-6,0,6,1,1,1,1,1,1
20160  DATA  0,0,0,6,0,12,4,0,4,6,8,0
20170  DATA  0,0,0,6,0,12,-4,0,-4,6,-8,0
20180  DATA  0,0,0,-6,0,-12,-4,0,-4,-6,-8,0
20190  DATA  0,0,0,-6,0,-12,4,-6,4,0,8,0
```

# Mind Reader

Tim Smith

*It's relatively easy for you to look into your computer's memory—but can it read your mind too? This program may make you think so.*

This game presents you with a bit of computerized ESP. Pick any odd number between 1 and 63, following the onscreen prompts. The computer will then show you five different arrays of numbers, asking you a question about each. After all five have been shown, it will guess your number—and the answer will always be correct.

But how could the computer possibly know? It's easy, once you understand the secret. All you have to do is add the number in the upper-left corner of each set for which the answer is "yes." The total will be the number that was initially chosen.

## Mind Reader

```
10  TEXT : HOME : PRINT  TAB( 13)"The Mind Reader!
    "
20  PRINT : PRINT  TAB( 19)"By Tim Smith": GOSUB 1
    000
23  SPEED= 190
25  HOME : PRINT "In this game, I will show you 6
    sets of numbers. You must answer the question
    s  below them."
26  PRINT : PRINT : PRINT : PRINT : PRINT : HTAB (
    8): PRINT "Press any key to continue."
27  GET SM$
29  HOME : FLASH : PRINT "IMPORTANT:": NORMAL
30  PRINT : PRINT : PRINT "Press RETURN after ever
    y answer unless  directed otherwise."
31  PRINT : PRINT : PRINT : HTAB (8): PRINT "Press
     any key to continue.": GET SP$
35  HOME : PRINT : PRINT : PRINT : PRINT  TAB(
    9)"1  3  5  7  9  11 13 15": PRINT : PRINT  TAB(
    9)"17 19 21 23 25 27 29 31"
40  PRINT : PRINT  TAB( 9)"33 35 37 39 41 43 45 47
    ": PRINT : PRINT  TAB( 9)"49 51 53 55 57 59 6
    1 63"
45  PRINT : PRINT : PRINT : PRINT : PRINT "Pick a
    number,and remember it! Press anykey when you
     have chosen.": GET SP$
53  LET Z = 1
```

```
55  HOME : PRINT : PRINT : PRINT : PRINT : PRINT  TAB(
     9)"2  3  6  7  10 11 14 15": PRINT : PRINT  TAB(
     9)"18 19 22 23 26 27 30 31"
60  PRINT : PRINT  TAB( 9)"34 35 38 39 42 43 46 47
     ": PRINT : PRINT  TAB( 9)"50 51 54 55 58 59 6
     2 63"
65  PRINT : PRINT : PRINT : PRINT : PRINT "Is the
     number you picked in this set?"
70  INPUT A$: IF A$ = "Y" THEN  GOSUB 1100
72  IF A$ = "YES" THEN  GOSUB 1100
73  IF A$ = "Y " THEN  GOSUB 1100
75  HOME : PRINT : PRINT : PRINT : PRINT : PRINT  TAB(
     9)"4  5  6  7  12 13 14 15": PRINT : PRINT  TAB(
     9)"20 21 22 23 28 29 30 31"
80  PRINT : PRINT  TAB( 9)"36 37 38 39 44 45 46 47
     ": PRINT : PRINT  TAB( 9)"52 53 54 55 60 61 6
     2 63"
85  PRINT : PRINT : PRINT : PRINT : PRINT "Is the
     number you picked in this set?  Look careful
     ly!"
90  INPUT B$: IF B$ = "Y" THEN  GOSUB 1200
91  IF B$ = "YES" THEN  GOSUB 1200
93  HOME
95  PRINT : PRINT : PRINT : PRINT : PRINT  TAB( 9)
     "8  9  10 11 12 13 14 15": PRINT : PRINT  TAB(
     9)"24 25 26 27 28 29 30 31"
100 PRINT : PRINT  TAB( 9)"40 41 42 43 44 45 46 4
     7": PRINT : PRINT  TAB( 9)"56 57 58 59 60 61
     62 63"
105 PRINT : PRINT : PRINT : PRINT : PRINT "Is the
     number you picked in this set?"
110 INPUT C$: IF C$ = "Y" THEN  GOSUB 1300
113 IF C$ = "YES" THEN  GOSUB 1300
115 HOME : PRINT : PRINT : PRINT : PRINT : PRINT
     TAB( 9)"16 17 18 19 20 21 22 23": PRINT : PRINT
     TAB( 9)"24 25 26 27 28 29 30 31"
120 PRINT : PRINT  TAB( 9)"48 49 50 51 52 53 54 5
     5": PRINT : PRINT  TAB( 9)"56 57 58 59 60 61
     62 63"
125 PRINT : PRINT : PRINT : PRINT : PRINT "Is the
     number you picked in this set?"
130 INPUT D$: IF D$ = "Y" THEN  GOSUB 1400
133 IF D$ = "YES" THEN  GOSUB 1400
135 HOME : PRINT : PRINT : PRINT : PRINT : PRINT
     TAB( 9)"32 33 34 35 36 37 38 39": PRINT : PRINT
     TAB( 9)"40 41 42 43 44 45 46 47"
140 PRINT : PRINT  TAB( 9)"48 49 50 51 52 53 54 5
     5": PRINT : PRINT  TAB( 9)"56 57 58 59 60 61
     62 63"
```

```
145  PRINT : PRINT : PRINT : PRINT : PRINT "Is the
      number you picked in this set?"
150  INPUT E$: IF E$ = "Y" THEN  GOSUB 1500
153  IF E$ = "YES" THEN  GOSUB 1500
154  GOSUB 1700: PRINT ""
155  GOSUB 1700: PRINT "Let me guess. your number
     was......."Z
156  GOSUB 1000: PRINT : PRINT : PRINT "
         Was I right?": INPUT R$
157  IF R$ = "N" THEN  GOSUB 2000
158  IF R$ = "NO" THEN  GOSUB 2000
164  HOME : PRINT  TAB( 15)"TRY ANOTHER?(Y/N)": INPUT
     J$
165  IF J$ = "Y" THEN 29
167  IF J$ = "YES" THEN 29
170  HOME : PRINT : PRINT : PRINT : PRINT  TAB( 19
     )">BYE<": GOSUB 1000
175  HOME : SPEED= 255: END
1000  FOR S = 1 TO 4000: NEXT S: RETURN
1100  LET Z = Z + 2: RETURN
1200  LET Z = Z + 4: RETURN
1300  LET Z = Z + 8: RETURN
1400  LET Z = Z + 16: RETURN
1500  LET Z = Z + 32: RETURN
1700  FOR Y = 1 TO 1000: NEXT Y: RETURN
2000  PRINT ""
2050  HOME : PRINT : PRINT : PRINT : HTAB (19): PRINT
     "OOPS!!"
2100  GOSUB 1000
2200  RETURN
```

# Canyon Runner

Vic Neale
Apple Version by Kevin Martin

*In "Canyon Runner" you are a pilot on a mission through a very perilous canyon. You must survive this test. The only way to do so is by maneuvering your tiny plane through the endlessly scrolling canyon.*

The object of "Canyon Runner" is to navigate through a twisting canyon while trying to shoot down an opponent. But beware, your opponent will also be shooting at you.

The program is written in two parts and requires game paddles and a disk drive. Program 1 is all machine language and must be entered with the built-in monitor (see your Apple manual if you are unsure of how this is done). After Program 1 is entered, BSAVE it with the filename CANYON.ML using a starting address of $6000 and length of $923. Once you have saved Program 1, enter the loader program (Program 2) and save it. To play the game, load and run Program 2, which will load in and check the machine language from Program 1, then start the game.

When the program is run, you will be presented with a screen containing many options. Each player can choose his own level of difficulty. Player 1 increases or decreases his difficulty level by pressing X or Z, respectively, while player 2 uses the left and right arrow keys to accomplish this. If you wish to play alone, press S for the solo option.

There are two types of shots. If you press A at the start of the game, you will be playing with altitude bombs which explode at the altitude at which they are fired.

The second type of bomb, the detonation bomb (chosen at the start of the game by pressing D), will change its altitude as you change the altitude of your plane, so you can continue to adjust your altitude to the altitude of your opponent after the shot is fired.

The overall width of the canyon can be adjusted by pressing the numbers from 1 to 3. The higher the number picked, the narrower the canyon.

Once the options have been chosen, you can start the game by pressing both paddle buttons simultaneously. The planes are moved from left to right using the paddle. Altitude

is changed using the keyboard. Player 1 can increase or decrease his altitude with the A and Z keys. Player 2 can make his plane climb using the semicolon and descend using the period. An altitude reading for each player is displayed at the bottom of the screen.

At any time during the game, you may fire a bomb at your opponent by pressing the paddle fire button. A countdown reading will appear at the bottom of the screen showing the time until impact.

## Program 1. Canyon Runner, ML

```
6000- 4C 2B 61 A0 85 A0 A0 CD
6008- C9 A0 B0 80 F0 D2 A0 A0
6010- A9 A0 A0 85 A3 C9 C8 E5
6018- D0 CC C1 D9 C5 D2 A0 B1
6020- A0 A0 A0 C1 CC D4 A0 A0
6028- A0 A0 A0 A0 A0 A0 A0 A0
6030- A0 A0 C1 CC D4 A0 A0 A0
6038- D0 CC C1 D9 C5 D2 A0 B2
6040- 8D A0 D0 CC C1 CE C5 D3
6048- A0 A0 A0 A0 A0 A0 A0 A0
6050- A0 A0 A0 A0 A0 A0 A0 A0
6058- A0 A0 A0 A0 A0 A0 A0 A0
6060- A0 A0 D0 CC C1 CE C5 D3
6068- 8D C3 CF D5 CE D4 A0 C4
6070- CF D7 CE A0 A0 A0 A0 A0
6078- A0 A0 A0 A0 A0 A0 A0 A0
6080- A0 A0 A0 C3 CF D5 CE D4
6088- A0 C4 CF D7 CE 00 A0 A0
6090- A0 A0 A0 A0 A0 A0 A0 C7
6098- C1 CD C5 A0 CF D6 C5 D2
60A0- AD D0 D2 C5 D3 D3 A0 D2
60A8- C5 D4 D5 D2 CE A0 A0 A0
60B0- A0 A0 A0 A0 A0 A0 A0 D3
60B8- C1 C4 A0 A0 A0 A0 A0 A0
60C0- D0 D2 C5 D3 D3 A0 C2 CF
60C8- D4 C8 A0 C2 D5 D4 D4 CF
60D0- CE D3 A0 D4 CF A0 D3 D4
60D8- C1 D2 D4 8D CC C5 D6 C5
60E0- CC BA A0 A0 A0 A0 A0 A0
60E8- A0 A0 A0 A0 A0 A0 A0 A0
60F0- A0 A0 A0 A0 A0 A0 A0 A0
60F8- A0 A0 A0 A0 C7 C1 CD C5
6100- BA 8D D0 CC C1 D9 C5 D2
6108- A0 B1 A0 A0 A0 A0 A0 A0
6110- A0 C4 C9 C6 C6 C9 C3 D5
6118- CC D4 D9 A0 A0 A0 A0 A0
```

```
6120- A0 A0 D0 CC C1 D9 C5 D2
6128- A0 B2 00 A9 E1 8D 0B 60
6130- A9 7A 8D 0C 60 20 71 61
6138- 20 E2 F3 AD F7 F6 20 F4
6140- F3 A9 00 8D 04 60 8D 05
6148- 60 20 AA 62 20 6C 67 20
6150- 21 65 20 24 66 20 0A 67
6158- 20 70 66 AD 3F 03 C9 01
6160- F0 03 20 7B 63 20 93 64
6168- 20 15 64 20 45 63 4C 4C
6170- 61 20 E2 F3 AD F7 F6 20
6178- F4 F3 A9 14 85 22 20 58
6180- FC A9 01 8D 3F 03 A9 04
6188- 8D 3D 03 8D 3E 03 A9 50
6190- 8D 3C 03 A2 00 BD BA 60
6198- F0 06 20 F0 FD E8 D0 F5
61A0- A9 07 85 24 A9 15 85 25
61A8- 20 22 FC AD 3C 03 C9 20
61B0- D0 05 A9 B1 4C C2 61 C9
61B8- 38 D0 05 A9 B2 4C C2 61
61C0- A9 B3 20 F0 FD A9 26 85
61C8- 24 AE 3F 03 BD B6 60 20
61D0- F0 FD A9 8D 20 F0 FD A9
61D8- 8D 20 F0 FD A9 04 85 24
61E0- AD 3D 03 18 69 B0 20 F0
61E8- FD A9 23 85 24 AD 3E 03
61F0- 18 69 B0 20 F0 FD AD 00
61F8- C0 10 11 8D 10 C0 29 7F
6200- C9 41 D0 0B A9 02 8D 3F
6208- 03 4C A0 61 4C 91 62 C9
6210- 44 D0 08 A9 03 8D 3F 03
6218- 4C A0 61 C9 53 D0 08 A9
6220- 01 8D 3F 03 4C A0 61 C9
6228- 31 D0 08 A9 20 8D 3C 03
6230- 4C A0 61 C9 32 D0 08 A9
6238- 38 8D 3C 03 4C A0 61 C9
6240- 33 D0 08 A9 50 8D 3C 03
6248- 4C A0 61 C9 5A D0 0B CE
6250- 3D 03 D0 03 EE 3D 03 4C
6258- A0 61 C9 58 D0 10 EE 3D
6260- 03 AD 3D 03 C9 0A D0 03
6268- CE 3D 03 4C A0 61 C9 08
6270- D0 0B CE 3E 03 D0 03 EE
6278- 3E 03 4C A0 61 C9 15 D0
6280- 10 EE 3E 03 AD 3E 03 C9
6288- 0A D0 03 CE 3E 03 4C A0
6290- 61 AD 62 C0 30 03 4C A0
6298- 61 AD 61 C0 30 03 4C A0
62A0- 61 A9 02 8D 0F 60 8D 10
```

```
62A8- 60 60 A9 14 85 22 20 58
62B0- FC A2 00 BD 18 60 F0 07
62B8- 20 F0 FD E8 4C B3 62 A9
62C0- 14 8D 11 60 A9 0A 8D 12
62C8- 60 A9 00 8D 15 60 8D 16
62D0- 60 A9 19 8D 07 60 A9 00
62D8- 8D 08 60 A9 A5 8D 09 60
62E0- A9 00 8D 0A 60 A9 01 85
62E8- E7 A9 70 85 E9 A9 00 85
62F0- E8 A9 0A 8D 0D 60 A9 00
62F8- 8D 0E 60 20 28 63 EE 0D
6300- 60 AD 0D 60 38 E9 0A CD
6308- 3C 03 D0 EF A9 96 8D 0D
6310- 60 A9 00 8D 0E 60 20 28
6318- 63 EE 0D 60 AD 0D 60 38
6320- E9 96 CD 3C 03 D0 EF 60
6328- A2 00 20 F0 F6 A9 00 AE
6330- 0D 60 AC 0E 60 20 11 F4
6338- AD 0D 60 AE 0E 60 A0 AC
6340- 20 3A F5 60 60 AD 15 60
6348- 18 6D 16 60 C9 00 F0 0D
6350- A2 0F AD 30 C0 A9 04 20
6358- A8 FC CA D0 F5 60 A9 10
6360- 8D 17 60 A0 01 A2 01 A9
6368- 50 20 A8 FC AD 30 C0 E8
6370- D0 FD 88 D0 F0 CE 17 60
6378- D0 E9 60 AD 61 C0 30 6F
6380- AD 62 C0 30 7D AD 15 60
6388- F0 26 CE 15 60 D0 21 AD
6390- 13 60 CD 12 60 90 0A AD
6398- 13 60 38 ED 12 60 4C A8
63A0- 63 AD 12 60 38 ED 13 60
63A8- CD 3D 03 B0 03 4C BD 65
63B0- AD 16 60 F0 26 CE 16 60
63B8- D0 21 AD 14 60 CD 11 60
63C0- 90 0A AD 14 60 38 ED 11
63C8- 60 4C D3 63 AD 11 60 38
63D0- ED 14 60 CD 3E 03 B0 03
63D8- 4C 7B 65 AD 3F 03 C9 03
63E0- D0 0C AD 11 60 8D 13 60
63E8- AD 12 60 8D 14 60 60 AD
63F0- 15 60 D0 8C A9 0A 8D 15
63F8- 60 AD 11 60 8D 13 60 4C
6400- 80 63 AD 16 60 D0 0B A9
6408- 0A 8D 16 60 AD 12 60 8D
6410- 14 60 4C 85 63 A9 15 85
6418- 25 20 22 FC A9 03 85 24
6420- AD 0F 60 18 69 B0 20 F0
6428- FD A9 0B 85 24 AD 11 60
6430- C9 0A B0 05 A9 A0 20 F0
```

```
6438- FD A9 ØØ AE 11 6Ø 2Ø 24
6440- ED A9 1A 85 24 AD 12 6Ø
6448- C9 ØA BØ Ø5 A9 AØ 2Ø FØ
6450- FD A9 ØØ AE 12 6Ø 2Ø 24
6458- ED A9 23 85 24 AD 1Ø 6Ø
6460- 18 69 BØ 2Ø FØ FD A9 8D
6468- 2Ø FØ FD A9 8D 2Ø FØ FD
6470- A9 ØB 85 24 A9 ØØ AE 15
6478- 6Ø 2Ø 24 ED A9 AØ 2Ø FØ
6480- FD A9 25 85 24 A9 ØØ AE
6488- 16 6Ø 2Ø 24 ED A9 AØ 2Ø
6490- FØ FD 6Ø AD ØØ CØ 1Ø 49
6498- 29 7F C9 41 FØ ØF C9 5A
64A0- FØ 18 C9 3B FØ 21 C9 2E
64A8- FØ 2A 4C E1 64 AD 11 6Ø
64B0- C9 1E FØ 2A EE 11 6Ø 4C
64B8- DE 64 AD 11 6Ø C9 Ø1 FØ
64C0- 1D CE 11 6Ø 4C DE 64 AD
64C8- 12 6Ø C9 1E FØ 1Ø EE 12
64D0- 6Ø 4C DE 64 AD 12 6Ø C9
64D8- Ø1 FØ Ø3 CE 12 6Ø 2C 1Ø
64E0- CØ 6Ø A9 A2 A2 ØØ 8E E3
64E8- 64 8E E2 64 A2 7F AD 7Ø
64F0- CØ AD 64 CØ 29 8Ø ØA 2A
64F8- 6D E2 64 8D E2 64 AD 65
6500- CØ 29 8Ø ØA 2A 6D E3 64
6508- 8D E3 64 CA DØ E3 A9 7F
6510- 38 ED E2 64 8D E2 64 A9
6518- 7F 38 ED E3 64 8D E3 64
6520- 6Ø 2Ø E4 64 AD E2 64 C9
6528- 46 9Ø 17 C9 64 BØ Ø3 4C
6530- 4F 65 A2 Ø3 CE Ø7 6Ø DØ
6538- Ø3 CE Ø8 6Ø CA DØ F5 4C
6540- 4F 65 A2 Ø3 EE Ø7 6Ø DØ
6548- Ø3 EE Ø8 6Ø CA DØ F5 AD
6550- E3 64 C9 46 9Ø 17 C9 64
6558- BØ Ø3 4C 7A 65 A2 Ø3 CE
6560- Ø9 6Ø DØ Ø3 CE ØA 6Ø CA
6568- DØ F5 4C 7A 65 A2 Ø3 EE
6570- Ø9 6Ø DØ Ø3 EE ØA 6Ø CA
6578- DØ F5 6Ø 68 68 AD ØF 6Ø
6580- C9 ØØ FØ Ø9 CE ØF 6Ø 2Ø
6588- 96 65 4C 38 61 2Ø 96 65
6590- 2Ø FF 65 4C 35 61 A9 5Ø
6598- AE Ø7 6Ø AC Ø8 6Ø 2Ø 11
65A0- F4 A2 Ø2 2Ø 3Ø F7 A6 1A
65A8- A4 1B A9 ØØ 2Ø 5D F6 A2
65B0- ØØ 88 DØ FD E8 EØ ØA DØ
65B8- F8 2Ø 5E 63 6Ø 68 68 AD
65C0- 1Ø 6Ø C9 ØØ FØ Ø9 CE 1Ø
```

```
65C8- 60 20 D8 65 4C 38 61 20
65D0- D8 65 20 FF 65 4C 35 61
65D8- A9 50 AE 09 60 AC 0A 60
65E0- 20 11 F4 A2 02 20 30 F7
65E8- A6 1A A4 1B A9 00 20 5D
65F0- F6 A2 00 88 D0 BB E8 E0
65F8- 0A D0 F8 20 5E 63 60 A9
6600- 17 85 25 20 22 FC A9 00
6608- 85 24 A2 00 BD 8E 60 20
6610- F0 FD E8 E0 27 D0 F5 2C
6618- 10 C0 AD 00 C0 10 FB C9
6620- 8D D0 F4 60 A9 4B AE 07
6628- 60 AC 08 60 20 11 F4 A2
6630- 01 20 30 F7 A6 1A A4 1B
6638- A9 00 20 5D F6 A5 EA C9
6640- 3A F0 03 4C 7B 65 AD 3F
6648- 03 C9 01 F0 22 A9 4B AE
6650- 09 60 AC 0A 60 20 11 F4
6658- A2 01 20 30 F7 A6 1A A4
6660- 1B A9 00 20 5D F6 A5 EA
6668- C9 3A F0 03 4C BD 65 60
6670- A9 4B AE 07 60 AC 08 60
6678- 20 11 F4 A2 01 20 30 F7
6680- A6 1A A4 1B A9 00 20 5D
6688- F6 AD 3F 03 C9 01 F0 19
6690- A9 4B AE 09 60 AC 0A 60
6698- 20 11 F4 A2 01 20 30 F7
66A0- A6 1A A4 1B A9 00 20 5D
66A8- F6 60 AD 0B 60 0A 0A 38
66B0- 6D 0B 60 8D 0B 60 AD 0C
66B8- 60 0A 0A 38 6D 0C 60 8D
66C0- 0C 60 60 AD 0B 60 C9 55
66C8- 90 07 C9 AC B0 0E 4C E6
66D0- 66 AD 04 60 F0 10 CE 04
66D8- 60 4C E6 66 AD 04 60 C9
66E0- 31 F0 03 EE 04 60 AD 0C
66E8- 60 C9 55 90 07 C9 AC B0
66F0- 0E 4C 09 67 AD 05 60 F0
66F8- 10 CE 05 60 4C 09 67 AD
6700- 05 60 C9 31 F0 03 EE 05
6708- 60 60 A9 A8 8D 06 60 20
6710- AA 66 20 C3 66 A2 00 20
6718- F0 F6 AD 04 60 18 69 0A
6720- AA AD 06 60 A0 00 20 11
6728- F4 AC 06 60 AD 04 60 18
6730- 69 0A 6D 3C 03 A2 00 20
6738- 3A F5 AD 05 60 18 69 96
6740- AA A0 00 AD 06 60 20 11
6748- F4 AC 06 60 AD 05 60 18
```

```
6750- 69 96 6D 3C 03 90 05 A2
6758- 01 4C 5E 67 A2 00 20 3A
6760- F5 EE 06 60 AC 06 60 C0
6768- AD D0 A7 60 A0 00 B9 A1
6770- 67 85 08 B9 62 68 85 09
6778- C8 C8 C8 C8 C8 B9 A1 67
6780- 85 06 B9 62 68 85 07 88
6788- 88 88 88 8C 03 60 A0 01
6790- B1 06 91 08 C8 C0 27 D0
6798- F7 AC 03 60 C0 AD D0 CE
67A0- 60 00 00 00 00 00 00 00
67A8- 00 80 80 80 80 80 80 80
67B0- 80 00 00 00 00 00 00 00
67B8- 00 80 80 80 80 80 80 80
67C0- 80 00 00 00 00 00 00 00
67C8- 00 80 80 80 80 80 80 80
67D0- 80 00 00 00 00 00 00 00
67D8- 00 80 80 80 80 80 80 80
67E0- 80 28 28 28 28 28 28 28
67E8- 28 A8 A8 A8 A8 A8 A8 A8
67F0- A8 28 28 28 28 28 28 28
67F8- 28 A8 A8 A8 A8 A8 A8 A8
6800- A8 28 28 28 28 28 28 28
6808- 28 A8 A8 A8 A8 A8 A8 A8
6810- A8 28 28 28 28 28 28 28
6818- 28 A8 A8 A8 A8 A8 A8 A8
6820- A8 50 50 50 50 50 50 50
6828- 50 D0 D0 D0 D0 D0 D0 D0
6830- D0 50 50 50 50 50 50 50
6838- 50 D0 D0 D0 D0 D0 D0 D0
6840- D0 50 50 50 50 50 50 50
6848- 50 D0 D0 D0 D0 D0 D0 D0
6850- D0 50 50 50 50 50 50 50
6858- 50 D0 D0 D0 D0 D0 D0 D0
6860- D0 00 20 24 28 2C 30 34
6868- 38 3C 20 24 28 2C 30 34
6870- 38 3C 21 25 29 2D 31 35
6878- 39 3D 21 25 29 2D 31 35
6880- 39 3D 22 26 2A 2E 32 36
6888- 3A 3E 22 26 2A 2E 32 36
6890- 3A 3E 23 27 2B 2F 33 37
6898- 3B 3F 23 27 2B 2F 33 37
68A0- 3B 3F 20 24 28 2C 30 34
68A8- 38 3C 20 24 28 2C 30 34
68B0- 38 3C 21 25 29 2D 31 35
68B8- 39 3D 21 25 29 2D 31 35
68C0- 39 3D 22 26 2A 2E 32 36
68C8- 3A 3E 22 26 2A 2E 32 36
68D0- 3A 3E 23 27 2B 2F 33 37
```

```
68D8- 3B 3F 23 27 2B 2F 33 37
68E0- 3B 3F 20 24 28 2C 30 34
68E8- 38 3C 20 24 28 2C 30 34
68F0- 38 3C 21 25 29 2D 31 35
68F8- 39 3D 21 25 29 2D 31 35
6900- 39 3D 22 26 2A 2E 32 36
6908- 3A 3E 22 26 2A 2E 32 36
6910- 3A 3E 23 27 2B 2F 33 37
6918- 3B 3F 23 27 2B 2F 33 37
6920- 3B 3F 20 00
```

## Program 2. BASIC Loader for Canyon Runner

```
10  PRINT  CHR$ (4);"BLOAD CANYON.ML"
90 CK = 0
100  FOR I = 28672 TO 28761: READ A:CK = CK + A: POKE
     I,A: NEXT
104  IF CK <  > 4288 THEN  PRINT "ERROR IN DATA": END

105 CK = 0
110  FOR I = 24576 TO 26915:CK = CK +  PEEK (I): NEXT

120  IF CK <  > 265976 THEN  PRINT "ERROR IN MACHI
     NE LANGUAGE"
130  CALL 24576
200  DATA  2,0,6,0,36,0,36,45
210  DATA  45,37,36,36,60,44,45,45
220  DATA  53,55,54,54,46,45,45,54
230  DATA  63,63,63,54,54,63,36,36
240  DATA  63,63,39,0,12,12,12,12
250  DATA  12,12,12,12,12,12,12,12
260  DATA  12,12,150,146,58,63,63,255
270  DATA  63,63,63,4,64,24,64,24
280  DATA  21,21,21,21,21,21,149,201
290  DATA  14,14,14,14,14,14,223,219
300  DATA  35,36,36,36,32,36,36,36
310  DATA  0,255,0,0,255,255,0,0
```

# Chapter 2

# Education

# Introduction

Your Apple never tires of quizzes and drills, and that makes it an excellent teacher. The teaching programs in this chapter turn your computer into a sophisticated educational tool, making learning more fun than ever before.

Preschoolers will delight in Garold R. Stone's "Letter and Number Play," a graphically exciting introduction to letters and numbers. Older children will enjoy Steve Hamilton's "First Math" and Soori Sivakumaran's "Snertle," programs that teach fundamental mathematics skills.

"Chemistry Lab," by Joanne Davis, will be of special interest to teachers. It uses animated graphics to illustrate basic concepts of chemistry, and it features an easy-to-use menu. And teachers and students alike will enjoy William Loercher's "Crosswords," a program that creates simple crossword puzzles from any given list of words.

Alan McCright's "Typing Teacher" will appeal to anyone who is learning to type. It is particularly valuable to computer users who would like to learn to type in programs.

Harvey B. Herman's "Memory Trainer" uses proven techniques to help students of all ages improve their memory. It's just the thing if you have a hard time recalling those names, dates, and phone numbers.

Finally, Rob Smythe's "Oscilloscope" will let you draw complex waveforms. Developed as a teaching tool, it is also an excellent demonstration of both the graphics and analytical capabilities of your Apple computer.

# Letter and Number Play

Garold R. Stone
Apple Translation by Patrick Parrish

*Even very young children can benefit from the educational power of the computer, as this program shows. It's designed to teach letters and numbers to preschoolers.*

This program was written to help my two-year-old son learn letters and numbers.

When the program starts, it's in the "alphabet" mode. A large letter A appears in the middle of the screen, while a small reverse video A appears near the bottom. Each time the child presses the space bar, the next letter in the alphabet replaces the previous one in the middle of the screen and the new letter is added to an alphabetic sequence at the bottom.

Initially, the program would only display letters in alphabetical order. But one day my son asked to see the Q when we were only up to D. As a result, I expanded the program so that he could put any letter at the top of the screen by pressing its key on the keyboard. After pressing any desired letter keys, he can continue through the alphabet simply by pressing the space bar again.

At any time you can press CTRL-L (the L stands for "letters") to start over with the letter A. If you get to the end of the alphabet, the string of letters at the bottom of the screen flashes ten times—a good opportunity to make a big deal out of the accomplishment and praise the child.

To round out the program, I added numbers too. To switch to numbers, press CTRL-N (for numbers). Pressing the space bar displays the next higher number in large print in the middle of the screen. Numbers greater than 9999 will not fit on the screen. Pressing any of the digits (0–9) displays that digit in large print at the top of the screen. To start counting over at one, press CTRL-N again. To return to the alphabet, press CTRL-L once again.

## Relaxed Learning

There are several ways that a parent can use this program. In my case, my son likes to sit on my lap and press the space bar to see the letters or numbers. I say the name of the figure that

he pressed, and he often repeats it after me. I may even ask him questions like "What is the first letter of the alphabet?" or "Can you find the *A*?" He can guess the name of the next letter or number or try to find a character on the keyboard. Sometimes he just wants to see some favorite letters and touch them on the screen.

Sessions with the program are rarely more than five minutes long. It's all quite relaxed, but the benefits are unmistakable: The child is learning the names, shapes, and order of the letters and numbers.

## Letter and Number Play

```
10  LOMEM: 16384
20  DIM L$(26)
30  HOME
40  VTAB 3: PRINT "FOR THE SUPERVISING ADULT:"
60  PRINT : PRINT "  PRESS LETTER KEYS OR <SPACE>
     TO PLAY."
70  PRINT "  <CONTROL> & <L> RESETS ALPHABET TO 'A
    '."
80  PRINT : PRINT : PRINT "PRESS <CONTROL> AND <N>
     FOR THE NUMBERS:"
90  PRINT : PRINT "  PRESS NUMBER KEYS OR <SPACE>
     TO PLAY."
110  REM   STORE LETTER COORDINATES IN A
120  DIM A(26,20): DIM N(10,20)
130  REM  SET UP LETTERS
140  FOR I = 1 TO 26
150  FOR J = 1 TO 20
160  READ A(I,J)
170  NEXT J: NEXT I
180  GOSUB 1370: REM  SET UP NUMBERS
190  PRINT : PRINT : PRINT "PRESS <SPACE> TO CONTI
    NUE, '/' TO STOP"
200  GET A$
210  GOSUB 2050
220  REM  LETTERS
230 L$ = "":L = 1: GOSUB 1170: GOSUB 1270
240  GET A$
250  GOSUB 2050
260  IF A$ =  CHR$ (14) THEN  GOSUB 1750: GOTO 230
    : REM  NUMBERS
270  IF A$ =  CHR$ (12) THEN 230
280  IF A$ = " " THEN L = L + 1:B = 0: IF L > 26 THEN
    230
290  IF A$ = " " THEN  IF L > 26 THEN 230
```

```
300  IF A$ = " " THEN  GOSUB 1170: GOSUB 1270: GOTO
     240
310  IF B <  > 0 THEN T = L:L = B - 64: HCOLOR= 0:
     Y7 = 30: GOSUB 1190:L = T: HCOLOR= 3
320 A =  ASC (A$):T = L: REM  REMEMBER L
330  IF A >  = 65 AND A <  = 90 THEN L = A - 64:B =
     A:Y7 = 30: GOSUB 1190
340 L = T
350  GOTO 240
360  TEXT : HOME : END
370  REM  LETTERS
380  REM  ---A---
390  DATA  0,40,13,0,13,0,26,40,6,21
400  DATA  20,21,-1,-1,-1,-1,-1,-1,-1,-1
410  REM  ---B---
420  DATA  0,0,0,40,0,1,25,1,25,1
430  DATA  25,39,0,39,25,39,0,20,25,20
440  REM  ---C---
450  DATA  25,0,0,0,0,0,0,40,0,40
460  DATA  25,40,-1,-1,-1,-1,-1,-1,-1,-1
470  REM  ---D---
480  DATA  0,0,0,40,0,1,25,1,25,1
490  DATA  25,39,25,39,0,39,-1,-1,-1,-1
500  REM  ---E---
510  DATA  25,40,0,40,0,40,0,0,0,0
520  DATA  25,0,0,20,13,20,-1,-1,-1,-1
530  REM  ---F---
540  DATA  0,40,0,0,0,0,25,0,0,20
550  DATA  13,20,-1,-1,-1,-1,-1,-1,-1,-1
560  REM  ---G---
570  DATA  25,0,0,0,0,0,0,40,0,40
580  DATA  25,40,25,40,25,20,25,20,15,20
590  REM  ---H---
600  DATA  0,0,0,40,25,0,25,40,0,20
610  DATA  25,20,-1,-1,-1,-1,-1,-1,-1,-1
620  REM  ---I---
630  DATA  0,0,24,0,0,40,24,40,12,0
640  DATA  12,40,-1,-1,-1,-1,-1,-1,-1,-1
650  REM  ---J---
660  DATA  25,0,25,40,25,40,0,40,0,40
670  DATA  0,30,-1,-1,-1,-1,-1,-1,-1,-1
680  REM  ---K---
690  DATA  0,0,0,40,0,20,25,0,0,20
700  DATA  25,40,-1,-1,-1,-1,-1,-1,-1,-1
710  REM  ---L---
720  DATA  0,0,0,40,0,40,25,40,-1,-1
730  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
740  REM  ---M---
750  DATA  0,0,0,40,0,0,13,20,13,20
```

```
760  DATA  26,0,26,0,26,40,-1,-1,-1,-1
780  DATA  0,40,0,0,0,0,25,40,25,40
790  DATA  25,0,-1,-1,-1,-1,-1,-1,-1,-1
800  REM  ---O---
810  DATA  0,0,25,0,25,0,25,40,25,40
820  DATA  0,40,0,40,0,0,-1,-1,-1,-1
830  REM  ---P---
840  DATA  0,40,0,0,0,0,25,0,25,0
850  DATA  25,20,25,20,0,20,-1,-1,-1,-1
860  REM  ---Q---
870  DATA  0,0,25,0,25,0,25,40,25,40
880  DATA  0,40,0,40,0,0,20,35,30,45
890  REM  ---R---
900  DATA  0,40,0,0,0,0,25,0,25,0
910  DATA  25,20,25,20,0,20,10,20,25,40
920  REM  ---S---
930  DATA  25,0,0,0,0,0,0,20,0,20
940  DATA  25,20,25,20,25,40,25,40,0,40
950  REM  ---T---
960  DATA  0,0,25,0,13,0,13,40,-1,-1
970  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
980  REM  ---U---
990  DATA  0,0,0,40,0,40,25,40,25,40
1000  DATA  25,0,-1,-1,-1,-1,-1,-1,-1,-1
1010  REM  ---V---
1020  DATA  0,0,13,40,13,40,25,0,-1,-1
1030  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1040  REM  ---W---
1050  DATA  0,0,5,40,5,40,13,0,13,0
1060  DATA  21,40,21,40,26,0,-1,-1,-1,-1
1070  REM  ---X---
1080  DATA  0,0,25,40,0,40,25,0,-1,-1
1090  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1100  REM  ---Y---
1110  DATA  0,0,13,20,13,20,26,0,13,20
1120  DATA  13,40,-1,-1,-1,-1,-1,-1,-1,-1
1130  REM  ---Z---
1140  DATA  0,0,25,0,25,0,0,40,0,40
1150  DATA  25,40,-1,-1,-1,-1,-1,-1,-1,-1
1160  REM  PRINT LETTER
1170  HOME :Y7 = 90
1180  HGR : HCOLOR= 3
1190 X7 = 130
1200  FOR J = 1 TO 20 STEP 4
1210 X1 = A(L,J):Y1 = A(L,J + 1):X2 = A(L,J + 2):Y
     2 = A(L,J + 3)
1220  IF X1 < 0 THEN 1240
1230  HPLOT X1 + X7,Y1 + Y7 TO X2 + X7,Y2 + Y7
1240  NEXT J
1250  RETURN
```

```
1260  REM  PRINT ALPHABET
1270 L$ = L$ +  CHR$ (L + 64)
1280  VTAB 22: HTAB 6: INVERSE : PRINT L$: NORMAL

1290  FOR I = 1 TO 250: NEXT
1300  IF L = 26 THEN  GOSUB 1330
1310  RETURN
1320  REM  FLASH ALPHABET
1330  VTAB 22: HTAB 6: FLASH : PRINT L$
1340  FOR I = 1 TO 3000: NEXT
1350  VTAB 22: HTAB 6: NORMAL : PRINT L$
1360  RETURN
1370  REM   STORE DIGIT COORDINATES IN N
1380  FOR I = 0 TO 9
1390  REM  SET UP A DIGIT
1400  FOR J = 0 TO 19
1410  READ N(I,J)
1420  NEXT J
1430  NEXT I: RETURN
1440  REM  DIGITS
1450  REM  ---0---
1460  DATA  0,0,20,0,20,0,20,40,20,40
1470  DATA  0,40,0,40,0,0,-1,-1,-1,-1
1480  REM  ---1---
1490  DATA  5,10,13,0,13,0,13,40,0,40
1500  DATA  26,40,-1,-1,-1,-1,-1,-1,-1,-1
1510  REM  ---2---
1520  DATA  0,10,12,0,12,0,24,10,24,10
1530  DATA  0,40,0,40,25,40,-1,-1,-1,-1
1540  REM  ---3---
1550  DATA  0,0,20,0,20,0,20,40,20,40
1560  DATA  0,40,0,20,20,20,-1,-1,-1,-1
1570  REM  ---4---
1580  DATA  20,0,0,35,0,35,25,35,20,0
1590  DATA  20,40,-1,-1,-1,-1,-1,-1,-1,-1
1600  REM  ---5---
1610  DATA  19,0,5,0,5,0,0,19,0,19
1620  DATA  20,19,20,19,20,40,20,40,0,40
1630  REM  ---6---
1640  DATA  2,0,0,20,0,20,22,20,22,20
1650  DATA  22,40,22,40,0,40,0,40,0,20
1660  REM  ---7---
1670  DATA  0,0,25,0,25,0,0,40,-1,-1
1680  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1690  REM  ---8---
1700  DATA  0,0,0,40,0,40,20,40,20,40
1710  DATA  20,0,0,20,20,20,20,0,0,0
1720  REM  ---9---
1730  DATA  0,0,22,0,22,0,22,20,22,20
1740  DATA  0,20,0,20,0,0,22,20,20,40
```

```
1750 N1 = 1: GOSUB 2040
1760  GET A$: GOSUB 2050
1770  IF A$ = " " THEN N1 = N1 + 1:C2 = 0: GOSUB 2
     040: GOTO 1760
1780  IF C2 = 1 THEN X = N2: HCOLOR= 0:Y7 = 30:X7 =
     135: GOSUB 1980: HCOLOR= 3
1790 T = N1:N1 =  ASC (A$) - 48:N2 = N1:C2 = 1: IF
     N1 < 0 OR N1 > 9 THEN N2 = 1
1800  IF N1 >  = 0 AND N1 < 10 THEN C5 = 1: GOSUB
     1850
1810 N1 = T
1820  IF A$ =  CHR$ (14) THEN N1 = 1: GOSUB 1850
1830  IF A$ =  CHR$ (12) THEN  RETURN
1840  GOTO 1760
1850  REM
1860 P = 1:F$ =  STR$ (N1)
1870 X =  VAL ( MID$ (F$,P,1))
1880  IF  VAL (A$) = N1 THEN Y7 = 30: GOSUB 1930: GOTO
     1900
1890  GOSUB 1920
1900 P = P + 1: IF P <  =  LEN (F$) THEN 1870
1910  RETURN
1920 Y7 = 90
1930 X7 = 135: FOR Q = 1 TO  LEN (F$):X7 = X7 - 33
     : NEXT
1940  FOR Q = 1 TO P:X7 = X7 + 33: NEXT
1950  IF P > 1 OR C5 = 1 THEN 1980
1970  HGR : HCOLOR= 3
1980  FOR J = 0 TO 19 STEP 4
1990 X1 = N(X,J):Y1 = N(X,J + 1):X2 = N(X,J + 2):Y
     2 = N(X,J + 3)
2000  IF X1 < 0 THEN 2030
2010  HPLOT X1 + X7,Y1 + Y7 TO X2 + X7,Y2 + Y7
2020  NEXT
2030 C5 = 0: RETURN
2040  HOME : GOSUB 1850: RETURN
2050  IF A$ = "/" THEN 360
2060  RETURN
```

# Snertle

Soori Sivakumaran
Apple Translation by Chris Poer

*By making simple selections from a menu, a child can change this arithmetic drill to fit his or her own tutoring needs. It features a smiling turtle and bold graphics sure to catch the young child's eye.*

"Snertle" is designed to help teach children the fundamentals of addition, subtraction, and multiplication. A turtle named Snertle is drawn on the screen to give encouragement and assistance to the player.

## An Individual Challenge

Snertle allows children to tailor math problems to fit their individual abilities and weaknesses. It first asks the child to select addition, subtraction, or multiplication problems. If addition or subtraction is selected, the child is then asked to specify the largest and smallest numbers (between 0 and 99) to be used in creating the problems.

If multiplication is chosen, the child may decide to practice a specific multiplication table or solve problems created randomly using numbers from 0 through 14. For example, if the 12 times table is selected, then one number in each question created will always be 12. The other number will be randomly selected from the range 0–14.

If the child chooses to attempt random multiplication problems, he or she must define the range of numbers (between 0 and 14) from which the problems can be created.

## The Smiling Turtle

Once a response is entered, Snertle checks it against the correct answer. If the response is correct, then the turtle will smile, *GOOD!* will appear on its shell, and a high beep will sound. If the response is incorrect, Snertle's head will disappear into its shell and the message TRY AGAIN will appear on its side.

The user always gets a second chance. If the new response is correct, Snertle will poke its head out from the shell. If the answer is again incorrect, the correct answer will be displayed on the screen.

The program will keep producing problems until the X key is pressed in response to a problem. The percentage of correctly answered questions is then calculated and displayed; it gives credit only for those problems answered correctly on the first attempt. Snertle then returns to the menu, where the child may END the program or select more problems.

## Snertle

```
110  TEXT : HOME : VTAB 2: HTAB 15: PRINT "**SNERT
     LE**": VTAB 5
120  PRINT : VTAB 5: HTAB 10: PRINT "SELECT ONE:"
130  PRINT : PRINT : HTAB 10: PRINT "1) ADDITION"
140  PRINT : HTAB 10: PRINT "2) SUBTRACTION"
150  PRINT : HTAB 10: PRINT "3) MULTIPLICATION"
155  PRINT : HTAB 10: PRINT "4) END PROGRAM"
160  PRINT : PRINT : HTAB 10: PRINT "(ENTER 1,2,3
     OR 4) ";: INPUT Q: IF Q < 1 OR Q > 4 THEN 160

185 C = 14: IF Q = 1 OR Q = 2 THEN C = 99
187  IF Q = 3 THEN 1000
188  IF Q = 4 THEN  END
190  HOME : VTAB 3: HTAB 10: PRINT "ENTER LARGEST
     VALUE"
200  HTAB 10: PRINT "(MIN.:1  MAX.:";C;")";: INPUT
     R: IF R < 1 OR R > C THEN 200
230  HTAB 10: VTAB 10: PRINT "ENTER SMALLEST VALUE
     "
240  HTAB 10: PRINT "(MIN.:0  MAX.:";R;")";: INPUT
     S: IF S < 0 OR S > R THEN 240
263  HOME : VTAB 10: HTAB 7: PRINT "TYPE ";: INVERSE
     : PRINT "X";: NORMAL : PRINT " TO RETURN TO T
     HE MENU"
265  FOR I = 1 TO 2000: NEXT I: HOME
270 Z = 0:ZZ = 0: GR
275  GOSUB 1100: COLOR= 12: GOSUB 1170: GOSUB 1230

301 TR = 0:ZZ = ZZ + 1
305 L =  INT ( RND (1) * (R - S + 1)) + S
310  IF Q = 3 AND T = 1 THEN 320
315 K =  INT ( RND (1) * (R - S + 1)) + S
320 F$ =  STR$ (K):W = 0
325  IF K < L AND Q = 2 THEN 305
330 W = 0: GOSUB 3000
340 F$ =  STR$ (L)
345 W = 6: GOSUB 3000
346  ON Q GOSUB 6000,6000,6004
350  IF Q = 1 THEN M = K + L
355  IF Q = 2 THEN M = K - L
```

```
365  IF Q = 3 THEN M = K * L
380  GOSUB 740:MM = 1: IF M > 9 THEN MM = 2
385  IF M > 99 THEN MM = 3
393 V = Ø: COLOR= 12: GOSUB 1170
395  FOR J = Ø TO MM - 1
397  COLOR= 1: PLOT 21 - (5 * J),34
399  POKE  - 16368,Ø
400 H$ = "":H =  PEEK ( - 16384) - 128: IF H > Ø THEN
     H$ =  CHR$ (H)
407  IF H$ = "X" AND ZZ = 1 THEN  POKE  - 16368,Ø:
      GOTO 110
410  IF H$ = "X" THEN  TEXT : HOME : HTAB 15: PRINT
     "PERCENTAGE="; INT (Z / (ZZ - 1) * 100): POKE
      - 16368,Ø: GOTO 120
412  IF H < 48 OR H > 57 THEN 400
415 P =  VAL (H$)
420 V = V + (P * 10 ^ J):W = 14:X = 21 - (5 * J): GOSUB
     480: NEXT J
450  IF M = V THEN 470
451  FOR I = 1 TO 40: FOR J = 1 TO 2: NEXT J:L =  PEEK
     ( - 16336): NEXT I
452  COLOR= Ø: FOR I = 33 TO 38: HLIN 7,34 AT I: NEXT
     I: COLOR= 1
456  IF TR = 1 THEN 460
458 TR = 1: COLOR= Ø: GOSUB 1170: GOSUB 770:V = Ø:
      GOTO 395
460 M$ =  STR$ (M)
461  IF MM < 3 THEN  FOR I = 1 TO 3 - MM: READ X: NEXT
     I
462  FOR OO = 1 TO MM
464 P =  VAL ( MID$ (M$,OO,1))
465  READ X: GOSUB 480: NEXT OO: RESTORE
467  FOR I = 1 TO 900: NEXT
470  COLOR= 12: GOSUB 1170: IF TR = Ø THEN  GOSUB
     2500: GOSUB 755:Z = Z + 1: GOSUB 6500: HOME
471  GOSUB 2225: GOTO 301
480  COLOR= 1: IF P = Ø THEN  GOSUB 720
485  ON P GOSUB 500,525,555,585,610,633,660,680,70
     Ø: RETURN
500  VLIN 20 + W,24 + W AT X: VLIN 20 + W,24 + W AT
     X + 1: RETURN
525  HLIN X,X + 3 AT 20 + W: PLOT X + 2,21 + W: PLOT
     X + 3,21 + W: HLIN X,X + 3 AT 22 + W
530  VLIN 23 + W,24 + W AT X: VLIN 23 + W,24 + W AT
     X + 1: PLOT X + 2,24 + W: PLOT X + 3,24 + W: RETURN
555  VLIN 20 + W,24 + W AT X + 2: PLOT X,20 + W: PLOT
     X,22 + W: PLOT X,24 + W
560  PLOT X + 1,20 + W: PLOT X + 1,22 + W: PLOT X +
     1,24 + W: RETURN
```

```
585  VLIN 20 + W,22 + W AT X: PLOT X + 1,22 + W: VLIN
     20 + W,24 + W AT X + 2: PLOT X + 3,22 + W: RETURN

610  HLIN X,X + 3 AT 20 + W: HLIN X,X + 3 AT 22 +
     W: HLIN X,X + 3 AT 24 + W: PLOT X + 2,23 + W:
      PLOT X + 3,23 + W
615  PLOT X,21 + W: PLOT X + 1,21 + W: RETURN
633  VLIN 20 + W,24 + W AT X: VLIN 20 + W,24 + W AT
     X + 1: VLIN 22 + W,24 + W AT X + 3: HLIN X +
     2,X + 3 AT 20 + W
635  PLOT X + 2,22 + W: PLOT X + 2,24 + W: RETURN

660  HLIN X + 1,X + 3 AT 20 + W: PLOT X + 3,21 + W
     : PLOT X + 2,22 + W
665  VLIN 23 + W,24 + W AT X + 1: RETURN
680  GOSUB 720: HLIN X + 1,X + 2 AT 22 + W: RETURN

700  HLIN X,X + 3 AT 20 + W: HLIN X,X + 3 AT 22 +
     W: HLIN X,X + 3 AT 24 + W: VLIN 20 + W,24 + W
      AT X + 3
705  VLIN 21 + W,22 + W AT X: RETURN
720  VLIN 20 + W,24 + W AT X: VLIN 20 + W,24 + W AT
     X + 3: HLIN X + 1,X + 2 AT 20 + W: HLIN X + 1
     ,X + 2 AT 24 + W: RETURN
740  HLIN 10,27 AT 32: RETURN
755  VTAB 21: HTAB 19: PRINT "GOOD!": FOR I = 1 TO
     300: NEXT I: RETURN
770  VTAB 21: HTAB 16: PRINT "TRY AGAIN": FOR I =
     1 TO 1000: NEXT I: HOME : RETURN
1000  HOME : VTAB 4: HTAB 13: PRINT "DO YOU WISH T
     O:"
1010  PRINT : HTAB 9: PRINT "1) PRACTICE TIMES TAB
     LES"
1020  PRINT : HTAB 9: PRINT "2) PRACTICE RANDOM NU
     MBERS"
1030  PRINT : HTAB 9: PRINT "(ENTER 1 OR 2) ";: INPUT
     T: IF T < 0 OR T > 2 THEN 1030
1050  IF T = 2 THEN 190
1060  HOME : VTAB 5: HTAB 11: PRINT "ENTER TIMES T
     ABLE (1-14)"
1070  INPUT K: IF K < 1 OR K > 14 THEN 1070
1090 S = 0:R = 14: GOTO 263
1100 J = 12:JJ = 20: COLOR= 4: FOR I = 0 TO 8: HLIN
     J,JJ AT I:J = J - 1:JJ = JJ + 1
1110  NEXT I: FOR I = 8 TO 11: HLIN J + 1,JJ - 1 AT
     I: NEXT I: RETURN
1170  HLIN 30,32 AT 5: FOR I = 6 TO 10: HLIN 29,33
      AT I: NEXT I: COLOR= 0: PLOT 32,7: RETURN
```

```
1230  COLOR= 12: FOR I = 12 TO 15: HLIN 10,12 AT I
     : HLIN 21,23 AT I: NEXT I
1240  FOR I = 16 TO 17: HLIN 10,14 AT I: HLIN 21,2
     5 AT I: NEXT I: RETURN
2225  COLOR= 0: FOR I = 20 TO 38: HLIN 10,39 AT I:
      NEXT I: COLOR= 1: RETURN
2500  COLOR= 0: PLOT 32,10: PLOT 31,9: COLOR= 1: RETURN

3000  IF  LEN (F$) > 1 THEN 3030
3015 P =  VAL ( MID$ (F$,1,1))
3020 X = 21: GOSUB 480
3025  RETURN
3030 P =  VAL ( MID$ (F$,1,1))
3035 X = 16: GOSUB 480
3040 P =  VAL ( MID$ (F$,2,1))
3045 X = 21: GOSUB 480
3050  RETURN
5000  DATA   12,16,22
6000  HLIN 11,14 AT 29: HLIN 11,14 AT 28: IF Q = 1
      THEN  VLIN 27,30 AT 12: VLIN 27,30 AT 13
6001  RETURN
6004  PLOT 12,27: PLOT 14,27: PLOT 13,28: PLOT 12,
     29: PLOT 14,29: RETURN
6500  FOR I = 1 TO 20:L =  PEEK ( - 16336): NEXT I
     : FOR I = 1 TO 10: NEXT I: FOR I = 1 TO 40:L =
      PEEK ( - 16336): NEXT I: RETURN
```

# First Math

Steve Hamilton
Apple Translation by Patrick Parrish

*This math game for children features graphics, color, and sound. It displays the correct answer after the child has entered an incorrect one. In addition, there's an exciting graphics demo and a musical fanfare after ten consecutive correct answers.*

I was introduced to home computing last May. I purchased a computer partly for my two young boys, so they would grow up with some knowledge about a computer. Since the older boy was just approaching kindergarten, I thought it would be at least a year or so before he would be ready to operate it. But he was ready long before I had anticipated.

The following is a simple math tutorial that I developed for him. In this program, the user is given a choice of exercises: addition, subtraction, multiplication or division. Upper and lower limits can be specified for each of the two numbers in each problem, and the computer will generate random problems using numbers within the range that you specified.

## First Math

```
100  GOSUB 670
110  GOTO 260
120  DIM X(100),Y(100)
130 P = 2 * (355 / 113): FOR I = 1 TO 100:ANGLE =
     P * (I / 100):X(I) = 15 *  SIN (ANGLE):Y(I) =
     15 *  COS (ANGLE): NEXT I
140  RETURN
150  POKE 230,32: CALL 62450: HGR : CALL  - 1994: GR
     : COLOR= 7: PLOT 16,15: PLOT 24,15: COLOR= 4:
      PLOT 20,19
160  COLOR= 11: IF C1 = 0 THEN 190
170  PLOT 15,23: PLOT 25,23: PLOT 16,24: PLOT 24,2
     4: PLOT 17,25: PLOT 23,25: HLIN 18,22 AT 26
180  GOTO 200
190  HLIN 18,22 AT 23: PLOT 17,24: PLOT 23,24: PLOT
     16,25: PLOT 24,25: PLOT 15,26: PLOT 25,26
200  COLOR= 1
210  FOR I = 1 TO 100: PLOT X(I) + 20,Y(I) + 20: NEXT
     I
220  VTAB 22: HTAB 10: FLASH : IF C1 = 1 THEN  PRINT
     "  G O O D   J O B !! ": NORMAL : GOTO 250
```

```
230  NORMAL : VTAB 22: HTAB 6: PRINT "S O R R Y ,
     B U T  ";B;" ";A$;" ";C;"=";" ";: INVERSE : PRINT
     E: NORMAL
240  FOR I = 1 TO 2000: NEXT I
250  FOR I = 1 TO 1500: NEXT I: HOME : HGR : POKE
     34,0: HOME : TEXT : RETURN
260  HOME : INVERSE : VTAB 10: HTAB 12: PRINT "F I
      R S T   M A T H": NORMAL : VTAB 18: HTAB 4: PRINT
     ".........P L E A S E   W A I T"
270  GOSUB 120
280  HOME : VTAB 4: HTAB 7: PRINT "TO ";: INVERSE
     : PRINT "ADD";: NORMAL : PRINT "       : TYPE
     +"
290  VTAB 6: HTAB 7: PRINT "TO ";: INVERSE : PRINT
     "SUBTRACT";: NORMAL : PRINT " : TYPE -"
300  VTAB 8: HTAB 7: PRINT "TO ";: INVERSE : PRINT
     "MULTIPLY";: NORMAL : PRINT " : TYPE *"
310  VTAB 10: HTAB 7: PRINT "TO ";: INVERSE : PRINT
     "DIVIDE";: NORMAL : PRINT " : TYPE /"
320  VTAB 13: HTAB 7: PRINT "YOUR CHOICE= ";
330  INPUT A$: IF A$ <  > ("*") AND A$ <  > ("+") AND
     A$ <  > ("-") AND A$ <  > ("/") THEN 330
340  VTAB 17: HTAB 7: INPUT "HIGHEST NUMBER= ?";UL
     : VTAB 19: HTAB 7: INPUT "LOWEST NUMBER=  ?";
     LL
350 R = UL + 1 - LL
360 C =  INT ( RND (1) * R) + LL:B =  INT ( RND (1
     ) * R) + LL
370  IF A$ = ("+") THEN  DEF  FN A(X) = B + C
380  IF A$ = ("-") THEN  DEF  FN A(X) = B - C
390  IF A$ = ("*") THEN  DEF  FN A(X) = B * C
400  IF A$ = ("/") AND C = 0 THEN 360
405 IF A$ <> ("/") THEN 430
410  IF A$ = ("/") AND  INT (B / C) <  > B / C THEN
     360
420  IF A$ = ("/") THEN  DEF  FN A(X) = B / C
430  HOME : VTAB 7: HTAB 8: PRINT "CORRECT ANSWERS
      IN A ROW= ";: INVERSE : PRINT D: NORMAL
440 E =  FN A(X): VTAB 15: HTAB 15: PRINT B;" ";A$
     ;" ";C;"= ";: INPUT F: IF F <  > E THEN 480
450  HOME :C1 = 1: GOSUB 150
460 D = D + 1: IF D = 10 THEN 500
470  GOTO 360
480  HOME :C1 = 0: GOSUB 150
490 D = 0: GOTO 430
500  REM  YOU WIN!!
510 D = 0: GOSUB 560
520  VTAB 22: HTAB 8: FLASH : PRINT "  Y O U   D I
      D   I T !!"
```

```
530  FOR I = 1 TO 5: POKE 768,1: POKE 769,200 - I *
     30: CALL 770: NEXT I: FOR I = 1 TO 10: POKE 7
     68,1: POKE 769,40 + I * 20: CALL 770: NEXT I
540  NORMAL : VTAB 24: HTAB 10: PRINT "TRY AGAIN (
     Y/N) ?";: GET A$: IF A$ = ("Y") THEN  TEXT : GOTO
     280
550  TEXT : HOME : HTAB 5: VTAB 8: PRINT "...SEE Y
     A LATER...": END
560  POKE 230,32: CALL 62450: HGR : CALL  - 1994: GR

570  FOR J = 1 TO 3
580 CL = 0:L0 = 0:H1 = 19:S1 = 1: GOSUB 620
590 CL = 17:L0 = 19:H1 = 0:S1 =  - 1: GOSUB 620
600  NEXT J
610  RETURN
620  FOR I = L0 TO H1 STEP S1: COLOR=  INT ( RND (
     1) * CL):X1 = 19 - I:X2 = 20 + I:Y1 = 19 - I:
     Y2 = 20 + I
630  HLIN X1,X2 AT Y1: VLIN Y1 + 1,Y2 AT X2
640  HLIN X2 - 1,X1 AT Y2: VLIN Y2 - 1,Y1 AT X1
650  NEXT I
660  RETURN
670  REM  LOAD MUSIC ROUTINE
680  FOR I = 770 TO 795: READ M: POKE I,M: NEXT I
690  DATA  172,01,03,174,01,03,169,04,32,168,252,1
     73,48,192,232,208,253,136,208,239,206,0,03,20
     8,231,96
700  RETURN
```

# Crosswords

William Loercher
Apple Translation by Patrick Parrish

*With this program, your Apple will be able to construct simple crossword puzzles. The finished puzzle can be displayed on the screen or printed on your printer.*

If you've ever tried to make your own crossword puzzles, you know the procedure can be very time-consuming. But can your Apple create the puzzles? With this program, it surely can.

The program can be embellished in several ways. For instance, after all 23 rows are tested (Z=23), you could add another section that tests the columns for word fits. This should result in a better puzzle. You could also keep track of the words that fit a given location in another array and then choose the longest word from that list.

Lines 2010–2110 are the DATA statements containing the words used in the puzzle. Feel free to substitute your own words for the ones given here.

## Crossword Puzzles

```
100  TEXT : HOME
110  HTAB 2: FOR X = 1 TO 38: PRINT "*";: NEXT X
120  VTAB 1: FOR Y = 2 TO 23: FOR X = 2 TO 39 STEP
     37: VTAB Y: HTAB X: PRINT "*";: NEXT X,Y
130  HTAB 2: VTAB 24: FOR I = 1 TO 38: PRINT "*";:
      NEXT I
140 A = 11: FOR F = 1 TO 16:A = A + 1:E = 18: READ
     A$
150  FOR B = 3 TO A: VTAB 19: HTAB B: PRINT " "A$:
      NEXT B
160  FOR C = 1 TO 10: HTAB 1
170  FOR D = 1 TO E: VTAB D + 1: NEXT D
180  HTAB A + 1: PRINT A$: HTAB A + 1: PRINT " ":E
      = E - 1: NEXT C: NEXT F
190  FOR X = 1 TO 2000: NEXT
200 A = 19: FOR F = 1 TO 7:E = 18: READ A$: FOR B =
     3 TO A - 1: VTAB 19: HTAB B: PRINT " "A$: NEXT
     B
210  VTAB 19: PRINT " *"
220  FOR C = 1 TO 13 - F: HTAB 1: FOR D = 1 TO E: VTAB
     D + 1: NEXT D
230  HTAB A: PRINT A$: HTAB A: PRINT " ":E = E - 1
     : NEXT C: POKE 1210,143: NEXT F
```

```
240  FOR X = 1 TO 19: READ A$: IF A$ = "Ø" THEN 27
     Ø
250  VTAB 14: HTAB X + 9: PRINT A$
260  GOTO 280
270  VTAB 14: HTAB X + 9: PRINT " "
280  FOR Y = 1 TO 200: NEXT Y: NEXT X
290  FOR I = 1 TO 2000: NEXT : HOME
300  VTAB 4: INPUT "HOW MANY WORDS (MAX:110)?";N
310  VTAB 7: INPUT "HOW MANY VERTICAL WORDS (15-25
      WORKS    WELL)?";K
320  VTAB 10: INPUT "RESULTS ON SCREEN OR PRINTER
     (S OR P)?";S$
330  DIM N$(N),L(N)
340  FOR X = 1 TO N: READ N$(X):L(X) =  LEN (N$(X)
     ): NEXT X: HOME
350  INVERSE : FOR I = 1 TO 23: FOR J = 1 TO 39: HTAB
     J: VTAB I: PRINT " ";: NEXT J: NEXT I: NORMAL

360  DIM XL%(23): FOR I = Ø TO 7
370 XL%(I) = 1024 + 128 * I
380 XL%(I + 8) = 1064 + 128 * I
390 XL%(I + 16) = 1104 + 128 * I: NEXT I
400  FOR Z = 1 TO K:E = Ø
410 R =  INT ( RND (1) * N) + 1: IF N$(R) = "Ø" THEN
     410
420 ROW =  INT ( RND (1) * 23):COL =  INT ( RND (1
     ) * 40)
430 P = XL%(ROW) + COL
440  FOR X = Ø TO L(R) + 1:B =  PEEK (XL%(ROW + X)
      + COL):C =  PEEK (XL%(ROW + X) + COL - 1):D =
      PEEK (XL%(ROW + X) + COL + 1)
450  IF B <  > 32 OR C <  > 32 OR D <  > 32 THEN X
      = L(R) + 1: NEXT X: GOTO 420
460 E = E + 1
470  NEXT X: IF E = L(R) + 1 THEN E = Ø
480  POKE P,170: REM  PLACE * ON EITHER SIDE OF WO
     RD
490  FOR X = 1 TO L(R): POKE (XL%(ROW + X) + COL),
      ASC ( MID$ (N$(R),X,1)) + 64
500  NEXT : POKE (XL%(ROW + X) + COL),170:N$(R) =
     "Ø": NEXT Z: REM  GET ANOTHER WORD
510 Z = Ø
520 Z = Z + 2:L = Ø
530  IF Z > 23 THEN 770
540  FOR X = 1 TO N:E = Ø:G = Ø
550  IF N$(X) = "Ø" OR L + L(X) + 2 > 39 THEN  NEXT
     X
560  IF X > N THEN 520
570  FOR Y = 1 TO L(X)
580 B =  PEEK (XL%(Z) + L + Y)
```

```
590 C =  ASC ( MID$ (N$(X),Y,1)) + 64
600  IF B = 32 OR B = C THEN E = E + 1
610  IF B = 32 THEN G = G + 1
620  IF E = 0 THEN 660
630  IF B = 160 OR B = 170 OR G = L(X) THEN L = L +
     1: GOTO 540
640  IF E = L(X) THEN 680
650  NEXT Y
660  NEXT X
670 L = L + 1: GOTO 540
680 B =  PEEK (XL%(Z) + L + L(X) + 1)
690  IF B = 170 OR B = 32 THEN 710
700 L = L + 1: NEXT X: GOTO 520
710 B =  PEEK (XL%(Z) + L)
720  IF B = 32 OR B = 170 THEN 740
730 L = L + 1: NEXT X: GOTO 520
740  POKE (XL%(Z) + L),170
750  FOR L1 = 1 TO L(X): POKE (XL%(Z) + L + L1), ASC
     ( MID$ (N$(X),L1,1)) + 64
760  NEXT L1: POKE (XL%(Z) + L + L1),170:N$(X) = "
     0":L = L + L1: GOTO 540
770  IF S$ = "P" THEN 790
780  GOTO 1030
790  PR# 1: PRINT  CHR$ (9)"255N"
800  FOR X = 0 TO 23:B = 20: FOR Y = 0 TO 39: IF Y
      > 0 THEN B = 0
810 A =  PEEK (XL%(X) + Y): IF A = 160 OR A = 170 OR
     A = 32 THEN A = 237
820 B$ =  CHR$ (A - 64)
830  PRINT  SPC( B)B$;: IF Y = 39 THEN  PRINT
840  NEXT Y: NEXT X: PR# 0: PRINT : GOTO 1030
850  DATA  C,R,O,S,S,W,O,R,D, ,P,U,Z,Z,L,E
860  DATA  P,R,O,G,R,A,M
870  DATA  B,Y,0,W,I,L,L,I,A,M,0,L,O,E,R,C,H,E,R
1030  PRINT "    DONE! ";: INVERSE : PRINT "E";: NORMAL
     : PRINT "ND OR ";: INVERSE : PRINT "C";: NORMAL
     : PRINT "ONTINUE?";
1033  GET R$: IF R$ = "" THEN 1033
1035  IF R$ = "C" THEN  RUN
1040  HOME : PRINT "BYE!": END
2000  REM  NUMBER OF WORDS=110
2010  DATA  ASSENT,ASTERISK,BAG,BITE,BOOT,BUFFER,B
     ULK,CELL,CEMENT,CLAIM
2020  DATA  CAT,PERSON,CHAIR,CAN,PAPER,NUMBER,OWL,
     PLATE,CIRCLE,PENCIL
2030  DATA  VICTORY,LETTER,DOORWAY,SAIL,LOVE,MOTHE
     R,SON,DAUGHTER,CAR,HAPPY
2040  DATA  TOMORROW,TRUCK,BUSINESS,FEELINGS,SUNSE
     T,BRIGHT,SUMMER,MOVIE,CHESS,PAINT
```

```
2050  DATA  TENNIS,NET,BALL,RACKET,COURT,PLAYER,OF
      FICIAL,BOOTH,SCORE,POINT
2060  DATA  PINS,RACK,NEEDLES,CHAIR,STOOL,CEILING,
      SOUND,PROFESSOR,TEACHER,SCHOOL
2070  DATA  COMPUTE,KEYBOARD,BYTE,BIT,STOP,GO,END,
      MICROCOMPUTER,SOLUTION,FINE
2080  DATA  ROOM,SAD,JOY,PEACE,BOATING,RIVER,LAKE,
      SWIMMING,BOARD,GRASS
2090  DATA  EGG,EXHALE,GLORY,ILLUSIVE,IMMORAL,DESK
      ,LET,LEVEL,MYSTERY,MYSELF
2100  DATA  NAIL,TWO,MUTE,OFF,OFFER,PALM,PANEL,PEN
      NY,CENT,DOLLAR
2110  DATA  RENDER,THE,WING,POLICE,HELP,TOIL,TREE,
      LIGHT,RUN,POLL
```

# Chemistry Lab

Joanne Davis

*This program, which will be of special interest to teachers, brings chemistry to life for elementary students. It features an easy-to-use menu and animated graphics. A color monitor is recommended.*

"Chemistry Lab" encourages elementary school students to hypothesize and review concepts by allowing them to duplicate laboratory experiences in chemistry. It uses standard chemical indicators to identify a variety of substance types, including acids, bases, sugars, or starches.

The program is menu-driven. After choosing a topic, the student is given instructions. Those are followed by a picture of an eyedropper containing the indicator (in the appropriate color), a beaker (containing the material to be tested), and the material and indicator names. The student predicts the result of the test, as he or she would before conducting a laboratory experiment, and INPUTs the prediction.

When the test is carried out, the eyedropper releases its contents drop-by-drop and the beaker fills with liquid. The liquid's color indicates the presence of acid, sugar, etc. Comments then reinforce the material's classification.

This procedure is repeated to test four more substances. More items can easily be added by DIMensioning the arrays and adding more DATA.

## Two Special Techniques

Two of the techniques used in this program should be of special interest. The animation is created by alternating between a color and black, and by time delays caused by empty FOR-NEXT loops. The inside of the dropper is blacked out a line at a time, with the delay making the action visible. The previous position of the drop is blacked out, and the drop is redrawn at a new location. Then the beaker is filled up (a line at a time).

Since the sugar and starch tests require virtually the same instructions, an easy way was found to make the needed alterations. The changes are READ in from DATA statements and inserted into the message.

A science curriculum can come alive with animated laboratory experiments. Try it and see.

## Chemistry Lab

```
20  TEXT : HOME
30  VTAB 8: HTAB 15: PRINT "***************"
40  HTAB 15: PRINT "WELCOME TO THE": HTAB 15: PRINT
     "CHEMISTRY LAB": HTAB 15: PRINT "************
     **"
50  FOR TT = 1 TO 4500: NEXT
60  HOME : VTAB 5: PRINT "CHOOSE TEST 1, 2, OR 3:"
     : PRINT : HTAB 5: PRINT "1. ACID": HTAB 5: PRINT
     "2. STARCH": HTAB 5: PRINT "3. SUGAR"
65  HTAB 5: PRINT "4. QUIT"
70  GET CH$:CH =  VAL (CH$): ON CH GOSUB 1000,2000
    ,2500,100
80  TEXT : GOTO 60
100  END
1000  REM  ACID/BASE***PHENOL
1020  REM  INSTRUC
1025  GOSUB 4000
1030  TEXT : HOME
1040  PRINT : PRINT : PRINT "YOU ARE GOING TO TEST
     SOME MATERIALS TO": PRINT "SEE IF THEY ARE A
    CIDS OR BASES. THE": PRINT "INDICATOR WILL TU
    RN ";: INVERSE : PRINT "PINK";: NORMAL : PRINT
    " IN AN ";: INVERSE : PRINT "ACID";: NORMAL :
     PRINT "."
1045  PRINT : PRINT "TYPE ";: INVERSE : PRINT "A";
    : NORMAL : PRINT " IF YOU THINK THAT THE MATE
    RIAL": PRINT "IS AN ACID."
1047  PRINT "TYPE ";: INVERSE : PRINT "B";: NORMAL
    : PRINT " IF YOU THINK THAT THE MATERIAL": PRINT
    "IS A BASE."
1050  PRINT : PRINT "HIT ANY KEY TO BEGIN.": GET A
    $
1070  HOME
1150  FOR X = 1 TO 5
1152 DROP = 15: GOSUB 5000
1155  VTAB 21
1160  PRINT "INDICATOR: ";: INVERSE : PRINT "PHENO
    LPHTHALEIN": NORMAL
1170  PRINT "NOW TESTING: ";: INVERSE : PRINT N$(X
    ): NORMAL
1180  PRINT : INVERSE : PRINT "A";: NORMAL : PRINT
    "CID OR ";: INVERSE : PRINT "B";: NORMAL : PRINT
    "ASE ?": GET AN$: IF AN$ <  > "A" AND AN$ <  >
    "B" THEN  HOME : GOTO 1155
1190  IF ID$(X) = "A" THEN BEAK = 11: GOTO 1210
1200 BEAK = DROP
1210  GOSUB 6000
```

```
1220  HOME : IF ID$(X) = "A" THEN  PRINT N$(X);" I
     S AN ACID.": GOTO 1240
1230  PRINT N$(X);" IS A BASE."
1240  FOR TT = 1 TO 4000: NEXT TT
1250  NEXT X
1400  FOR TT = 1 TO 1000: NEXT TT
1500  RETURN
2000 Y = 1: GOSUB 4000: GOSUB 3000: RETURN
2500 Y = 2: GOSUB 4000: GOSUB 3000: RETURN
3000  REM  STARCH/SUGAR INSTRUCTIONS
3010  TEXT : HOME
3020  PRINT : PRINT : PRINT "YOU ARE GOING TO TEST
      SOME MATERIALS TO": PRINT "SEE IF THEY CONTA
     IN ";B$(Y);".": PRINT : PRINT "THE INDICATOR
     WILL TURN ";: INVERSE : PRINT C$(Y);: NORMAL
     : PRINT " IN A ": INVERSE : PRINT B$(Y);: NORMAL
     : PRINT "."
3030  PRINT : PRINT "TYPE ";: INVERSE : PRINT "Y";
     : NORMAL : PRINT " IF YOU THINK THAT THE MATE
     RIAL": PRINT "CONTAINS ";B$(Y);" ."
3040  PRINT : PRINT "HIT ANY KEY TO BEGIN.": GET A
     $
3050  HOME
3060  FOR X = 1 TO 5
3070 DROP = P(Y): GOSUB 5000
3080  VTAB 21
3090  PRINT "INDICATOR: ";: INVERSE : PRINT IN$(Y)
     : NORMAL
3092 :
3094 :
3096 :
3100  PRINT "NOW TESTING: ";: INVERSE : PRINT N1$(
     X): NORMAL
3110  PRINT : INVERSE : PRINT B$(Y);: NORMAL : PRINT
     " (Y/N) ?": GET AN$
3120  IF AN$ <  > "Y" AND AN$ <  > "N" THEN  HOME
     : GOTO 3080
3130  ON Y GOSUB 3500,3600
3140  FOR TT = 1 TO 4000: NEXT TT
3150  NEXT X
3200  RETURN
3500  REM  STARCH MESSAGE
3510  IF IH$(X) = "S" THEN BEAK = 3: GOTO 3530
3520 BEAK = DROP
3530  GOSUB 6000
3535  HOME
3540  IF IH$(X) = "S" THEN  PRINT N1$(X);" CONTAIN
     S STARCH.": GOTO 3560
3550  PRINT N1$(X);" DOES NOT CONTAIN STARCH."
3560  RETURN
```

```
3600  REM  SUGAR MESSAGE
3610  IF IR$(X) = "S" THEN BEAK = 9: GOTO 3625
3620 BEAK = DROP
3625  GOSUB 6000
3627  HOME
3630  IF IR$(X) = "S" THEN  PRINT N1$(X);" CONTAIN
     S SUGAR.": GOTO 3660
3650  PRINT N1$(X);" DOES NOT CONTAIN SUGAR."
3660  RETURN
4000  RESTORE
4005  FOR X = 1 TO 5
4010  READ N$(X),ID$(X),N1$(X),IH$(X),IR$(X)
4020  NEXT X
4030  FOR X = 1 TO 2
4040  READ B$(X),C$(X),IN$(X),P(X)
4050  NEXT X
4060  RETURN
5000  REM  SCREEN**OUTLINE BEAK AND DROP
5010  GR : COLOR= 10
5020  VLIN 0,20 AT 14: VLIN 0,20 AT 18
5030  HLIN 15,17 AT 0: HLIN 13,19 AT 6
5040  HLIN 15,17 AT 21: VLIN 21,24 AT 16
5050  PLOT 9,28: VLIN 28,38 AT 10
5060  VLIN 28,38 AT 21: HLIN 11,20 AT 38
5065  REM  INSIDE DROPPER
5070  COLOR= DROP
5080  VLIN 15,20 AT 15: VLIN 15,20 AT 16: VLIN 15,
     20 AT 17
5500  RETURN
6000  REM  ANIMATION
6010  COLOR= 0
6015 P = 31:S =  - 16336
6020  FOR G = 15 TO 20
6025  PLOT 16,P
6030  HLIN 15,17 AT G:SO =  PEEK (S) -  PEEK (S) -
     PEEK (S)
6033  GOSUB 6500
6035  FOR TT = 1 TO 400: NEXT TT
6037  COLOR= 0
6040  NEXT G
6050  FOR TT = 1 TO 400: NEXT TT
6100  COLOR= BEAK
6110  FOR G = 37 TO 32 STEP  - 1
6120  HLIN 11,20 AT G: FOR TT = 1 TO 250: NEXT
6130  NEXT G
```

```
6140  RETURN
6500  COLOR= DROP:P = P + 1: PLOT 16,P
6510  RETURN
7000  DATA SOAP,B,BREAD,S,S,LEMON JUICE,A,CRACKER,
     S,0,COLA,A,CHOCOLATE,0,S,BAKING SODA,B,COLA,0
     ,S,VINEGAR,A,FLOUR,S,0
7010  DATA  STARCH,PURPLE,IODINE,13,SUGAR,ORANGE,B
     ENEDICTS SOLUTION,7
```

# Typing Teacher

Alan McCright
Apple Translation by Patrick Parrish

*Typing program listings is much easier if you know the keyboard and don't need to watch your fingers. The program given here helps you learn touch typing; it will show your progress, too, by giving you a score in characters per minute or in words per minute.*

Those who must rely on hunt-and-peck typing have probably discovered just how tedious it can be, especially when typing in programs. This program will familiarize you with keyboard layout and help you learn to touch type.

The idea is to let your fingers find the correct key without looking at the keyboard. When the program is run, a representation of the keyboard layout appears on the screen. The characters are printed in an approximation of their keyboard positions. Check the key's location on the display, and try to get your finger to move there without looking down at the keyboard.

When you start the program, a "clock" begins to run. The program puts a character on the screen, waits for your response, and checks to see if it matches the test character. If so, your score will be incremented by one. After approximately one minute, the test will end and your score will be printed.

Because the Apple lacks a realtime clock, a special counter routine is used. Incrementing occurs in line 320 (while waiting for a keyboard response) and again in line 350 (to account for the time required to process each response). After approximately a minute, a certain counter value will be reached (in line 330). The testing routine will halt, and a score will be displayed. The score can be given in words per minute by changing line 440 as follows:

```
440 HTAB12:VTAB7:INVERSE:PRINT"WORDS/MINUTE=";
    "";CCTN/5:NORMAL
```

If you modify this program, check to be sure that the timing is still correct. If not, adjust line 350 as required.

Scoring assumes that the average English word is five letters long. However, since the characters are chosen at random (which I found ideal for learning to type in programs), each individual character has to be recognized rather than recalled

as part of a word. Thus, scoring in words per minute will lead to some low (though accurate) scores, even for good typists.

How fast can the program run? In the word-per-minute mode, by deleting line 360 and all of the REMs, and holding down any key after running, a score of 60–70 words per minute is typical. However, when you are actually running a test, your own reaction time will keep you from reaching that level. You might want to modify the routine using word lists instead of random characters to get an idea of your effective speed.

## Typing Teacher

```
100  FOR I = 770 TO 795: READ M: POKE I,M: NEXT
110  HOME : PRINT : HTAB 14: INVERSE : PRINT "TYPI
     NG TEACHER": NORMAL
120 CCNT = 0: REM  ZERO CHARACTER COUNTER
130  REM  ** ROUTINE TO ENTER CHARACTER POSITION D
     ATA **
140  FOR ROW = 11 TO 17 STEP 2: REM  ROW DATA TO P
     OKE
150  FOR COL = 9 TO 33 STEP 2: REM  COLUMN DATA T
     O POKE
160  READ CHAR
170  IF CHAR = 0 THEN  NEXT ROW: GOTO 150
180  IF CHAR =  - 1 THEN 250
190  POKE 796 + (CHAR * 2),COL: POKE 796 + (CHAR *
     2) + 1,ROW
200  IF CHAR = 32 THEN 220
210  HTAB COL: VTAB ROW: INVERSE : PRINT  CHR$ (CH
     AR): NORMAL
220  NEXT COL
230  GOTO 150
240  REM  ** TIMER AND SELECT RANDOM CHARACTER **
250  HTAB 10: VTAB 20: INVERSE : PRINT "HIT ANY KE
     Y TO START": NORMAL : GET A$
260  HTAB 10: VTAB 20: FOR I = 1 TO 20: PRINT " ";
     : NEXT I
270 N =  INT (( RND (1) * 47) + 44): REM  CHOOSE A
      RANDOM CHARACTER
280  IF N >  = 60 AND N <  = 64 OR N = OLDCHAR THEN
     270
290 OLDCHAR = N
300  HTAB 20: VTAB 7: PRINT  CHR$ (N): REM  PRINT
     RANDOM NUMBER CHARACTER

310  REM  **PROCESS YOUR RESPONSE**
```

```
320  IF  PEEK ( - 16384) < 128 AND TIME < 2710 THEN
     TIME = TIME + 1: GOTO 320
330  IF TIME >  = 2710 THEN 440
340  GET A$:CHAR =  ASC (A$): POKE 768,30: POKE 76
     9,1: CALL 770:CCNT = CCNT + 1: REM  *ADD ONE
     TO TOTAL*
350 TIME = TIME + 10
360  GOSUB 420
370  PRINT  CHR$ (CHAR)
380  FOR I = 1 TO 10: NEXT I
390  GOSUB 420: INVERSE : PRINT  CHR$ (CHAR): NORMAL

400  IF CHAR <  > N THEN CCNT = CCNT - 1: POKE 768
     ,1: POKE 769,175: CALL 770
410  GOTO 270
420  IF CHAR <  > N THEN  POP : GOTO 400
430  HTAB ( PEEK (796 + 2 * CHAR)): VTAB ( PEEK (7
     97 + 2 * CHAR)): RETURN
440  HTAB 9: VTAB 7: INVERSE : PRINT "CHARACTERS/M
     INUTE =";" ";CCNT: NORMAL
450  HTAB 10: VTAB 20: INVERSE : PRINT " HIT 'R' T
     O RESTART ": NORMAL
460  POKE 768,250: POKE 769,2: CALL 770
470  GET A$: IF A$ = "R" THEN  RUN
480  END
490  REM  **MUSIC ML DATA**
500  DATA  172,01,03,174,01,03,169,04,32,168,252,1
     73,48,192,232,208,253,136,208,239,206,0,03,20
     8,231,96
510  REM  **ASCII DATA FOR KEYBOARD**
520  DATA  49,50,51,52,53,54,55,56,57,48,58,45,0
530  DATA  81,87,69,82,84,89,85,73,79,80,0
540  DATA  65,83,68,70,71,72,74,75,76,59,0
550  DATA  32,90,88,67,86,66,78,77,44,46,47,-1
```

# Memory Trainer

Harvey B. Herman
Apple Translation by Patrick Parrish

*Can't remember your phone number? This program might help you improve your memory skills. Using similar training aids, some people have learned to memorize 80-digit numbers.*

An article entitled "Exceptional Memory" appeared recently in *American Scientist* (vol. 70, no. 6, 1982, p. 607). When most people read a random sequence, they can remember only five to nine digits, the apparent limit of short-term memory (STM). However, the authors described experiments in which a person with normal memory was trained to recall a sequence of more than 80 random digits.

Many people would call such a feat (recalling 80 digits) *exceptional.* But the authors said that this skill may not be uncommon. In fact, diligent practice frequently resulted in improvement in the ability to rapidly transfer information into long-term memory (LTM). A "normal" memory could thereby be transformed into an "exceptional" one.

It seemed like it would be easy to automate the task of memory training. Consequently, I wrote "Memory Trainer." Random digits are flashed on the screen at a specified rate; if the sequence is repeated correctly, the next sequence of digits is increased by one. When an error is made, the length of the sequence decreases by one. You can stop the experiment at any point, and the maximum sequence length achieved will be displayed.

## Memory Trainer

```
210  DIM N(76)
220 MA = Ø: REM  MA=MAX CORRECT SPAN
230  TEXT : HOME : INVERSE : PRINT "MEMORY TRAININ
     G PROGRAM": NORMAL
240  PRINT
250  INPUT "DIGIT RATE (1-1Ø) ? ";DR
260  IF DR < 1 OR DR > 1Ø THEN 23Ø
270  PRINT
280  INPUT "INITIAL SEQUENCE LENGTH ? ";SL
290  IF SL < 2 THEN SL = 2
300  IF SL > 76 THEN SL = 76
320  REM  SEQ LEN - MIN 2:MAX 76
```

```
330  PRINT : INVERSE : PRINT "CURRENT DIGIT SPAN";
     : NORMAL : PRINT " ";SL
340  REM  FLASH GET SET AND DIGITS
350  PRINT : FLASH : PRINT "GET SET";: FOR I = 1 TO
     300: NEXT I: NORMAL : HTAB 1: PRINT "GET SET"
     ;
360  PRINT  CHR$ (7): PRINT "*";: FOR I = 1 TO 125
     0: NEXT I
370  FOR I = 1 TO SL
380 N(I) =  INT ( RND (1) * 10)
400  HTAB 1: INVERSE : PRINT N(I);: FOR J = 1 TO 1
     00: NEXT J: NORMAL
410  HTAB 1: PRINT N(I);: IF I = SL THEN  HTAB 1: PRINT
     " ";
420  FOR K = 1 TO DR * 100: NEXT K
430  NEXT I
450  PRINT : PRINT : PRINT "INPUT DIGITS":FL = 0
460  PRINT "
       ";
470  PRINT "                                          ":
      VTAB 13: PRINT "*";: HTAB 1: INPUT "";A$
480  IF  LEN (A$) <  > SL THEN FL = 1: GOTO 540
490  FOR I = 1 TO SL
500  IF  VAL ( MID$ (A$,I,1)) <  > N(I) THEN FL =
     1:I = SL
510  NEXT I
520  REM  FL=0 - CORRECT - INCREASES SEQ LEN BY ON
     E
530  REM  FL=1 - INCORRECT - DECREASES SEQ LEN BY
     ONE
540  IF FL = 1 THEN  INVERSE : VTAB 15: PRINT "INC
     ORRECT";: NORMAL : PRINT " - TRY A SHORTER SP
     AN NEXT ":SL = SL - 1
550 IF FL = 1 THEN PRINT "
                        ";:REM 41 SPACES
560  IF FL = 1 THEN  PRINT "
                      ": VTAB 16
570  IF FL = 1 THEN  FOR J = 1 TO SL + 1: PRINT  RIGHT$
     ( STR$ (N(J)),1);: NEXT J: GOTO 620
580  VTAB 15: INVERSE : PRINT "CORRECT";: NORMAL :
      PRINT " - TRY A LONGER SPAN NEXT      ":SL =
     SL + 1
590  IF MA < SL - 1 THEN MA = SL - 1
600 PRINT "
         ";:REM 43 SPACES
610 PRINT "
         ": VTAB 18 :REM 43 SPACES
```

```
620  HTAB 1: VTAB 19: INPUT "AGAIN (Y OR N) ? ";N$
     : VTAB 19: HTAB 18: PRINT "    ": IF SL < 1 THEN
     SL = 1
630  IF SL > 76 THEN SL = 76
640  IF  LEFT$ (N$,1) = "Y" THEN  VTAB 6: GOTO 330

650  PRINT : HTAB 7: PRINT "HOPE YOU IMPROVED YOUR
      SPAN!": PRINT
660  HTAB 7: INVERSE : PRINT "HIGHEST CORRECT DIGI
     T SPAN";: NORMAL : PRINT " ";MA: VTAB 23
```

# Oscilloscope

Rob Smythe

*Here is a program, designed especially for physics teachers, that makes good use of the Apple's high-resolution graphics.*

Unless your school's equipment is better than mine, you probably find it tricky to demonstrate waveforms in class. It's hard to stabilize an oscilloscope pattern whenever the input frequency is changed; if you're mixing several frequencies it can be almost impossible.

With this program, however, you can demonstrate complex waveforms on your Apple. You can show effects of varying amplitude and frequency, add up to five overtones (each with its own amplitude), and show the resultant wave pattern for up to six different notes (particularly useful for demonstrating the cause of beat notes).

When you run this program you will be presented with a table (initially showing that there are no notes in memory) and a menu prompting you for single keystroke selection of commands. Use the 1, 2, 3, 4, 5 or 6 key to set the amplitude and frequency of a note. Enter as many notes as you wish, or change them one by one. Press P to plot the resultant waveform. After the oscilloscope pattern is drawn and you have finished studying it, return to the menu by pressing any key.

The S key will let you alter the plotting speed, which is initially set at 4. This determines the increment along the x-axis (time axis) between plotted points. When using frequencies over about 500 Hz, you might have to set speed at 1 or 2 (because at coarser settings significant changes in wave shape might occur between points and be missed). Try 800 Hz at speed 4 and at speed 1 to see how this affects the display.

To clear all notes from the table, press C and confirm with a Y.

Try notes of amplitude 10 to 20 in a frequency range of 100 to 500. Create a complicated note using all overtones, with amplitudes 10 or less (so that you don't go off the top of the screen). Beat patterns look nice when you play notes of frequency 1000 and 1050 together.

Note that you can change TIME in line 2120 to allow for a different range of suitable frequencies. You might add TIME

input to the menu, so that beats can be shown effectively with frequencies that are very close together.

## How It Works

| Line(s) | |
|---|---|
| **1000-1020** | Print table and menu routine |
| **1030** | Formats numbers in display |
| **1100** | Waits for single keystroke input |
| **1110** | Inputs data |
| **1120 on** | Process data and reject invalid input |
| **2000's** | Plot routine |
| **2000-2100** | Draw axes |
| **2150-2160** | Pick X value in radians |
| **2170** | Sums the waves |
| **2190** | Scales X and Y to fit screen |
| **2200** | Checks for off-scale values |
| **2210** | Plots |
| **3000-3040** | Subroutine to check that points are not off-scale |

## Oscilloscope

```
50 G$ =  CHR$ (7): REM ERROR BEEP
100 SP = 4: REM  PLOTTING SPEED  FROM 1 (SLOW=MOST
     ACCURATE) TO 5
997 :
998  REM  DATA INPUT
999 :
1000  TEXT : HOME
1010  PRINT "    NOTE    AMP    FREQ": PRINT
1020  FOR I = 1 TO 6: PRINT  TAB( 7);I;"       ";
1030 A$ =  RIGHT$ ("       " +  STR$ (FR(I)),6): IF
    AMP(I) < 10 THEN  PRINT " ";
1040  PRINT AMP(I);"  ";A$
1050  PRINT : NEXT I
1060  PRINT : PRINT : PRINT "SPEED - ";SP
1070  VTAB 21
1080  PRINT "CHANGE NOTE: 1/2/3/4/5/6    PLOT: P"
1090  PRINT "CLEAR NOTES: C   EXIT: E   SPEED: S"
1100  POKE  - 16368,0: WAIT  - 16384,128
1110  GET A$:I =  VAL (A$): IF I > 6 THEN  PRINT G
    $: GOTO 1000
1120  IF I = 0 THEN 1180
1130  VTAB 21: CALL  - 958: PRINT "NOTE ";I;": ";:
      INPUT "AMPLITUDE (1-10) ";A$:AMP(I) =  VAL (
    A$): IF AMP(I) = 0 THEN 1130
1140  IF AMP(I) > 20 THEN  PRINT G$;: GOTO 1130
```

```
1150  PRINT  TAB( 9):: INPUT "FREQUENCY - ";FR(I):
      IF FR(I) < 0 OR FR(I) > 99999 THEN  PRINT G$
     :: VTAB 22: CALL  - 868: GOTO 1150
1160 F(I) = FR(I) / 27.75
1170  GOTO 1000
1180  IF A$ = "E" THEN  END
1190  IF A$ = "P" THEN 2000
1200  IF A$ = "C" THEN 1240
1210  IF A$ <  > "S" THEN  PRINT G$: GOTO 1000
1220  VTAB 21: CALL  - 958: INPUT "ENTER SPEED (1-
     5) - ";SP: IF SP < 1 OR SP > 5 OR  INT (SP) <
      > SP THEN  PRINT G$: GOTO 1220
1230  GOTO 1000
1240  VTAB 21: CALL  - 958: PRINT "CLEAR ALL NOTES
      IN MEMORY? (Y/N) ": GET A$: IF A$ <  > "Y" THEN
     1000
1250  FOR I = 1 TO 6:F(I) = 0:FR(I) = 0:AMP(I) = 0
     : NEXT : GOTO 1000
1997 :
1998  REM  PLOTTING ROUTINE
1999 :
2000  HOME
2010  VTAB 24
2020  HGR
2030  HCOLOR= 3
2040  HPLOT 0,80 TO 279,80
2050  HPLOT 0,16 TO 0,143
2060  FOR I = 0 TO 279 STEP 70
2070  HPLOT I,78 TO I,82: HPLOT 279,78 TO 279,82
2080  NEXT I
2090  FOR I = 16 TO 144 STEP 16
2100  HPLOT 0,I TO 4,I
2110  NEXT I
2120 TIME = 400
2130 S = 280 / TIME
2140  HPLOT 0,80
2150  FOR I = 0 TO TIME STEP SP
2160 X = I * 3.14159 / 180
2170 Y = 0: FOR J = 1 TO 6:Y = AMP(J) / 5 *  SIN (
     F(J) * X) + Y: NEXT J
2180 Y = 80 - Y * 16
2190 X = I * S
2200  GOSUB 3000
2210  HPLOT  TO X,Y
2220  NEXT I
2230  POKE  - 16368,0: WAIT  - 16384,128
```

```
2240  GET A$
2250  GOTO 1000
2997 :
2998  REM  SUBROUTINE CHECK RANGE
2999 :
3000  IF X < 0 THEN X = 0
3010  IF X > 279 THEN X = 279
3020  IF Y < 0 THEN Y = 0
3030  IF Y > 159 THEN Y = 159
3040  RETURN
```

# Chapter 3

# Home Applications

# Introduction

Your Apple is a willing helper in everything from financial planning to weather forecasting, and the five programs in this chapter will help you put it to work.

For instance, George Miller's "Weather Forecaster" combines proven scientific principles with your Apple's computing power to make accurate local weather forecasts. It won't stop the rain, but at least you'll know when to carry your umbrella.

Counting calories? Gerald P. Graham's "Calorie Cop" will help you watch your waistline with computer precision.

"IRA Planner," by Richard and Betty Givan, shows you exactly how your IRA fund will grow. It can be of great help in planning your retirement nest egg.

David Swain's "Home Energy Calculator" can be useful, too, in checking to see which energy saving schemes will actually pay off. Then you can confirm your results with Larry L. Bihlmeyer's "Utility Bill Audit," a comprehensive program for evaluating utility bills.

# Weather Forecaster

George W. Miller
Apple Translation by Patrick Parrish

*The National Weather Service uses computers when forecasting the weather. With this program, you can use your Apple to turn out weather forecasts of your own.*

Everybody talks about the weather. "Weather Forecaster" won't let you do anything about it, but at least it will help you be prepared.

To use this program, you'll need a barometer (available at many hardware stores and department stores). You'll also need a weather vane or other device to indicate wind direction. If you don't have a weather vane, you can use a compass and observe the wind yourself.

## The Word on Weather

This program is based on sound scientific principles. In the Northern Hemisphere, winds blow counterclockwise around a low pressure system and clockwise around a high pressure system. Thus, if you stand outside with the wind at your back, a low pressure system will be on your left. If the barometer is falling, this low is heading in your direction.

By considering the wind direction along with changes in barometric pressure, you can get some idea of what kind of weather to expect. Your Apple can figure this out in a matter of seconds, and the program will even tell you what the normal weather for the month should be.

One word of warning. This is a very long program. Save it often as you type it in. It can be very frustrating to lose the program after several hours work due to a momentary power glitch.

## Using the Program

The Weather Forecaster menu offers a great deal of help. For example, you can store data in RAM, generate a weather forecast, display the data you have stored, display normal conditions for your area, STOP the program, search for a specific date, and make corrections—all directly from the menu.

Note that high and low temperatures are entered as four characters, one of which is a plus or minus sign (for instance, +076 or −012). To avoid the necessity of right justifying, each entry *must* have four characters. Barometric pressure is entered as a five-character entry, for instance, 30.15. General weather conditions are entered as a single digit, 1–7, from the following table:

1 = FAIR
2 = CLOUDY
3 = RAIN
4 = SNOW
5 = THUNDERSHOWERS
6 = SNOW FLURRIES
7 = HEAVY RAIN

Precipitation, in inches, must be input as a five-character entry (such as 02.75). Snowfall, in inches, must be given as a two character entry.

Wind direction takes the form of a four-character entry. The first two characters reflect wind direction and come from the following list:

| | |
|---|---|
| 01 = N | 05 = S |
| 02 = NE | 06 = SW |
| 03 = E | 07 = W |
| 04 = SE | 08 = NW |

The last two characters are wind speed in miles per hour, and speed and direction are combined to create a complete wind entry. For instance, you would use an entry of 0705 for wind from the west at five miles per hour.

You'll have to determine the average temperatures, rainfall, and snowfall amounts for your area. A good source of this information is *The Weather Almanac,* edited by James A. Ruffner and Frank E. Bair, published by Avon Books. It's available from most libraries. Look up the city nearest you and make your substitutions in lines 915–970.

The subroutine starting at line 3000 allows you to check the weather conditions on any day in your file. You enter the date in question, and the computer searches for that data. If the date is in memory, the computer will display the information.

This subroutine contains a disk error trapping routine (line 3200) that gives you the disk error number and the line

in the program where it has occurred. If a correctable disk error occurs, return to the main program and resave the data so that no data is lost.

## Weather Forecaster

```
5  GOTO 80
7  HOME : VTAB 5: HTAB 5: RETURN
10  PRINT P$;: INPUT B$: IF B$ = "" THEN 110
12  IF  LEN (B$) <  > B THEN  HOME : PRINT "INPUT
     MISTAKE": FOR I = 1 TO 500: NEXT I: PRINT P$;
     : INPUT B$
15  RETURN
20 A$(L) = A$(L) + B$: RETURN
80  DIM A$(365)
90  GOSUB 2000
100  REM  WEATHER FORECASTER
110  TEXT : HOME
120  HTAB 12: INVERSE : PRINT "WEATHER ANALYSIS": NORMAL
125  VTAB 5: HTAB 7: PRINT "TO LOAD DATA: ENTER ";
     : INVERSE : PRINT "L": NORMAL
130  VTAB 7: HTAB 7: PRINT "UPDATE DATA: ENTER ";:
      INVERSE : PRINT "U": NORMAL
140  VTAB 9: HTAB 7: PRINT "FORECAST FROM DATA: EN
     TER ";: INVERSE : PRINT "F": NORMAL
150  VTAB 11: HTAB 7: PRINT "DISPLAY DATA: ENTER "
     ;: INVERSE : PRINT "D": NORMAL
160  VTAB 13: HTAB 7: PRINT "DISPLAY NORMALS: ENTE
     R ";: INVERSE : PRINT "N": NORMAL
170  VTAB 15: HTAB 7: PRINT "TO MEMORIZE DATA: ENT
     ER ";: INVERSE : PRINT "M": NORMAL
180  VTAB 17: HTAB 7: PRINT "SEARCH DATE: ENTER ";
     : INVERSE : PRINT "S": NORMAL
190  VTAB 19: HTAB 7: PRINT "CORRECTIONS: ENTER ";
     : INVERSE : PRINT "C": NORMAL
195  VTAB 21: HTAB 7: PRINT "TO QUIT: ENTER ";: INVERSE
     : PRINT "Q": NORMAL
200  VTAB 23: HTAB 3: PRINT "CHOICE? ";: GET Y$
205  IF Y$ = "U" THEN 250
210  IF Y$ = "F" THEN 400
215  IF Y$ = "D" THEN 700
220  IF Y$ = "N" THEN 900
225  IF Y$ = "S" THEN 1250
230  IF Y$ = "C" THEN 1500
235  IF Y$ = "M" OR Y$ = "L" THEN 3000
240  IF Y$ = "Q" THEN 2500
245  GOTO 200
250  HOME : HTAB 16: VTAB 2: INVERSE : PRINT "DATA
      UPDATE": NORMAL
```

```
255  IF L = 365 THEN  PRINT "FILE FULL": FOR I = 1
      TO 2000: NEXT I: GOTO 110
260 L = L + 1
270  VTAB 5:P$ = "ENTER DATE (AS 01-05-83):":B = 8
     : GOSUB 10
275 A$(L) =  LEFT$ (B$,2) +  MID$ (B$,4,2) +  RIGHT$
     (B$,2)
285  PRINT :P$ = "ENTER HIGH TEMPERATURE (AS +076)
     : ":B = 4: GOSUB 10: GOSUB 20
290  PRINT :P$ = "ENTER LOW TEMPERATURE (AS -006):
      ": GOSUB 10: GOSUB 20
295  PRINT :P$ = "ENTER BAROMETRIC PRES. (IN INCHE
     S - AS  30.15): ":B = 5: GOSUB 10: GOSUB 20
300  PRINT : PRINT "ENTER GENERAL WEATHER CONDITIO
     N": PRINT "1 = FAIR";: HTAB 23: PRINT "2 = CL
     OUDY": PRINT "3 = RAIN";: HTAB 23: PRINT "4 =
      SNOW"
305  PRINT "5 = THUNDERSHOWERS";: HTAB 23: PRINT "
     6 = SNOW FLURRIES": PRINT "7 = HEAVY RAIN"
315 P$ = "":B = 1: GOSUB 10: GOSUB 20
325  PRINT :P$ = "ENTER PRECIPITATION (INCHES - AS
      02.75):":B = 5: GOSUB 10: GOSUB 20
335  PRINT :P$ = "ENTER SNOWFALL AMOUNT (AS 07): "
     :B = 2: GOSUB 10: GOSUB 20
345  HOME : PRINT "ENTER WIND DIRECTION AND SPEED:
      "
350  HTAB 3: PRINT "USE THIS CODE:": PRINT "01=N",
     "02=NE","03=E": PRINT "04=SE","05=S","06=SW"
355  PRINT "07=W","08=NW"
360  PRINT : PRINT "ENTER DIRECTION AND SPEED AS F
     OUR":P$ = "DIGIT NUMBER - AS 0312): ":B = 4: GOSUB
     10: GOSUB 20
365  HOME : VTAB 3: PRINT "DATE: "; LEFT$ (A$(L),6
     ): PRINT "HI TEMP: "; MID$ (A$(L),7,4)
370  PRINT "LOW TEMP: "; MID$ (A$(L),11,4): PRINT
     "BAROMETRIC PRESSURE: "; MID$ (A$(L),15,5)
375  PRINT "CONDITIONS: "; MID$ (A$(L),20,1): PRINT
     "PRECIPITATION: "; MID$ (A$(L),21,5)
380  PRINT "SNOWFALL: "; MID$ (A$(L),26,2): PRINT
     "WINDS: "; MID$ (A$(L),28,4)
385  HTAB 3: PRINT "RECALL THE CODE: ": PRINT "01=
     N","02=NE","03=E","04=SE","05=S","06=SW","07=
     W","08=NW"
387  IF D = 1 THEN  RETURN
390  VTAB 20: PRINT "IS THIS CORRECT (Y/N)? ";: GET
     B$: IF B$ = "N" THEN 270
392  IF C = 1 THEN  RETURN
395  GOTO 110
```

```
400  HOME : HTAB 17: VTAB 2: INVERSE : PRINT "FORE
     CAST": NORMAL
405  VTAB 5: INPUT "ENTER BAROMETRIC PRESSURE: ";A
     $:A =  VAL (A$): IF A$ = "" THEN 110
410  VTAB 7: PRINT "IS BAROMETER ?": PRINT : PRINT
     "1.STEADY","2.SLOW RISE": PRINT "3.RAPID RISE
     ","4.SLOW FALL"
412  PRINT "5.RAPID FALL": PRINT : PRINT "(RAPID C
     HANGE IS ANY CHANGE IN": PRINT "EXCESS OF 0.0
     6 PER HOUR.)"
415  INPUT B$:B =  VAL (B$): IF B$ = "" THEN 110
417  PRINT : PRINT "WIND FROM: ?"
420  PRINT "1=N","2=NE","3=E","4=SE","5=S","6=SW",
     "7=W","8=NW"
425  INPUT C$:C =  VAL (C$): IF C$ = "" THEN 110
430  IF A >  = 30.2 AND B = 4 AND C >  = 6 AND C <
      = 8 THEN 625
440  IF A >  = 30.2 AND B = 1 AND C >  = 6 AND C <
      = 8 THEN 620
445  IF A >  = 30.1 AND B = 1 AND C >  = 6 AND C <
      = 8 THEN 600
450  IF A >  = 30.1 AND B = 3 AND C >  = 6 AND C <
      = 8 THEN 605
455  IF A >  = 30.1 AND B = 4 AND C >  = 6 AND C <
      = 8 THEN 610
460  IF A >  = 30.1 AND B = 5 AND C >  = 6 AND C <
      = 8 THEN 615
465  IF A >  = 30.1 AND B = 4 AND (C = 4 OR C = 5)
      THEN 630
470  IF A >  = 30.1 AND (B = 4 OR B = 5) AND (C =
     4 OR C = 5) THEN 635
475  IF A >  = 30.1 AND (B = 4 OR B = 5) AND C >  =
     2 AND C <  = 4 THEN 645
485  IF A >  = 30.1 AND B = 4 AND (C = 2 OR C = 3)
      THEN 650
490  IF A >  = 30.1 AND B = 5 AND (C = 2 OR C = 3)
      THEN 655
492  IF A <  = 29.8 AND B = 5 AND C >  = 1 AND C <
      = 3 THEN 680
493  IF A <  = 29.8 AND B = 5 AND C >  = 3 AND C <
      = 5 THEN 675
494  IF A <  = 29.8 AND B = 3 THEN 685
495  IF A <  = 30.1 AND B = 4 AND C >  = 2 AND C <
      = 4 THEN 660
500  IF A <  = 30.1 AND B = 5 AND C >  = 2 AND C <
      = 4 THEN 665
505  IF A <  = 30.1 AND B = 2 AND (C = 5 OR C = 6)
      THEN 670
590  GOSUB 7: PRINT "LITTLE CHANGE FOR NEXT DAY
      OR TWO.": GOTO 690
```

```
600  GOSUB 7: PRINT "FAIR, LITTLE CHANGE IN TEMP":
      PRINT "FOR NEXT DAY OR TWO.": GOTO 690
605  GOSUB 7: PRINT "FAIR TODAY, RAINY AND WARMER"
     : PRINT "WITHIN 48 HOURS.": GOTO 690
610  GOSUB 7: PRINT "WARMER, RAIN WITHIN 24 TO 36"
     : PRINT "HOURS.": GOTO 690
615  GOSUB 7: PRINT "WARMER, RAIN WITHIN 18 TO 24"
     : PRINT "HOURS.": GOTO 690
620  GOSUB 7: PRINT "CONTINUED FAIR WITH LITTLE": PRINT
     "OR NO CHANGE IN TEMPERATURE.": GOTO 690
625  GOSUB 7: PRINT "FAIR AND WARMER FOR NEXT 48 H
     OURS.": GOTO 690
630  GOSUB 7: PRINT "RAIN WITHIN 24 HOURS.": GOTO
     690
635  GOSUB 7: PRINT "WINDY, WITH RAIN WITHIN": PRINT
     "12 TO 24 HOURS.": GOTO 690
640  GOSUB 7: PRINT "RAIN IN 12 TO 18 HOURS.": GOTO
     690
645  GOSUB 7: PRINT "WINDY AND RAIN WITHIN 12 HOUR
     S.": GOTO 690
650  GOSUB 7: PRINT "IN SUMMER WITH LIGHT WINDS: "
     : PRINT "RAIN MAY NOT FALL FOR DAYS.": PRINT
     "IN WINTER: ": PRINT "RAIN WITHIN 24 HOURS.":
      GOTO 690
655  GOSUB 7: PRINT "IN SUMMER: RAIN LIKELY WITHIN
     ": PRINT "12 TO 24 HOURS.": PRINT "IN WINTER:
      ": PRINT "RAIN OR SNOW WITH INCREASING WINDS
     .": GOTO 690
660  GOSUB 7: PRINT "RAIN FOR NEXT DAY OR TWO.": GOTO
     690
665  GOSUB 7: PRINT "RAIN WITH HIGH WINDS FOLLOWED
     ": PRINT "WITHIN 24 HOURS BY CLEARING AND COO
     LER": PRINT "TEMPERATURES.": GOTO 690
670  GOSUB 7: PRINT "CLEARING WITHIN A FEW HOURS."
     : PRINT "FAIR FOR NEXT SEVERAL DAYS.": GOTO 6
     90
675  GOSUB 7: PRINT "SEVERE STORM WARNING. WINDY,"
     : PRINT "WITH RAIN OR SNOW IMMINENT": PRINT "
     FOLLOWED WITHIN 24 HOURS BY CLEARING": PRINT
     "AND COLDER.": GOTO 690
680  GOSUB 7: PRINT "SEVERE STORM WARNING. SEVERE"
     : PRINT "NORTHEAST GALES, HEAVY RAIN OR SNOW,
     ": PRINT "FOLLOWED IN WINTER BY A COLD": PRINT
     "WAVE.": GOTO 690
685  GOSUB 7: PRINT "CLEARING AND COLDER."
690  HTAB 6: VTAB 23: INVERSE : PRINT "PRESS ANY K
     EY TO CONTINUE";: GET B$: NORMAL : GOTO 110
700 D = 1: HOME : IF L = 0 THEN 708
```

```
703 X = L: FOR I = 1 TO X:L = I: GOSUB 365: PRINT
    : PRINT "PRESS C TO CONTINUE ";: GET B$: IF B
    $ <  > "C" THEN I = X
705  PRINT : NEXT I:L = X:D = Ø: GOTO 11Ø
708 D = Ø: PRINT "NO DATA FOUND.": FOR T = 1 TO 2Ø
    ØØ: NEXT T: GOTO 11Ø
9ØØ  HOME : HTAB 15: INVERSE : PRINT "WEATHER NORM
    S": NORMAL
905  PRINT : HTAB 4: PRINT "TEMP"
91Ø  HTAB 2: PRINT "HI LO MO  RAIN   SNOW"
915  PRINT : PRINT "J 48 27 38  3.51   6.1": PRINT
    "F 51 29 4Ø  3.37   6.6"
92Ø  PRINT "M 59 37 48  3.88   2.5": PRINT "A 71 4
    6 59  3.16   Ø.3"
93Ø  PRINT "M 78 55 67  3.37   Ø.Ø": PRINT "J 84 6
    3 74  3.93   Ø.Ø"
94Ø  PRINT : PRINT "J 87 67 77  4.27   Ø.Ø": PRINT
    "A 86 66 76  4.19   Ø.Ø"
95Ø  PRINT "S 8Ø 59 7Ø  3.64   Ø.Ø": PRINT "O 7Ø 4
    7 59  3.18   Ø.1"
955  PRINT "N 6Ø 37 49  2.59   2.1": PRINT "D 5Ø 3
    Ø 4Ø  3.38   6.Ø"
96Ø  PRINT : PRINT "YR 63 43 53 37.96 35.4"
97Ø  HTAB 8: PRINT "DATA FROM NWS GBO.,N.C.": HTAB
    8: PRINT "ELEVATION 83Ø FT."
98Ø  PRINT : INVERSE : PRINT "PRESS ANY KEY TO CON
    TINUE";: GET B$: NORMAL : GOTO 11Ø
125Ø  HOME : HTAB 16: VTAB 2: INVERSE : PRINT "SEA
    RCH DATE": NORMAL :D = 1
126Ø  VTAB 5:P$ = "ENTER DATE (AS Ø1-Ø5-83): ":B =
    8: GOSUB 1Ø:C$ =  LEFT$ (B$,2) +  MID$ (B$,4,
    2)
127Ø X = L: FOR I = 1 TO L: IF C$ =  LEFT$ (A$(I),
    4) THEN T = I:I = L: NEXT I:L = T: GOSUB 365:
    L = X:D = Ø: GOTO 69Ø
1275  NEXT I: PRINT : PRINT "DATE NOT FOUND": GOTO
    69Ø
15ØØ  HOME : HTAB 16: VTAB 2: INVERSE : PRINT "COR
    RECTIONS": NORMAL :C = 1
151Ø  VTAB 5:P$ = "ENTER DATE (AS Ø1-Ø5-83) TO CHA
    NGE: ":B = 8: GOSUB 1Ø:C$ =  LEFT$ (B$,2) +  MID$
    (B$,4,2)
157Ø X = L: FOR I = 1 TO L: IF C$ =  LEFT$ (A$(I),
    4) THEN T = I:I = L: NEXT I:L = T: GOSUB 275:
    L = X:C = Ø: GOTO 69Ø
1575  NEXT I: PRINT : PRINT "DATE NOT IN FILE": GOTO
    69Ø
2ØØØ  TEXT : HOME
2Ø1Ø  HTAB 13: VTAB 1Ø: INVERSE : PRINT "WEATHER A
    NALYSIS": NORMAL : FOR I = 1 TO 1ØØØ: NEXT I
```

```
2020  HOME : HTAB 5: VTAB 3: PRINT "THIS PROGRAM I
      S DESIGNED TO STORE ON";
2030  PRINT "DISK A YEAR'S WORTH OF DATA IN THE DI
      SK": PRINT "FILE ENTITLED 'WEATHER FILE'.  IT
       IS"
2040  PRINT "SUGGESTED THAT FOR STORAGE OF MORE TH
      AN"
2050  PRINT "ONE YEAR OF DATA, A SEPARATE WEATHER"

2060  PRINT "RECORD BE MAINTAINED."
2080  HTAB 5: VTAB 11: PRINT "THIS PROGRAM WILL OF
      FER A FORECAST"
2090  PRINT "OF EXPECTED WEATHER CONDITIONS USING"

2100  PRINT "BAROMETRIC PRESSURE AND WIND DIRECTIO
      N."
2110  HTAB 5: VTAB 16: PRINT "YOU WILL BE SURPRISE
      D AT THE"
2120  PRINT "ACCURACY OF THE METHOD, YET IT IS BAS
      ED"
2130  PRINT "ON SOUND SCIENTIFIC PRINCIPLES."
2140  VTAB 22: HTAB 8: INVERSE : PRINT "PRESS ANY
      KEY TO CONTINUE";: NORMAL
2150  GET C$: HOME : RETURN
2500  HOME : VTAB 13: INPUT "HAVE YOU MEMORIZED TH
      E DATA (Y/N) ?";H$: IF H$ = "N" THEN 110
2510  END
3000  REM  APPLE DISK SAVE OR LOAD
3010  HOME : PRINT
3015  ONERR  GOTO 3200
3020 D$ =  CHR$ (4)
3022  PRINT D$;"OPEN WEATHER FILE": IF Y$ = "M" THEN
      3040
3024  PRINT D$;"READ WEATHER FILE"
3026  INPUT L: FOR I = 1 TO L: INPUT A$(I): NEXT I

3030  GOTO 3060
3040  PRINT D$;"WRITE WEATHER FILE"
3050  PRINT L: FOR I = 1 TO L: PRINT A$(I): NEXT I

3060  PRINT D$;"CLOSE WEATHER FILE": POKE 216,0: GOTO
      110
3200  HOME : VTAB 5: PRINT "ERROR # "; PEEK (222);
      " OCCURRED AT LINE "; PEEK (219) * 256 +  PEEK
      (218)
3210  VTAB 10: PRINT "HINT: HAVE YOU PREVIOUSLY SA
      VED THE": PRINT "DATA FILE TO DISK?"
3220  PRINT D$;"CLOSE WEATHER FILE": POKE 216,0: GOTO
      690
```

# IRA Planner

Richard and Betty Givan

*You've seen the bank ads: "Retire a Millionaire." Type in this short program and see for yourself how IRA accounts compute. The program uses very little memory. memory.*

Most get-rich schemes have proven to be of questionable legality and dubious worth. The latest promotion, however, is endorsed by the U.S. Government and seems foolproof. It's the Individual Retirement Account (IRA), expanded in 1982 to allow up to a $2000 ($2250 in a joint plan with a nonworking spouse) yearly deposit to be put into a private retirement account.

This amount is deductible from the person's gross income during the year deposited, decreasing the income tax accordingly. The retirement fund is then free to grow at the prevailing competitive interest rate—compounded daily and tax free—until it is withdrawn during retirement. Although taxes are then due, presumably the taxpayer will be in a lower tax bracket at that time and thus will have to pay a lesser tax.

## The Relationship Between Inflation and Interest

The allure of the plan lies in the rapid growth of the principal through compound interest at the current high rates. That's why you see ads in which banks all but guarantee that you can be a millionaire upon retirement via a $2000 yearly deposit for 35 years at a 12 percent return. Actually, your account *would* be worth an astounding $1,161,059. Who would have thought that such a modest sacrifice would let you retire a millionaire!

As with all get-rich plans, however, there is a catch—but in this case it's a matter of economics. The IRA promotion campaigns conveniently overlook the devastating effects of inflation on your million dollar nest egg. At the same time that compound interest is building your fortune, inflation is eroding it. Historically, the interest rate is fairly well dictated by the rate of inflation. Although temporary imbalances occur, economists generally agree that, in the long run, the interest rate will seek out a level approximately 3 to 4 percent higher than the inflation rate.

If the rate of inflation were to stay at 9 percent, for example, your retirement fortune of $1,161,059 would really be worth only $56,875 in terms of 1982 dollars. You may have a carload of dollars in the year 2017, but the Cadillac you buy to haul them home would cost $306,000 and the gasoline to power it would be $25 a gallon!

This is not to say that an IRA is a bad way to save. It does offer immediate tax relief, and that in itself might provide you with the incentive to put aside some funds for your golden years. But it would be well to put the numbers in perspective when planning for your future.

The program asks you several questions: the amount of money you wish to set aside each year; the tax bracket you are currently in (which can be found by reference to the IRS booklet accompanying your tax forms, but is not really essential to the rest of the program); your age when you begin and end the plan; and the average interest and inflation rates you expect to experience.

The program then displays the tax savings you would receive the first year in the plan. (Your income and tax rate would probably fluctuate too much to benefit from attempting to compute these over the life of your IRA.) The sum of your deposits is displayed, followed by the principal of the account increased by accumulated interest. Then the *real spending power* of your final nest egg is shown by reducing the principal to reflect the inflation rate. You can see its worth in terms of the 1982 dollar. Bear in mind that this money is taxable when withdrawn, too.

One note: The two questions about inflation and interest ask for the figures *expressed as decimals.* For example, if you want to calculate 12 percent inflation, you should type .12. Similarly, 6 percent interest would be entered as .06.

## IRA Planner

```
20  HOME
40  PRINT "AT WHAT AGE DO YOU PLAN"
50  INPUT "TO OPEN AN IRA ACCOUNT? ";A
60  VTAB 8: INPUT "AT WHAT AGE DO YOU PLAN TO RETI
    RE? ";AI
70 Y = AI - A
80  VTAB 16: PRINT "HOW MUCH DO YOU PLAN TO"
```

```
90  INPUT "DEPOSIT EACH YEAR? ";D:C = D
100  HOME : PRINT "WHAT IS YOUR TAX BRACKET?"
110  INPUT "(ENTER % AS DECIMAL EX. 30% AS .30) ";
     P
120  VTAB 8: PRINT "WHAT IS THE AVERAGE INTEREST R
     ATE YOU"
130  PRINT "EXPECT FOR THE ACCOUNT OVER THE "
140  INPUT "YEARS IT EXISTS?(% AS DECIMAL) ";R
160  VTAB 16: PRINT "WHAT IS THE AVERAGE INFLATION
      RATE"
170  PRINT "YOU EXPECT DURING THE YEARS BETWEEN"
180  PRINT "OPENING THE ACCOUNT AND RETIREMENT"
190  INPUT "(% AS DECIMAL)  ";I
200 S = D * P
210  HOME : PRINT "YOU WILL SAVE $"; INT (S);" ON
     TAXES THIS YEAR."
215  VTAB 8: PRINT "<MORE>"
220 T = D * Y
230  GET R$: IF R$ = "" THEN 230
235  HOME : PRINT "THE TOTAL AMOUNT DEPOSITED INTO
      YOUR    ACCOUNT OVER ";Y:
237  PRINT " YEARS IS": PRINT "$"; INT (T);"."
250  FOR J = 1 TO Y
260 X = D * (1 + R / 365) ^ 365
270 D = X + C
280  NEXT J
290  PRINT "<MORE>"
295  GET R$: IF R$ = "" THEN 295
300  PRINT "WHEN YOU RETIRE, THE AMOUNT IN YOUR"
310  PRINT "ACCOUNT WILL BE $"; INT (X);"."
315 Z = (1 + I) ^ Y
320 W = X / Z
330  VTAB 12: PRINT "WHICH IS WORTH $"; INT (W);"
     IN 1984 DOLLARS"
340  PRINT "TRY AGAIN?"
350  GET R$: IF R$ = "" THEN 350
360  IF R$ = "Y" THEN  RUN
370  HOME : PRINT "BYE"
```

# Home Energy Calculator

David Swaim
Apple Version by David Florance

*You (and your computer) may become extremely popular when your neighbors learn that you can analyze the energy-saving benefits of home improvements.*

In recent years, there has been a great deal of interest in saving energy in the home. No one needs to be reminded that fuel costs are rising, and everyone wants to reduce energy bills. The way to do this is simple: reduce household energy consumption. There are a number of ways this can be done.

You can change your habits (for instance, by setting the thermostat back to a lower temperature and wearing heavier clothes) or you can add more effective insulation to your home. Most people find it easier to follow the latter course. But which improvements will save you the most money? Which will cost the least to implement? Finally, which will yield the greatest savings for the least amount of cost? That last question is the important one, and this program will give you the answer.

## Predicting Effectiveness

If you know the weather as well as the heat loss characteristics of a house, you can estimate heating costs. By calculating the heating costs based on heat loss characteristics of the house both before and after the improvements, you can figure the estimated savings due to the improvements. That is what this program does.

To gather the preliminary data, you will need to make some measurements and observations. The first thing the program calculates is the heat loss of the house, and that depends on three things: the thermal resistance (R-value) of the structure; the total area of the structure exposed to the elements; and the temperature difference between the inside and outside of the house. You simply need to know the area, R-value, and the difference in temperature.

The only problem is that different parts of the house have different R-values. For example, windows will have a lower R-value than walls. However, you can divide the external area of the house into five categories (windows, doors, walls, ceiling,

and floor). The program requests information on each of these five categories in turn.

For windows it requests height, width, and number of windows, as well as type of frame and number of layers of glass. The number of types and/or sizes of windows is requested first. Most houses will have several sizes of windows, and there may be storm windows on some and not on others. The program allows for up to ten different types and/or sizes of windows. If you need more, change the dimension of S in statement 180.

Only one size and type of door is allowed. If you have sliding glass doors, you should consider them as windows. You'll need to know the height, width, and number of doors. Remember: Consider exterior doors only.

Information needed concerning the external walls consists of two things: the type of construction and the R-value of the insulation in the wall. If you enter a negative number for the R-value of the wall insulation, the program will give you a list of typical R-values for wall insulation. To get the area of the wall, the program asks for the ceiling height, total perimeter of the house, and the number of stories in the house. The program will calculate the gross wall area from this data and subtract the total window and door area to obtain the proper wall area.

## Measurements

The only items you have to find for yourself concern the ceiling and floor. For the ceiling, you will be asked for the number of inches of insulation in the attic and the type of insulating material. For the floor, the type of foundation is requested.

Additional heat loss comes from two other sources. The first is infiltration of outside air through cracks in windows and doors. The program asks if the windows and doors are weather-stripped. It uses this information (and the total length of the cracks around windows and doors) to calculate the total infiltration.

The second source of heat loss is via the heat ducts running from the furnace to the heat registers. The program asks if your heat ducts are insulated and where they are located.

That concludes the input needed for calculating the total heat loss of the house. At that point the heat losses are dis-

played, and you are asked if you wish to make improvements to the house.

If the answer is Y, you will be asked if you wish to improve each item. You can make improvements to a single item or to any number of items.

As you probably noticed, the first question you were asked is what the outside design temperature is. The outside design temperature for my area (Atlanta, Georgia) is 23 degrees. The outside design temperatures for other areas are tabulated in Table 1. For a more complete list, consult one of the references listed at the end of this article.

## Table 1: Winter Design Temperatures

| CITY | TEMPERATURE |
|---|---|
| MONTGOMERY AL | 26 |
| JUNEAU AK | −4 |
| PHOENIX AZ | 34 |
| LITTLE ROCK AR | 23 |
| SACRAMENTO CA | 32 |
| DENVER CO | 3 |
| HARTFORD CT | 5 |
| DOVER DE | 15 |
| TALLAHASSEE FL | 29 |
| ATLANTA GA | 23 |
| BOISE ID | 10 |
| SPRINGFIELD IL | 4 |
| INDIANAPOLIS IN | 4 |
| DES MOINES IA | −3 |
| TOPEKA KS | 6 |
| LEXINGTON KY | 10 |
| BATON ROUGE LA | 30 |
| AUGUSTA ME | −3 |
| BALTIMORE MD | 20 |
| BOSTON MA | 10 |
| LANSING MI | 6 |
| ST. PAUL MN | −10 |
| JACKSON MS | 24 |
| JEFFERSON CITY MO | 6 |
| HELENA MT | −13 |
| LINCOLN NE | 0 |
| CARSON CITY NV | 7 |
| CONCORD NH | −7 |
| TRENTON NJ | 16 |
| SANTA FE NM | 11 |

| | |
|---|---|
| ALBANY NY | 5 |
| RALEIGH NC | 20 |
| BISMARCK ND | −19 |
| COLUMBUS OH | 7 |
| OKLAHOMA CITY OK | 15 |
| SALEM OR | 25 |
| HARRISBURG PA | 13 |
| PROVIDENCE RI | 10 |
| COLUMBIA SC | 23 |
| PIERRE SD | −9 |
| NASHVILLE TN | 16 |
| AUSTIN TX | 29 |
| SALT LAKE CITY UT | 9 |
| BURLINGTON VT | −7 |
| RICHMOND VA | 18 |
| OLYMPIA WA | 25 |
| CHARLESTON WV | 14 |
| MADISON WS | −5 |
| CHEYENNE WY | −2 |

Actually, you do not need to enter any specific temperature as long as it is less than 75 degrees, the inside design temperature used by the program. The program will still give you valid results for savings and payback. However, using the correct outside design temperature gives you the advantage of seeing what the necessary furnace size would be with and without the improvements. In fact, heating engineers use the same basic method to size furnaces for houses.

When the program finishes calculating the heat loss of the house *after* improvements, it is ready to do the cost analysis. First you are asked for the type of heating fuel you use: electricity, fuel oil, or natural gas. Next you must input the cost per fuel unit of the heating fuel. Note that this unit cost is in dollars, so if natural gas in your area is 25 cents per therm, you should input .25 dollars per therm.

Using this data, and the number of heating degree days for your area, the program calculates the total energy needed to heat the house for the entire heating season. The degree days and name of the city are part of line 7010, and you should change this line to reflect your own location. Some sample degree days for different cities are listed in Table 2; a more complete list can be found in any of the references.

## Table 2: Yearly Heating Degree Days

| CITY | DEGREE DAYS |
|---|---|
| MONTGOMERY AL | 2291 |
| JUNEAU AK | 9075 |
| PHOENIX AZ | 1765 |
| LITTLE ROCK AR | 3219 |
| SACRAMENTO CA | 2419 |
| DENVER CO | 5524 |
| HARTFORD CT | 6235 |
| WILMINGTON DE | 4930 |
| TALLAHASSEE FL | 1485 |
| ATLANTA GA | 2961 |
| BOISE ID | 5809 |
| SPRINGFIELD IL | 5429 |
| INDIANAPOLIS IN | 5699 |
| DES MOINES IA | 6588 |
| TOPEKA KS | 5182 |
| LEXINGTON KY | 4683 |
| BATON ROUGE LA | 1560 |
| PORTLAND ME | 7511 |
| BALTIMORE MD | 4111 |
| BOSTON MA | 5634 |
| LANSING MI | 6909 |
| MINNEAPOLIS MN | 8382 |
| JACKSON MS | 2239 |
| ST. LOUIS MO | 4484 |
| HELENA MT | 8129 |
| LINCOLN NE | 5864 |
| RENO NV | 6332 |
| CONCORD NH | 7383 |
| TRENTON NJ | 4980 |
| ALBUQUERQUE NM | 4348 |
| ALBANY NY | 6201 |
| RALEIGH NC | 3393 |
| BISMARCK ND | 8851 |
| COLUMBUS OH | 5211 |
| OKLAHOMA CITY OK | 3725 |
| SALEM OR | 4754 |
| HARRISBURG PA | 5251 |
| PROVIDENCE RI | 5954 |
| COLUMBIA SC | 2484 |
| RAPID CITY SD | 7345 |
| NASHVILLE TN | 3578 |
| AUSTIN TX | 1711 |

| | |
|---|---|
| SALT LAKE CITY UT | 6052 |
| BURLINGTON VT | 8269 |
| RICHMOND VA | 3865 |
| OLYMPIA WA | 5236 |
| CHARLESTON WV | 4476 |
| MADISON WS | 7863 |
| CHEYENNE WY | 7381 |

The last thing you must input is the total cost of the improvements you made. At that point, the program calculates the payback period in years.

I learned a great deal about my own home from running this program. Much of what I concluded was what I expected, but some conclusions surprised me. The program can definitely help home owners in assessing home energy improvements; it can also help you spot dishonest energy-saving schemes pretty quickly.

References

*ASHRAE Handbook 1981 Fundamentals.* Atlanta, Georgia: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Incorporated, 1981.

*Other Homes and Garbage.* Jim Leckie, Gil Masters, Harry Whitehouse, and Lilly Young. San Francisco, California: Sierra Club Books, 1975.

*Refrigeration and Air-Conditioning.* Air-Conditioning and Refrigeration Institute. Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

## Home Energy Calculator

```
150  GOSUB 8000
170  REM  INITIALIZATION
180  DIM A(6),Q(6),R(6),RW(4,3),D(4),IW(2,3),S(10)
190  DIM RF(3),TC(3),N$(5),IC(5),DM(2,3,3),IN(2)
200  REM  WINDOW R VALUES
210  DATA  1.01,2.22,1.815,3.155
220  DATA  .909,1.667,1.437,2.137
230  DATA  .909,2,1.724,2.564
240  REM  DOOR R VALUES
250  DATA  .41,.75,.95,1.1
260  REM  FLOOR R VALUES AND TEMP CORR
270  DATA  3.2,0,3.2,30,1.23,0
280  REM  CEILING INSULATION R PER INCH
```

```
290  DATA  3.5,3,2.5,4.5,5.5
300 N$(1) = "WINDOWS":N$(2) = "DOORS":N$(3) = "WAL
    LS"
310 N$(4) = "CEILING":N$(5) = "FLOOR "
320  REM  DUCT MULTIPLIERS
330  DATA  .2,.15,.1,.15,.1,.05,.1,.05,.05
340  DATA  .2,.15,.1,.1,.1,.05,.05,.05,.05
350  REM  AIR CHANGES PER FOOT OF CRACK
360  DATA  39,74,52,24,32,33
370  REM  READ WINDOW R VALUES
380  FOR F = 1 TO 3
390  FOR G = 1 TO 4
400  READ RW(G,F)
410  NEXT G,F
420  REM  READ DOOR R VALUES
430  FOR I = 1 TO 4: READ D(I): NEXT I
440  REM  READ FLOOR R VAL AND TEMP CORR
450  FOR I = 1 TO 3: READ RF(I),TC(I): NEXT I
460  REM  READ INSULATION R PER INCH
470  FOR I = 1 TO 5: READ IC(I): NEXT I
480  REM  READ DUCT MULTIPLIERS
490  FOR KD = 1 TO 2
500  FOR K = 1 TO 3
510  FOR J = 1 TO 3
520  READ DM(KD,J,K)
530  NEXT J,K,KD
540  REM  READ AIR CHANGES FOR INFILTRATION
550  FOR I = 1 TO 2
560  FOR J = 1 TO 3
570  READ IW(I,J)
580  NEXT J,I
590  REM  INSIDE DESIGN TEMPERATURE
600 IT = 75:PK = 1
610  PRINT "WINTER OUTSIDE DESIGN TEMPERATURE";
620  INPUT OT
630 DT = IT - OT
640  GOSUB 1000: REM    WINDOWS
650  GOSUB 2000: REM    DOORS
660  GOSUB 3000: REM    WALLS
670  GOSUB 4000: REM    CEILING
680  GOSUB 5000: REM    FLOOR
690  GOSUB 5200: REM    DUCTS
700  GOSUB 6000: REM    REPORT RESULTS
710 Q1 = TQ / DT
720  PRINT "DO YOU WISH TO MAKE IMPROVEMENTS";
730  INPUT A$
740 PK = 2: IF  LEFT$ (A$,1) <  > "Y" THEN 999
750  HOME : INPUT "DO YOU WISH TO IMPROVE WINDOWS?
    (Y/N)";A$
```

```
760  IF  LEFT$ (A$,1) = "Y" THEN  GOSUB 1000
770  HOME : INPUT "DO YOU WISH TO IMPROVE DOORS?(Y
     /N)";A$
780  IF  LEFT$ (A$,1) = "Y" THEN  GOSUB 2000
790  HOME : INPUT "DO YOU WISH TO IMPROVE WALLS?(Y
     /N)";A$
800  IF  LEFT$ (A$,1) = "Y" THEN  GOSUB 3000
810  HOME : INPUT "DO YOU WISH TO IMPROVE CEILING?
     (Y/N)";A$
820  IF  LEFT$ (A$,1) = "Y" THEN  GOSUB 4000
830  HOME : INPUT "DO YOU WISH TO IMPROVE FLOOR?(Y
     /N)";A$
840  IF  LEFT$ (A$,1) = "Y" THEN  GOSUB 5000
850  HOME : INPUT "DO YOU WISH TO IMPROVE DUCTS?(Y
     /N)";A$
860  IF  LEFT$ (A$,1) = "Y" THEN  GOSUB 5200
870  GOSUB 6000: REM  REPORT RESULTS
880 Q2 = TQ / DT
890  PRINT : PRINT "HIT RETURN TO GET SAVINGS"
900  GET A$: IF A$ = "" THEN 900
910  GOSUB 7000: REM  CALCULATE A YEAR OF SAVINGS
999  END
1000  REM  WINDOW SUBROUTINE
1010 I = 1: IF PK > 1 THEN 1040
1020  HOME : PRINT "HOW MANY DIFFERENT SIZES OF WI
     NDOWS";
1030  INPUT NX
1040 IX = 1:CW = 0:A(I) = 0:Q(I) = 0
1050  PRINT : INPUT "ARE WINDOWS WEATHERSTRIPPED?(
     Y/N)";WW$
1070  IF  LEFT$ (WW$,1) = "Y" THEN IX = 2: GOTO 10
     80
1075  IF  LEFT$ (WW$,1) = "N" THEN 1080
1077  HOME : INVERSE : PRINT "Y";: NORMAL : PRINT
     "ES OR ";: INVERSE : PRINT "N";: NORMAL : PRINT
     "O PLEASE"
1078  GOTO 1050
1080  FOR J = 1 TO NX
1090  PRINT "SIZE ";J: IF PK > 1 THEN 1160
1100  PRINT "NUMBER OF WINDOWS";
1110  INPUT NW
1120  PRINT "SIZE OF WINDOWS (H,W) FT";
1130  INPUT H,W
1140 S(J) = H * W * NW
1150 CW = CW + (H + W) * NW
1160 A(I) = A(I) + S(J)
1170  PRINT "TYPE OF WINDOWS"
1180  PRINT "    1. SINGLE GLASS"
1190  PRINT "    2. SINGLE + STORM"
```

```
1200  PRINT "    3. DOUBLE PANE"
1210  PRINT "    4. TRIPLE (DOUBLE + STORM)"
1220  INPUT G
1230  PRINT "TYPE OF WINDOW FRAME"
1240  PRINT "    1. WOOD"
1250  PRINT "    2. METAL OR JALOUSIE"
1260  PRINT "    3. FIXED"
1270  INPUT F
1280 RM = RW(G,F)
1290 Q(I) = Q(I) + S(J) * DT / RM
1300 R(I) = RM
1310  PRINT "";
1320  NEXT J
1330 IN(I) = 0.018 * DT * IW(IX,F) * CW
1340  RETURN
2000  REM  DOORS SUBROUTINE
2010 I = 2: IF PK > 1 THEN 2080
2020  HOME : PRINT "NUMBER OF DOORS";
2030  INPUT N
2040  PRINT "SIZE OF DOORS (H,W) FT";
2050  INPUT H,W
2060 A(I) = H * W * N
2070 CD = (H + W) * N
2080  PRINT "TYPE OF DOORS"
2090  PRINT "    1. WOOD"
2100  PRINT "    2. WOOD + STORM"
2110  PRINT "    3. METAL URETHANE CORE"
2120  PRINT "    4. METAL POLYSTYRENE CORE"
2130  INPUT T
2140 R(I) = D(T)
2150 Q(I) = A(I) * DT / R(I)
2160 DW = 138
2170  INPUT "ARE DOORS WEATHERSTRIPPED?(Y/N)";DW$
2190  IF  LEFT$ (DW$,1) = "Y" THEN DW = 69: GOTO 2
     200
2194  IF  LEFT$ (DW$,1) = "N" THEN 2200
2196  HOME : INVERSE : PRINT "Y";: NORMAL : PRINT
     "ES OR ";: INVERSE : PRINT "N";: NORMAL : PRINT
     "O PLEASE"
2198  GOTO 2170
2200 IN(I) = 0.018 * DT * DW * CD
2210  RETURN
3000  REM  WALLS SUBROUTINE
3010 I = 3:HO = .17:HI = .68
3020  HOME : PRINT "TYPE OF WALL CONSTRUCTION"
3030  PRINT "    1. BRICK VENEER"
3040  PRINT "    2. STONE"
3050  PRINT "    3. WOOD SHINGLES"
3060  PRINT "    4. STUCCO"
```

```
3070  PRINT "    5. MASONRY BLOCK"
3080  PRINT "    6. LOG"
3090  PRINT "    7. OTHER:"
3100  PRINT "       ENTER CALCULATED R VALUE DIRECT
     LY"
3110  PRINT "       WHEN ASKED FOR INSULATION R VAL
     UE"
3120  INPUT TY
3130  ON TY GOTO 3140,3150,3160,3170,3180,3190,320
     0
3140 RM = .2 * 3.5: GOTO 3210: REM   BRICK
3150 RM = .08 * 5: GOTO 3210: REM   STONE
3160 RM = .87: GOTO 3210: REM   WOOD
3170 RM = .2 * 2: GOTO 3210: REM   STUCCO
3180 RM = 2: GOTO 3210: REM   MASONRY
3190 RM = 1.25 * 8: GOTO 3210: REM   LOG
3200 RM = 0: REM                    OTHER
3210  PRINT "   FOR LIST OF R VALUES FOR INSULATIO
     N"
3220  PRINT "   ENTER -1 FOR INSULATION R VALUE"
3230  PRINT "INSULATION R VALUE";
3240  INPUT RI
3250  IF RI < 0 THEN  GOSUB 3500: GOTO 3230
3260 R(I) = HO + RM + RI + HI: IF PK > 1 THEN 3340

3270  HOME : PRINT "HOW MANY STORIES IN HOUSE";
3280  INPUT NT
3290  PRINT "WHAT IS THE CEILING HEIGHT (FT)";
3300  INPUT CH
3310  PRINT "WHAT IS TOTAL PERIMETER (FT)";
3320  INPUT P
3330 A(I) = NT * CH * P - A(1) - A(2)
3340 Q(I) = A(I) * DT / R(I)
3350  RETURN
3500  REM  LIST OF INSULATION R VALUES
3510  PRINT "LIST OF INSULATION R VALUES, WALLS"
3520  PRINT "         NO INSULATION (AIR) = .94"
3530  PRINT "     BATT INSULATION IN WALL = 11"
3540  PRINT "     HALF INCH ASPHALT BOARD = 2.4
3550  PRINT "    1/2 IN GYPSUM OR PLASTER = 1.39
3560  PRINT "     1/4 IN WOOD FIBER BOARD = 1.12
3570  PRINT "       FIR OR PINE SHEATHING = 1.92
3580  PRINT "       3/4 IN PLYWOOD PANELS = 1.88
3590  PRINT "              1/2 IN PLYWOOD = 1.57
3600  PRINT : PRINT
3610  RETURN
4000  REM  CEILING ROUTINE
4010 I = 4
4020 HI = .61:HO = .61: IF PK > 1 THEN 4060
4030  HOME : PRINT "WHAT IS TOTAL CEILING AREA"
```

```
4040  PRINT "OF THE HOUSE";
4050  INPUT A(I)
4060  PRINT "HOW MANY INCHES OF INSULATION IN CEIL
     ING";
4070  INPUT CI
4080  PRINT "TYPE OF INSULATING MATERIAL"
4090  PRINT "  1. FIBERGLASS"
4100  PRINT "  2. MINERAL WOOL"
4110  PRINT "  3. VERMICULITE OR PERLITE"
4120  PRINT "  4. CELLULOSE FIBER"
4130  PRINT "  5. U-F FOAM"
4140  INPUT T
4150 RM = CI * IC(T)
4160 R(I) = HO + RM + HI
4170 Q(I) = A(I) * DT / R(I)
4180  RETURN
5000  REM  FLOOR ROUTINE
5010 I = 5: IF PK > 1 THEN 5040
5020  HOME : PRINT "WHAT IS TOTAL FLOOR AREA";
5030  INPUT A(I)
5040  PRINT "HOW MANY INS OF INSULATION IN FLOOR";

5050  INPUT FI: IF PK > 1 THEN 5110
5060  PRINT "TYPE OF FOUNDATION"
5070  PRINT "    1. OPEN CRAWLSPACE"
5080  PRINT "    2. ENCLOSED CRAWLSPACE OR BASEMENT
     "
5090  PRINT "    3. CONCRETE SLAB"
5100  INPUT TF
5110 R(I) = HO + FI * 3.1 + RF(TF) + HI
5120 Q(I) = A(I) * (DT - TC(TF)) / R(I)
5130  RETURN
5200  REM  DUCTS
5210 DI = .1
5220  IF TF = 3 THEN KD = 3: RETURN
5230  INPUT "IS YOUR DUCTWORK INSULATED?(Y/N)";D$
5240  IF PK > 1 THEN 5310
5245  IF  VAL (D$) <  > 0 THEN  HOME : INVERSE : PRINT
     "Y";: NORMAL : PRINT "ES OR";: INVERSE : PRINT
     " N";: NORMAL : PRINT "O PLEASE": GOTO 5230
5247  IF D$ <  > "Y" AND D$ <  > "N" THEN 5230
5250  PRINT "LOCATION OF HEAT DUCTS:"
5260  PRINT "     1. ATTIC OR CRAWLSPACE"
5270  PRINT "     2. UNCONDITIONED BASEMENT"
5280  PRINT "     3. IN SLAB FLOOR"
5290  PRINT "     4. INSIDE CONDITIONED SPACE"
5300  INPUT KD
5310  RETURN
6000  REM  WRITE A REPORT
```

```
6010  HOME : VTAB 1: HTAB 11: PRINT "HEAT LOSS EVA
     LUATION"
6020  PRINT : PRINT :TQ = 0
6030  PRINT " ITEM";: HTAB 10: PRINT "AREA";: HTAB
     20: PRINT "R-VAL";: HTAB 30: PRINT "HEAT LOSS
     "
6040  HTAB 8: PRINT " SQ.FT.";"                BTU/
     HR": PRINT
6050  FOR I = 1 TO 5
6060 A(I) =  INT (A(I) * 100 + .5) / 100
6070 R(I) =  INT (R(I) * 100 + .5) / 100
6080 Q(I) =  INT (Q(I) + .5)
6090  PRINT N$(I);: HTAB 10: PRINT A(I);: HTAB 20:
      PRINT R(I);: HTAB 30: PRINT Q(I)
6100 TA = TA + A(I):TQ = TQ + Q(I)
6110  NEXT I
6120  REM  PRINT INFILTRATION LOSS
6130  PRINT "INFILTRATION";: HTAB 30: PRINT  INT (
     (IN(1) + IN(2)) / 2 + .5)
6140 TQ = TQ + (IN(1) + IN(2)) / 2
6150  REM  CALCULATE DUCT LOSS
6160 X = TQ / (A(5) * CH * NT):J = 3:K = 3
6170  IF X < 45 THEN K = 2
6180  IF X < 35 THEN K = 1
6190 DI = .15 + .05 * (3 - K)
6200  IF  LEFT$ (D$,1) = "N" AND KD < 2 THEN 6240
6205  IF KD > 2 THEN DI = 0: GOTO 6240
6210  IF OT < 15 THEN J = 2
6220  IF OT < 0 THEN J = 1
6230 DI = DM(KD,J,K)
6240  PRINT "DUCT LOSS";: HTAB 30: PRINT  INT (DI *
     TQ + .5)
6250 TQ = TQ + TQ * DI
6260  HTAB 10: PRINT "--------";: HTAB 30: PRINT "
     --------"
6270  PRINT "TOTAL";: HTAB 10: PRINT  INT (TA);: HTAB
     30: PRINT  INT (TQ)
6272  PRINT : PRINT : PRINT "<MORE>"
6275  GET R$: IF R$ = "" THEN 6275
6280  PRINT : PRINT
6290  PRINT "DESIGN CONDITIONS:"
6300  PRINT "   OUTSIDE DESIGN TEMP";,OT
6310  PRINT "    INSIDE DESIGN TEMP";,IT
6320  PRINT "TEMPERATURE DIFFERENCE";,DT
6330  RETURN
7000  REM  FIND SAVINGS USING DEGREE-DAYS
7010 DD = 2961:DD$ = "MODERATE CLIMATE"
7012 E1 =  INT (Q1 * DD * 24)
7014 E2 =  INT (Q2 * DD * 24)
```

```
7030  PRINT "TYPE OF HEATING FUEL USED"
7040  PRINT "  1. ELECTRICITY"
7050  PRINT "  2. NATURAL GAS"
7060  PRINT "  3. FUEL OIL"
7070  INPUT FT:PC = .55
7080  ON FT GOTO 7100,7200,7300
7090  GOTO 7030
7100  REM  ELECTRICITY
7110  HOME : INPUT "IS HEATING UNIT A HEAT PUMP?(Y
     /N)";HP$:ER = 3413
7130  IF  LEFT$ (HP$,1) <  > "Y" THEN 7150
7140  INPUT "ENTER EER OF HEAT PUMP...";ER:ER = ER
      * 1000
7150  INPUT "AVERAGE $ COST PER KWH--$";CO:FU$ = "
     KWH"
7160 E1 =  INT (E1 / ER + .5)
7165 M1 = E1 * CO
7170 E2 =  INT (E2 / ER + .5)
7175 M2 = E2 * CO
7180 MS = M1 - M2
7190  GOTO 7400
7200  REM  NATURAL GAS
7210  HOME : INPUT "AV.$ COST PER THERM OF NAT. GA
     S--- $";CO
7220 E1 =  INT (E1 / (103000 * PC) + .5)
7225 M1 = E1 * CO
7230 E2 =  INT (E2 / (103000 * PC) + .5)
7235 M2 = E2 * CO
7240 MS = M1 - M2
7250 FU$ = "THERMS": GOTO 7400
7300  REM  FUEL OIL
7310  HOME : INPUT "AVERAGE $ COST PER GAL.OF FUEL
      OIL----  $";CO
7320 E1 =  INT (E1 / (138000 * PC) + .5)
7325 M1 = E1 * CO
7330 E2 =  INT (E2 / (138000 * PC) + .5)
7335 M2 = E2 * CO
7340 MS = M1 - M2:FU$ = "GALLONS"
7400  REM  GIVE RESULTS
7410 M1 =  INT (M1 * 100) / 100
7420 M2 =  INT (M2 * 100) / 100
7430 MS =  INT (MS * 100) / 100
7440  HOME : INPUT "TOT.$ COST OF IMPROVEMENTS--$"
     ;CI
7450 PB =  INT (CI / MS * 1000) / 1000
7460  REM  REPORT SAVINGS AND PAYBACK
7470  HOME : VTAB 1: HTAB 11: PRINT "ANALYSIS OF I
     MPROVEMENTS"
7480  PRINT : PRINT
```

```
7490  HTAB 24: PRINT "ENERGY NEEDED"
7500  HTAB 1: PRINT "ORIGINAL HOUSE";: HTAB 27: PRINT
     E1;: PRINT " ";FU$
7510  HTAB 1: PRINT "IMPROVED HOUSE";: HTAB 27: PRINT
     E2;: PRINT " ";FU$
7520  HTAB 26: PRINT "----------"
7530  HTAB 8: PRINT "SAVINGS";: HTAB 27: PRINT E1 -
     E2;: PRINT " ";FU$
7540  PRINT
7550  HTAB 24: PRINT "OPER. COSTS"
7560  HTAB 1: PRINT "ORIGINAL HOUSE";: HTAB 27: PRINT
     "$";M1
7570  HTAB 1: PRINT "IMPROVED HOUSE";: HTAB 27: PRINT
     "$";M2
7580  HTAB 26: PRINT "----------"
7590  HTAB 8: PRINT "SAVINGS";: HTAB 27: PRINT "$"
     ;MS
7600  HTAB 8: PRINT : PRINT "PAYBACK";: HTAB 27: PRINT
     PB;: PRINT " YEARS"
7610  PRINT : PRINT
7620  PRINT "ABOVE IS BASED ON ONE YEAR OF OPERATI
     ON"
7630  PRINT "IN ";DD$
7640  RETURN
8000  REM  DRAW HOUSE
8010  HOME : GR : COLOR= 07
8015  VLIN 14,28 AT 12
8020  VLIN 14,28 AT 24
8030  HLIN 12,23 AT 14
8040  HLIN 12,23 AT 28
8050  FOR X = 0 TO 8: PLOT 10 + X,15 - X: PLOT 18 +
     X,7 + X: NEXT X
8060  VLIN 24,28 AT 17: VLIN 24,28 AT 18: VLIN 24,
     28 AT 19
8070  COLOR= 02: VLIN 25,26 AT 18: COLOR= 12: VLIN
     16,17 AT 14: VLIN 16,17 AT 15
8080  VLIN 16,17 AT 21: VLIN 16,17 AT 22
8090  VLIN 23,24 AT 14: VLIN 23,24 AT 15
8100  VLIN 23,24 AT 21: VLIN 23,24 AT 22
8110  VTAB 22: HTAB 8: INVERSE : PRINT "HOME ENERG
     Y CALCULATOR": NORMAL
8115  HTAB 12: PRINT "PRESS ANY KEY"
8120  GET R$: IF R$ = "" THEN 8120
8130  TEXT : HOME : RETURN
```

# Utility Bill Audit

Larry L. Bihlmeyer

*With the high price of today's utilities, it's a good idea to check your bills for accuracy. Here's a practical program enabling you to verify your electric, gas, water, and phone bills.*

"Utility Bill Audit" is a versatile program that lets you check electric, gas, water, and phone bills. It also makes it easy to split the costs of those bills among the people living in your household. In addition, if you are interested in energy savings it will help you monitor electricity and gas consumption.

## Personalizing the Program

Before using Utility Bill Audit, you need to have a thorough understanding of how each bill is calculated in the program. First, a particular bill is split up according to the values (2, 1, 2, 4) given in the DATA statement in line 1640. They are assigned to the variable N(X) and represent the number of individuals who must pay for each bill. In its present form, the program assumes that the electric and water bills will be paid by two individuals, the phone bill by four, and the gas bill by one. However, it's unlikely that these numbers will correspond to the financial arrangements in your household. Be sure to substitute the appropriate values before you continue. Of course, if the bills are paid by one individual, simply replace the numbers in line 1640 with 1,1,1,1.

Since the program works on the actual cost of utilities, based on local rates, certain information about those rates must be provided. That information is READ in lines 380 and 400 from the DATA statements in lines 1650–1680.

Notice that the first three DATA statements in this sequence have nine entries and apply to the electric, gas, and water bills respectively. Consider line 1650 as an example. The first entry in that line is the name of the utility (ELECTRIC) for which the rates that follow apply. The second entry is the unit of measurement for that particular utility (KWH, for kilowatt hours). The next entry is the minimum service charge for the utility ($5.40 for electricity). The fourth entry is the tax rate based on the sum of the service charge and the rate

charge (0 percent for electric use). Those first four DATA entries are READ in as A$(1), B$(1), M(1), and Z(1), respectively.

The next two numbers are cutoff limits for each electric rate and are represented in the program by L1(1) and L2(1). The last three numbers are the actual rate charged per KWH use for each level of usage (R1(1), R2(1), and R3(1) in the program). Thus, the program is set up so that the rate charged for electricity is $.0495 for the first 350 KWH, $.0565 for the next 950 KWH (1300 minus 350), and $.0541 for any usage exceeding 1300 KWH.

The DATA statement in line 1680 is easier to follow. It includes the utility (PHONE), the minimum service charge ($13.50), and the tax rate on the service charge and long distance calls (3 percent).

So get out your most recent bills and read off the various rates (per KWH for electric and per CCF or hundred cubic feet for gas and water). If the rates are not given on a bill, contact the utility company to get a schedule of the latest rates. Then substitute your local rates for those in the DATA statements in lines 1650–1680.

## Program Operation

After inserting the correct rates, run the program. You will then be asked which utility bill you wish to check. The first three menu choices are electric, gas, and water. Bills for these three utilities are all calculated in the routine beginning at line 560.

Look at an electric bill as an example. When the routine at line 560 is executed, you will be required to INPUT the present and previous meter readings. Those values can be read directly from your latest electric bill. Next, you must INPUT the number of days in the billing period. Then you will be asked to INPUT any adjustments to the bill, either positive (for example, connection fees or previous balances) or negative (credits).

The program will calculate the amount of electricity consumed for the given period (defined as U in line 720). Then, depending on the value of U relative to the rate limits L1(1) and L2(X) (lines 740 and 750), an amount owed (T) before tax and adjustments will be calculated (lines 760, 780, and 800). Next, the tax on this amount will be determined (T1). Finally, a total electric bill—the sum of the minimum charge, usage

cost, tax, and adjustments—will be calculated (T3) in line 830.

The results are then PRINTed on the screen with provisions for formatting the output to two places past the decimal. Any numbers beyond the second decimal place are simply dropped. If you prefer rounded numbers, you could easily modify the program to give them.

The routine beginning at line 560, as mentioned, also calculates the gas and water bills. They are based on the rates READ from the DATA statements in lines 1660 and 1670. Notice the sets of large numbers (precisely, 99999) in line 1660. The rates for gas where I live are the same, regardless of the amount used. By using large numbers here for the cutoff limits, L1(2) and L2(2), for this utility, it's unlikely that the actual usage will exceed these amounts (see lines 740 and 750). Thus, the charge for this commodity will always be based on the first rate, or R1.

The rates for water, as READ from the DATA statement in line 1670, are based on a single cutoff limit (L1(3)) of 1000 CCF. For less than this, a usage rate (R1(3)) of $.144 per CCF is charged. If water usage exceeds 1000 CCF, a second rate (R2(3)) of $.160 is charged. Again, using a very large number (99999) for the second cutoff limit (L2(3)) assures that the overall usage cost is based only on two rates.

## Analyzing the Phone Bill

Phone bills are checked in a separate routine beginning at line 1110. Adjustments to the bill are initially INPUT in the same manner as they are with the electric, gas, and water bills. Next, the person responsible for each long distance charge is asked to INPUT the amount of each long distance call. A separate routine (lines 1260–1360) allows the individual to correct any typing mistakes. Finally, the amount owed by each individual is displayed.

The portion of the phone bill that each person must pay is the sum of the appropriate long distance tolls, a proportional amount of both the service charge and the billing adjustments, and a proportional amount of the tax levied on the service and long distance calls. Again, if only one person in the household foots the bills, the last number in line 1640 should be 1.

In addition to enabling you to catch billing errors and helping you to easily divide up household bills, this program

can help you monitor costs. If you add an energy-saving device that is supposed to save 10 percent of your total electric bill, take a meter reading when it is installed and verify the savings with a later reading. You can also project weekly, monthly, and yearly savings for any utility in this manner.

## Utility Bill Audit

```
90  DIM A$(4),B$(4),L1(3),L2(3),M(4),R1(3),R2(3),R
    3(3),W(50),Z(4)
100  GOTO 340
110 A1 = 0
120  PRINT "INPUT ADJUSTMENTS TO BILL(+ OR - ,'0'
    WHEN DONE)"
130  INPUT E
140 A1 = A1 + E
150  IF E = 0 THEN 170
160  GOTO 130
170  GOSUB 250
180  GOSUB 290
190  RETURN
200  PRINT "": REM   HOME
210  RETURN
220  PRINT "  ";A$(X);" BILL(CONT')"
230  PRINT
240  RETURN
250  FOR I = 1 TO 3
260  PRINT
270  NEXT I
280  RETURN
290  PRINT "INPUT C TO CONTINUE";
300  INPUT C$
310  GOSUB 200
320  RETURN
340  FOR I = 1 TO 4
350  READ N(I)
360  NEXT I
370  FOR I = 1 TO 3
380  READ A$(I),B$(I),M(I),Z(I),L1(I),L2(I),R1(I),
    R2(I),R3(I)
390  NEXT I
400  READ A$(4),M(4),Z(4)
410  GOSUB 200
420  PRINT "  UTILITY BILL AUDIT"
430  GOSUB 250
440  PRINT " 1. ELECTRIC BILL"
450  PRINT " 2. GAS BILL"
460  PRINT " 3. WATER BILL"
470  PRINT " 4. PHONE BILL"
```

```
480  PRINT " 5. ALL OF THE ABOVE"
490  PRINT " 6. EXIT"
500  PRINT
510  PRINT
520  PRINT " CHOOSE AN OPTION ";
530  INPUT P
540  IF (P < 1) + (P > 6) THEN 530
550  ON P GOTO 1020,1050,1080,1110,1590,1690
560  GOSUB 200
570  PRINT "     ";A$(X);" BILL"
580  PRINT
590  PRINT "PREVIOUS METER READING"
600  INPUT E1
610  PRINT
620  PRINT "PRESENT METER READING"
630  INPUT E2
640  PRINT
650  PRINT "INPUT DAYS IN THE BILLING PERIOD"
660  INPUT D
670  GOSUB 250
680  GOSUB 290
690  GOSUB 200
700  GOSUB 220
710  GOSUB 110
720 U = E2 - E1
730 Y = U / D
740  IF U > L2(X) THEN 800
750  IF U > L1(X) THEN 780
760 T = M(X) + R1(X) * U
770  GOTO 810
780 T = M(X) + R1(X) * L1(X) + R2(X) * (U - L1(X))
790  GOTO 810
800 T = M(X) + R1(X) * L1(X) + R2(X) * (L2(X) - L1
    (X)) + R3(X) * (U - L2(X))
810 T1 = T * Z(X)
820 T2 = T + T1
830 T3 = T2 + A1
840  GOSUB 200
850  GOSUB 220
860  PRINT "USE FOR THE PERIOD IS "; INT (U * 100)
      / 100;" ";B$(X)
870  PRINT
880  PRINT "USE/DAY IS "; INT (Y * 100) / 100;" ";
    B$(X);" OR $";
890  PRINT  INT (T2 / D * 100) / 100;"/DAY INCLUDI
    NG TAX"
900  PRINT
910  PRINT A$(X);" BILL:"
```

```
920  PRINT " W/OUT TAX   :$"; INT (T * 100) / 100
930  PRINT " TAX IS      :$"; INT (T1 * 100) / 100
940  PRINT " ADJ'TS      :$";A1
950  PRINT
960  PRINT "  *TOTAL*    :$"; INT (T3 * 100) / 100
970  IF N(X) = 1 THEN 990
980  PRINT "SPLIT ";N(X);" WAYS:$"; INT (T3 / N(X)
     * 100) / 100
990  PRINT
1000  GOSUB 290
1010  RETURN
1020 X = 1
1030  GOSUB 560
1040  GOTO 430
1050 X = 2
1060  GOSUB 560
1070  GOTO 430
1080 X = 3
1090  GOSUB 560
1100  GOTO 430
1110  GOSUB 200
1120 X = 4
1130  PRINT "      ";A$(X);" BILL"
1140  PRINT
1150  GOSUB 110
1160  FOR K = 1 TO N(X)
1170 I = 1
1180  IF N(X) = 1 THEN 1200
1190  PRINT "FOR PERSON #";K;",";
1200  PRINT "INPUT CHARGE FOR EACH LONG DISTANCE C
     ALL (INPUT '0' WHEN DONE)"
1210  INPUT W(I)
1220  IF W(I) = 0 THEN 1250
1230 I = I + 1
1240  GOTO 1210
1250  GOSUB 200
1260  PRINT " PERSON #";K;",";
1270  FOR J = 1 TO I - 1
1280  PRINT "CALL #";J;"  :$";W(J)
1290  PRINT
1300  PRINT "IS THIS CORRECT (Y/N)"
1310  INPUT C$
1320  IF C$ = "Y" THEN 1350
1330  PRINT "TYPE IN CORRECTION"
1340  INPUT W(J)
1350  PRINT
1360  NEXT J
1370  GOSUB 200
1380  GOSUB 220
```

```
1390 T = 0
1400  FOR J = 1 TO I - 1
1410 T = T + W(J)
1420  NEXT J
1430  PRINT "SERVICE:    $"; INT (M(X) / N(X) * 100
     ) / 100
1440  PRINT
1450  PRINT "LD CALLS:   $";T
1460  PRINT
1470  PRINT "ADJ'TS :    $"; INT (A1 / N(X) * 100) /
     100
1480  PRINT
1490 T1 = T +  INT (M(X) / N(X) * 100) / 100
1500 T2 =  INT (T1 * Z(X) * 100) / 100
1510  PRINT "TOTAL TAX: $";T2
1520  PRINT
1530  PRINT
1540  PRINT " TOTAL BILL: $";T1 + T2 +  INT (A1 /
     N(X) * 100) / 100
1550  GOSUB 250
1560  GOSUB 290
1570  NEXT K
1580  GOTO 430
1590  FOR F = 1 TO 3
1600 X = F
1610  GOSUB 560
1620  NEXT F
1630  GOTO 1110
1640  DATA  2,1,2,4
1650  DATA  ELECTRIC,KWH,5.40,0,350,1300,.0495,.05
     65,.0541
1660  DATA  GAS,CCF,4.05,0,99999,99999,.49541,0,0
1670  DATA  WATER,CCF,3.26,0,500,99999,.144,.160,0

1680  DATA  PHONE,13.50,.03
1690  END
```

# Calorie Cop

Gerald P. Graham
Apple translation by Kevin Martin

*This program determines your calorie output for a variety of activities (ranging from sitting still to weight lifting) and gives you your total daily energy output. With onscreen instructions and an easy-to-understand menu, it's simple to use.*

In the December 1982 issue of *COMPUTE!* magazine, Charles Brannon presented a program for calculating the calories in your diet. It also estimated your daily needs and then predicted how long it would take you to lose any weight you wanted to lose.

"Calorie Cop" is a companion program that tells you the caloric output, per pound of body weight, for each activity you perform. It also determines the calories that you actually expend for each activity, allowing you to see your total daily energy output.

When you run the program you are given instructions, followed by a seven-page alphabetical menu of activities from archery to wrestling. Just press the letter corresponding to your activity; if you don't see it, press RETURN to turn the pages until you find it. If your chosen activity is not listed, then use one that is comparable.

Keep in mind that the results should be modified by knowledge of the context of the activities. In cases where an unskilled person is competing against a skilled person, for instance, the former usually works harder. Desire and effort are also factors. One research study involved filming very heavy individuals playing tennis doubles. The very heavy players were found to be standing still 65 percent of the time. While vigorous tennis doubles requires .046 calories per minute per pound of weight, standing is worth only .011 calories per minute per pound.

To exit the program type a 0 when you are prompted for CHOICE. Before ENDing, the program will tell you the total number of calories used.

*Do not use this or any other diet/exercise program except under the advice and consent of your physician.*

## Calorie Cop

```
1 WI = 40:LE = 24
5  HOME
10  PRINT : PRINT : PRINT : PRINT
15  PRINT  TAB( WI / 2 - 5);"CALORIE COP"
30  FOR I = 1 TO 1000: NEXT I
40  HOME
50  PRINT "THIS PROGRAM WILL TELL YOU HOW MANY"
55  PRINT : PRINT "CALORIES YOU USE FOR A PARTICUL
     AR"
60  PRINT : PRINT "ACTIVITY.  YOU WILL BE PROVIDED
      AN"
65  PRINT : PRINT "ACTIVITY MENU THAT WILL TELL YO
     U HOW"
70  PRINT : PRINT "MANY CALORIES EACH ACTIVITY USE
     S EACH"
75  PRINT : PRINT "MINUTE FOR EACH POUND OF YOUR B
     ODY"
80  PRINT : PRINT "WEIGHT.  IT WILL ALSO GIVE YOU
     A TOTAL"
85  PRINT : PRINT "OF ALL CALORIES USED."
90  PRINT : PRINT "PRESS ANY KEY WHEN FINISHED";
95  GET A$
250  IF CAL = 1 THEN 730
260  GOSUB 1020
270  FOR I = 1 TO LE - 4
280  READ ACTIVITY$,CL
290  IF ACTIVITY$ = "END" THEN 330
300  PRINT  CHR$ (64 + I);"-";ACTIVITY$
320  NEXT I
330 I = I - 1
340  PRINT : PRINT "CHOICE ";
350  GET A$
355  IF (A$ < "A" OR A$ >  CHR$ (I + 64)) AND A$ <
      > "0" AND A$ <  > CHR$ (13) THEN 350
360  IF A$ <  > CHR$ (13) THEN 410
370 NX = NX + 1: IF ACTIVITY$ = "END" THEN  RESTORE
     :NX = 0
400  GOTO 260
410  RESTORE
430  IF A$ = "0" THEN 660
440  FOR I = 1 TO NX * (LE - 4) +  ASC (A$) - 64
450  READ ACTIVITY$,CL
460  NEXT I
470  HOME
480  PRINT : PRINT "ACTIVITY: ";ACTIVITY$
490  PRINT : PRINT "THIS ACTIVITY USES:": PRINT CL
```

```
500  PRINT "CALORIES PER MINUTE PER"
510  PRINT "POUND"
520  PRINT : PRINT "ENTER LENGTH OF ABOVE
530  PRINT "ACTIVITY IN MINUTES";
540  INPUT MIN
545  IF MIN = 0 THEN 590
547  IF MIN < 0 THEN 540
550  PRINT : PRINT "ENTER YOUR BODY WEIGHT"
555  IF MIN < 0 THEN 540
560  PRINT "IN POUNDS";
570  INPUT LBS
572  IF LBS < 0 THEN 570
575 OUTPUT = LBS * MIN * CL
577  PRINT "CALORIES USED FOR THIS"
578  PRINT "ACTIVITY= ";OUTPUT
579  PRINT : PRINT "TOTAL CALORIES USED"
580  PRINT "SO FAR= ";:CAL = CAL + OUTPUT: PRINT C
     AL
582  PRINT "PRESS RETURN TO CONTINUE..."
585  GET A$: IF A$ <  > CHR$ (13) THEN 585
590  RESTORE :NX = 0: GOTO 260
660  HOME
665  PRINT : PRINT : PRINT "YOUR TOTAL CALORIES": PRINT
     "ARE ";CAL
670  PRINT : PRINT : PRINT : PRINT "THAT'S ALL FOL
     KS!"
680  FOR PAUSE = 1 TO 1000: NEXT PAUSE
690  END
1020  HOME
1030  PRINT "-=<ACTIVITY MENU>=-"
1035  RETURN
1036  REM  BE SURE TO INCLUDE THE DATA STATEMENTS
     IN PROGRAM 4
1040  DATA  ARCHERY,.034
1050  DATA  BADMINTON-MODERATE,.039
1060  DATA  BADMINTON-VIGOROUS,.065
1070  DATA  BASEBALL,.031
1080  DATA  BASEBALL-PITCH&CATCH,.040
1090  DATA  BASKETBALL-MODERATE,.047
1100  DATA  BASKETBALL-VIGOROUS,.066
1110  DATA  BED MAKING,.031
1120  DATA  BICYCLING-DOWNHILL,.018
1130  DATA  BICYCLING-SLOW-LEVEL,.030
1140  DATA  BICYCLING-MODERATE,.050
1150  DATA  BICYCLE-FAST-UPHILL,.072
1160  DATA  BOXING-IN RING,.101
1170  DATA  BOXING-SPARRING,.063
1180  DATA  BOWLING,.028
1190  DATA  CANOEING,.029
```

```
1200  DATA  CONVERSING,.011
1210  DATA  COOKING,.013
1220  DATA  DANCING-SLOW,.029
1230  DATA  DANCING-MODERATE,.045
1240  DATA  DANCING-FAST,.064
1250  DATA  DRESSING&UNDRESSING,.030
1260  DATA  DRIVING A CAR,.019
1270  DATA  DUSTING,.010
1280  DATA  EATING,.011
1290  DATA  EXERCISES-ABDOMINAL,.020
1300  DATA  EXERCISES-BALANCING,.016
1310  DATA  EXERCISES-JUMPING,.043
1320  DATA  EXERCISES-BENDING,.023
1330  DATA  FENCING-MODERATE,.033
1340  DATA  FENCING-VIGOROUS,.057
1350  DATA  FIELD HOCKEY,.063
1360  DATA  FIELD HOCKEY-GOALIE,.030
1370  DATA  FISHING,.016
1380  DATA  FOOTBALL-BACKS&ENDS..050
1390  DATA  FOOTBALL-LINEMEN,.040
1400  DATA  GARDENING,.030
1410  DATA  GOLF-CROWDED&WALKING,.030
1420  DATA  GOLF-UNCROWDED&WALK,.035
1430  DATA  GYMNASTICS-LIGHT,.030
1440  DATA  GYMNASTICS-HEAVY,.056
1450  DATA  HANDBALL,.063
1460  DATA  HIKING,.042
1470  DATA  HILL&STAIR CLIMBING,.060
1480  DATA  HORSEBACK RIDE-WALK,.019
1490  DATA  HORSEBACK RIDE-TROT,.046
1500  DATA  HORSEBACK RIDE-GALLOP,.067
1510  DATA  IRONING,.018
1520  DATA  JUDO,.087
1530  DATA  JUMPING ROPE,.087
1540  DATA  KARATE,.087
1550  DATA  LACROSSE,.063
1560  DATA  LACROSSE-GOALIE,.030
1570  DATA  LISTENING TO RADIO,.010
1580  DATA  MOTOR BOATING,.016
1590  DATA  MOUNTAIN CLIMBING,.086
1600  DATA  PAINTING-INSIDE,.015
1610  DATA  PAINTING-OUTSIDE,.035
1620  DATA  PLAYING CARDS,.011
1630  DATA  PLAYING DRUMS,.030
1640  DATA  PLAYING HORN,.013
1650  DATA  PLAYING PIANO,.018
1660  DATA  RACQUETBALL,.063
1670  DATA  RESTING-LYING DOWN,.008
1680  DATA  RESTING-SITTING,.009
```

```
1690  DATA  ROWING-SLOW,.036
1700  DATA  ROWING-VIGOROUS,.118
1710  DATA  RUNNING-11 MIN./MILE,.071
1720  DATA  RUNNING-10 MIN./MILE,.078
1730  DATA  RUNNING-9 MIN./MILE,.085
1740  DATA  RUNNING-8 MIN./MILE,.092
1750  DATA  RUNNING-7 MIN./MILE,.100
1760  DATA  RUNNING-6 MIN./MILE,.110
1770  DATA  RUNNING-5 MIN./MILE,.130
1780  DATA  SAILING,.020
1790  DATA  SCRUBBING,.032
1800  DATA  SEWING OR KNITTING,.010
1810  DATA  SHOPPING,.028
1820  DATA  SHOWERING,.034
1830  DATA  SINGING-STANDING,.017
1840  DATA  SITTING-QUIETLY,.010
1850  DATA  SITTING-WRITING,.013
1860  DATA  SKATING-MODERATE,.036
1870  DATA  SKATING-VIGOROUS,.064
1880  DATA  SKIING-DOWNHILL,.059
1890  DATA  SKIING-LEVEL-SLOW,.054
1900  DATA  SKIING-LEVEL-FAST,.078
1910  DATA  SLEEPING,.007
1920  DATA  SOCCER,.063
1930  DATA  SOCCER-GOALIE,.030
1940  DATA  SQUASH,.070
1950  DATA  STANDING,.011
1960  DATA  STATIONARY RUNNING,.078
1970  DATA  STUDYING,.014
1980  DATA  SWIM-CRAWL-30YDS/MIN,.058
1990  DATA  SWIM-CRAWL-40YDS/MIN,.071
2000  DATA  SWIM-BKSTRK-30YDS/MN,.035
2010  DATA  SWIM-BKSTRK-40YDS/MN,.055
2020  DATA  SWIM-BREAST-30YDS/MN,.048
2030  DATA  SWIM-BREAST-40YDS/MN,.064
2040  DATA  SWIM-BUTTERFLY,.078
2050  DATA  TABLE TENNIS-MOD.,.046
2060  DATA  TABLE TENNIS-VIG.,.065
2070  DATA  TELEPHONING,.011
2080  DATA  TENNIS-SNGLS-MOD.,.046
2090  DATA  TENNIS-SNGLS-VIG.,.065
2100  DATA  TENNIS-DBLES-MOD.,.038
2110  DATA  TENNIS-DBLES-VIG.,.046
2120  DATA  TYPING,.015
```

```
2130  DATA  VOLLEYBALL-BEG.-MOD.,.020
2140  DATA  VOLLEYBALL-BEG.-VIG.,.036
2150  DATA  VOLLEYBALL-SKILL-MOD.,.040
2160  DATA  VOLLEYBALL-SKILL-VIG.,.065
2170  DATA  WALKING-2 MPH,.022
2180  DATA  WALKING-3 MPH,.030
2190  DATA  WALKING-4 MPH,.039
2200  DATA  WALKING-5 MPH,.064
2210  DATA  WASHING DISHES,.015
2220  DATA  WASHING HANDS & FACE,.020
2230  DATA  WATCHING TV,.010
2240  DATA  WATER SKIING,.053
2250  DATA  WEIGHT LIFTING-ARMS,.050
2260  DATA  WEIGHT LIFTING-LEGS,.060
2270  DATA  WEIGHT LIFTING-BODY,.065
2280  DATA  WRESTLING,.091
2290  DATA  END,0,0
```

# Chapter 4

# Programming

# Introduction

Many Apple users enjoy programming, and the utilities and programming techniques described in this chapter will make the experience even more satisfying.

For instance, John Sarver's "Apple Fast Sort" can be incorporated into your programs to make short work out of sorting long lists.

You can dress up your programs with the custom directory headers described in G.J. Vulling's "Custom Headers." Similar techniques are described by Dan Jordan in "Apple Input and Menu Screens."

"Using Commas, Colons, and Quote Marks in Apple Input Statements," by Craig Peterson, will tell you how to solve that particularly knotty programming problem.

Finally, Michael P. Antonovich's "Undeletable Lines" shows you an innovative way to personalize programs—by creating lines that cannot be deleted by ordinary means.

# Apple Editing Hints

Patrick Moyer

*Most computer owners develop a love-hate relationship with at least one feature of their machines. For the Apple this feature is often the editing functions. Here is a review of Apple editing controls and protocols and some tips on making the process easier and more effective.*

The Apple uses a combination of screen editing and line editing. Changes are made by moving the cursor to a particular line which has been listed on the screen and retyping that line. This retyping is usually accomplished with the right arrow key. As the right arrow is pressed, the cursor moves to the right, reentering all it passes over. A change is made by typing over what is already there, or by inserting the correction through a combination of cursor moves.

## Physical, Logical

Therefore, to make a change, we must specify the line to be changed. In this case, we are talking about a line of BASIC, not a line displayed on the screen. The BASIC line is called a *logical line,* as opposed to the *physical line* that is displayed on the screen. A logical line may contain multiple BASIC statements and may be up to 255 characters long. The physical display line is the 40-letter width of the screen.

Before a BASIC line can be changed, it must be listed. It is best to clear the screen with the HOME command initially. This eliminates confusion about what was changed and what wasn't.

When a line is listed, the computer puts one space between words or variables, two spaces after the line number, seven spaces at the end of the first physical line, and five spaces on the right and left sides of the remaining physical lines.

Most of the time, these extra spaces and lines are of little consequence. One can just merrily right-arrow over them with no harm. The one exception occurs in string information (characters in quotes). This causes a problem. If a string is broken between two or more physical lines during the listing process, and you right-arrow to retype, 12 additional spaces will be inserted between the last character on the first line and the

first character on the next line. Certainly not what's wanted. The common solution is to avoid the right arrow and use the cursor with the <ESC>K sequence instead.

## Simplified Cursor Control

There's an even simpler solution. Let's edit a line step by step to demonstrate this technique

(<ESC> is the ESC KEY, <RET> is the RETURN KEY):
Here's the line as originally typed:

```
10PRINT"THIS IS A LONG LINE OF STRING
DATA"<RET>
```

List the line. It looks like this:

```
LIST10<RET>
10 PRINT "THIS IS A LONG LINE OF STR
   ING DATA"
```

We then type <ESC>I, repeating the I key until the cursor is over the second digit of the line number; J is pressed to move the cursor one space to the left. (This J keypress is important. If you forget it and continue the editing process, you will gain a line in your program. Line 0 will be created, but more about that later.)

Once you've moved left, leave <ESC> mode. This is done by pressing any key not having meaning in <ESC> mode. Because some keys not normally used for cursor movement do have special meaning, it's best to press the space bar. Remember, this will not move the cursor.

We can now use the right arrow to "retype" the line to the place of the change. The repeat key can be used to speed this process. Let's say you've used the right arrow until it appears after the last quote. The line on the screen looks no different. However, if we LIST the line, we now see this:

```
10 PRINT "THIS IS A LONG LINE OF STR
   ING DATA"
```

If we type RUN we get:

```
RUN<RET>
THIS IS A LONG LINE OF STR     ING DAT
A
```

## Eliminating Problem Margins

The common solution, again, is to right-arrow to the R in STR,

then type <ESC> and press K repeatedly to move the cursor until you reach the I in ING. Anyone who has done this often will know how easy it is to forget <ESC> K, and end up with a string of K's.

The solution is simply to eliminate those extra margins unless you need them. Let's start with the same original line:

```
10PRINT"THIS IS A LONG LINE OF STRING
DATA"<RET>
```

To edit the line we type:

```
HOME:POKE33,30:LIST10<RET>
```

The HOME gives us a clean screen to work with; the LIST puts the line to be edited on the screen. A POKE instruction places a single number into an address in the computer's memory. Address 33 controls the width of the screen display. Placing the number 30 in it reduces the size of the screen to 30 characters wide rather than 40.

*Caution*: The POKE must be done before the LIST for this method to work. The HOME is optional, but prevents a very confusing screen. (Try it. You'll see what I mean.) The screen will erase and display:

```
10  PRINT"THIS IS A LONG LINE OF S
TRING DATA"
```

As you can see, the line is 30 characters wide without the extra margin spaces. Move the cursor to the line number as usual. The right arrow may be used without ill effect. It will go directly from the S on the first display line to the T on the second line without inserting any blanks. This eliminates the need to use the <ESC> K sequence.

Once you have finished editing, you will need to type TEXT. This command will return you to normal 40-character screen mode.

## Duplicating Lines

One strength of Apple editing is the ability to duplicate lines. Let's try an example:

```
HOME: POKE33,30:LIST10<RET>
10    PRINT"THIS IS A LINE TO BE
DUPLICATED"
```

Next move the cursor up to the line using the normal <ESC>I. When the cursor arrives over the number, move it left until it is over the first digit of the number. Then press the space bar as before; but prior to using the right arrow, retype the line number, say, 20. Then use the right arrow to "retype" the line as described above until you reach the end of the logical line. At this point, press RETURN. If you LIST the program, you'll see:

```
HOME:POKE33,30:LIST<RET>
10    PRINT"THIS IS A LINE TO BE
DUPLICATED"
20    PRINT"THIS IS A LINE TO BE
DUPLICATED"
```

Once you have moved your cursor up to the number and changed it, you do not have to reuse the entire line. You can treat it like any line to be edited further if necessary.

## Easy Program Merge

This technique can also be used on a limited scale to merge two programs. Let's say you have a favorite subroutine of three or four lines which you wish to add to a program. You could use the merge function of the "Renumber" program on the *System Master*, or the program that is part of the *Programmer's Toolkit*. If you don't have these programs or you don't have them handy, here is a simple procedure:

1. Save the program you are working on.
2. Load the program which contains the lines to be copied to your new program.
3. Clear the screen, change width, and list lines (using HOME:POKE33,30:LIST statements).
4. Now, load the program the lines are to be added to.
5. Using the normal <ESC> and right-arrow commands, edit each line without changes. It's best to edit the last line first and work up the screen, entering each line one at a time. This is because when multiple lines are listed and edited, once <RET> is pressed, the line number below it is partially destroyed and has to be retyped by hand. There's

nothing wrong with changing the line numbers to fit your new program if the current line numbers are a problem.

6. Once all lines are edited, save the program. If you list it, you'll find the lines are now part of your program.

Finally, if you want to cancel a particular change, as long as you have not pressed <RET> yet, cancel the editing of the line by typing <CTRL> X. Be sure that you press the <CTRL> key first, then X. The machine will answer with a backward slash. If you list the line, it will be unchanged.

# Apple Fast Sort

John Sarver

*Using this program, you will be able to alphabetize a list in near-record time.*

Until now, it may have taken you a long time to alphabetize a list of names or programs. In fact, in a recent test using a BASIC bubble sort routine, it took my Apple eight hours and 57 minutes to sort 1000 randomly created strings. But with this subroutine you'll be able to put both one- and two-dimensional Apple arrays in alphabetical order in only one minute and 45 seconds.

String values, when assigned, are stored at the very top of Apple's free RAM. As more strings are assigned, they are stored below the strings already in memory. A table, created when you use the DIM statement, keeps track of where each string is in RAM.

Some important information is stored at the beginning of this table. The first byte represents the first character in the variable name. The second byte represents the second character in the variable name plus $80 (adding $80 designates it as a string array rather than an integer or decimal point number array). The next pair of bytes gives the length of this pointer table.

The fifth byte is the number of dimensions that you have used with the DIM statement. If you used a two-dimensional array, the next two bytes tell how many variables are in the second part of the dimension. If it's a three-dimensional array, it uses the next four bytes, and so on.

The final two bytes indicate the number of strings in the first dimension. The table begins there, and each variable is located by a three-byte pointer. The first byte is the length of the record, and the next two point to where the first character of the variable is stored. Those pointers are always in order from the zero dimension to the *n*th dimension.

At the end of this grouping of pointers are the pointers for the first group of the second dimensioned part of the array. Following this is the second group of pointers for the second dimensioned part of the array, and so on. If you used a one-dimensional array, there would be only one group of pointers.

As you can see, there is no need to sort the strings themselves. It's much quicker just to sort the pointers. Pointer sorting wastes no time in garbage collection—and, in most cases, the length of the strings does not affect the time of execution.

## Simple to Use

Using this sort is quite simple. Apple stores the last variable used in $81 and $82, so you may need to insert a statement in your BASIC program such as A$(0)=A$(0) (see line 90 of Program 2). You can also POKE these values in if you are putting this utility on another machine.

The sort can be easily changed to use the zero dimension of an array if you wish. To do this, simply change the following lines in the BASIC loader (Program 1):

```
120 IF CK <> 56854 THEN PRINT "CHECK DATA
    STATEMENTS FOR ERROR":STOP
200 DATA 169,0,133,253,133,239,169,1
400 DATA 165,6,105,2,133,6,169,0
```

If you are using a two-dimensional array, you will need to store the records that are to be put in order by using the zero subscript of the second dimension (that is, A$(1,0), A$(2,0), etc.). The accompanying arrays (A$(1,1), A$(2,1), A$(1,2), A$(2,2), etc.) will be kept with their respective zero-subscripted record.

The sort will automatically ascertain if you are using a one- or two-dimensional array and will adjust itself accordingly. You may use any number of subscripts desired in one-dimensional arrays and in the first part of the two-dimensional array. Some of the corresponding subarrays would not be properly aligned.

Program 1, the ML fast sort loader, loads the machine language sorting routine into RAM. You should save this on disk by typing:

```
BSAVE SORT, A$944A, L$1B6
```

Program 2 illustrates how you might use the routine.

## Program 1. ML Fast Sort Loader

```
100  REM  THIS PROGRAM INSTALLS BUT DOES NOT RUN T
     HE ML FAST SORT
110  FOR I = 37962 TO 38399: READ A:CK = CK + A: POKE
     I,A: NEXT
```

```
120  IF CK < > 56857 THEN  PRINT "CHECK DATA STAT
     EMENTS FOR ERROR": STOP
130  TEXT : HOME : PRINT "TYPE 'BSAVE SORT,A$944A,
     L$1B6'"
140  PRINT "TO SAVE SORT ROUTINE ON DISK"
150  NEW
200  DATA  169,0,133,253,169,1,133,239
210  DATA  133,31,166,107,134,6,166,108
220  DATA  134,7,165,129,160,0,209,6
230  DATA  208,3,32,126,148,200,208,246
240  DATA  232,134,7,228,112,208,239,209
250  DATA  6,208,3,32,126,148,200,196
260  DATA  111,208,244,96,165,130,200,208
270  DATA  2,230,7,209,6,240,10,192
280  DATA  0,208,2,198,7,136,165,129
290  DATA  96,192,0,208,2,198,7,136
300  DATA  24,152,101,7,133,7,169,0
310  DATA  101,7,133,7,104,104,56,160
320  DATA  4,177,6,233,1,240,8,200
330  DATA  200,177,6,133,31,169,2,24
340  DATA  101,6,105,5,133,6,169,0
350  DATA  101,7,133,7,160,0,177,6
360  DATA  133,249,133,251,133,26,200,177
370  DATA  6,133,250,133,25,162,2,24
380  DATA  165,250,101,25,133,25,165,251
390  DATA  101,26,133,26,202,208,240,24
400  DATA  165,6,105,5,133,6,169,0
410  DATA  101,7,133,7,56,165,250,229
420  DATA  239,133,250,133,252,176,10,165
430  DATA  239,240,6,198,249,165,249,133
440  DATA  251,165,6,133,237,165,7,133
450  DATA  238,169,0,198,250,197,250,208
460  DATA  42,197,249,240,5,198,249,24
470  DATA  144,33,197,253,240,18,133,253
480  DATA  198,252,165,251,133,249,165,252
490  DATA  133,250,208,213,165,251,208,1
500  DATA  96,56,233,1,133,249,133,251
510  DATA  24,144,198,24,165,237,133,235
520  DATA  105,3,133,237,165,238,133,236
530  DATA  105,0,133,238,160,0,132,254
540  DATA  177,235,208,6,177,237,240,177
550  DATA  208,54,209,237,240,8,144,6
560  DATA  177,237,240,165,133,254,133,255
570  DATA  162,0,200,177,235,149,0,177
580  DATA  237,149,2,232,192,2,208,242
590  DATA  160,0,177,0,209,2,240,4
600  DATA  144,135,176,12,200,196,255,208
610  DATA  241,165,254,208,3,76,19,149
620  DATA  169,1,133,253,160,0,177,235
630  DATA  72,177,237,145,235,104,145,237
```

```
640  DATA  200,192,3,208,241,166,31,202
650  DATA  240,45,24,165,235,101,25,133
660  DATA  27,165,236,101,26,133,28,165
670  DATA  237,101,25,133,29,165,238,101
680  DATA  26,133,30,160,0,177,27,72
690  DATA  177,29,145,27,104,145,29,200
700  DATA  192,3,208,241,202,208,3,76
710  DATA  19,149,24,165,27,101,25,133
720  DATA  27,165,28,101,26,133,28,165
730  DATA  29,101,25,133,29,165,30,101
740  DATA  26,133,30,24,144,205,141,183
```

## Program 2. Using Fast Sort

```
10  HIMEM: 37962
20 D$ =  CHR$ (4)
30  PRINT D$"BLOAD SORT"
40  INPUT "HOW MANY RECORDS";N
45  DIM A$(N)
50  FOR A = 1 TO N
60  PRINT "WHAT IS RECORD #"A;
70  INPUT " ";A$(A)
80  NEXT
90 A$(0) = A$(0)
100  CALL 37962
110  FOR A = 1 TO N
120  PRINT A$(A)
130  NEXT
140  END
```

# Custom Headers

G. J. Vullings

*This program lets you create customized directory headers, with inverse or normal input. For Apples with DOS 3.2.1 or 3.3.*

Have you ever wished to identify the theme of a series of programs on a disk or improve the appearance of the directory as it appears on the screen? "Custom Catalog" may be what you've been looking for. It will create seven bogus files at the top of the directory, and those files will provide a header for the disk's directory by displaying contents, ownership, DOS version, or whatever you wish.

The program is designed to run with DOS 3.2.1 or 3.3. It will permit either inverse or normal input and will allow toggling between the two input states. These features give you some element of artistic control over your disk directories.

## Choose Your Input Types

The program should be used only with newly initialized disks, since it will occupy the first seven entries in the directory. Thus, if the program is used with established disks, the first seven programs will become inaccessible. To implement Custom Catalog, initialize a disk the normal way and then delete the HELLO program. Run Custom Catalog and, when prompted, insert the disk to be customized.

You have an initial choice of input states (normal or inverse) and can then design seven lines of 23 characters each (using all but control characters) to represent your identifying remarks or messages. The program sets aside a buffer of 256 bytes, using the input/output block at decimal location 896. There it stores the last sector of the directory track (track 17, sector 12 or track 17, sector 15, depending on the DOS version being used).

Each directory entry occupies 35 bytes. The first two represent the track and sector of the track/sector list (header). They are directed to an empty sector, generally track 17, sector 1. The third byte represents the file type. Here we will use 00 to indicate an unlocked text file.

The next 30 bytes represent the filename. We will make the first seven bytes backspaces to eliminate the "t" (for text)

and the sector count from the display. The remaining 23 bytes can be anything you wish, in normal or inverse. Byte 34 is the file length and is set to 00. The last byte is the end marker, which is also normally 00.

You have now altered the entries in the buffer, but one problem remains. The output for the directory listing is via the COUT routine at $FDED using screen ASCII values, but keyboard ASCII values which you entered are in a different range. Translate these values listing logical variables (one of the least used, yet most powerful, variable types). See page 15 of the *Apple Reference Manual* to determine screen ASCII values for both normal and inverse display. After altering the buffer, the revised version is written back to the disk.

Using a similar technique, track 2, sector 2 is then read into the buffer. The DISK VOLUME message which occupies bytes 176–186, inclusive, may be optionally changed. A typical directory header is shown in Figure 1; it looks especially attractive in inverse mode. Additionally, if the seventh line is left blank, a natural break is formed to separate the header from the rest of the directory.

## Figure 1. A Typical Custom Directory Header

```
&&&&&&&&&&&&&&&&&&&&&&&
& APPLE II - DOS 3.3  &
&    DISK UTILITIES   &
&     JANUARY 1983    &
&PROP. OF G.J.VULLINGS&
&&&&&&&&&&&&&&&&&&&&&&&
```

Both backspace and forwardspace editing may be used as usual, although you cannot backspace beyond the first column (column numbers are provided to make centering easier). Therefore, pressing RETURN or typing past column 23 is final.

After the seventh line is entered, you are given the choice of accepting or rejecting the header that you have constructed. If you reject it, the procedure will begin again. Rejecting headers helps you learn how to create headers that are aesthetically pleasing. If you accept the header, it will be permanently written to the disk. The choice will then be of-

fered to change the DISK VOLUME message to any 11 (or fewer) characters of your choice.

There are many ways in which you can use this program. You can create additional custom features, for example, or you might want to create flashing entries, which you can get by translating to the required ASCII values.

## How It Works

| Line(s) | |
|---|---|
| **30–220** | The input routine, which allows input in two modes as well as forwardspace and backspace editing. |
| **250–260** | Translate keyboard ASCII into screen ASCII and store into disk buffer. |
| **280–290** | Toggle input status. |
| **310–330** | Backspace edit routine. |
| **350–390** | Forwardspace edit routine. Translate screen ASCII to keyboard ASCII. |
| **410–450** | Point each of the bogus header files to empty track 17, sector 1; declare each file to be of type "text-unlocked" of length zero; and set the end marker. |
| **470** | Inputs a series of seven backspaces into the filenames so that the lock indicator, file type, and sector count do not appear on screen. |
| **480** | Checks the memory size of your Apple and sets up a disk buffer, making the program virtually memory-size independent. |
| **500–570** | Organize screen display. |
| **590–620** | Set HIMEM: to protect the buffer and initialize the variables. |
| **640–670** | Use track 17, sector 0, to find the directory, thus making it possible to use the program with either DOS 3.2.1 or 3.3, or even with disks having directories on tracks other than track 17. |
| **680–800** | Main routine. |
| **820–840** | Write the catalog header to the disk. |
| **860–920** | Change DISK VOLUME message. |
| **940–990** | Finishing touches. |
| **1020–1040** | Set up the input/output block for the READ/WRITE track sector routine. |

## Custom Catalog

```
5  TEXT : HOME : ONERR  GOTO 1000
10  GOTO 480
20  REM

***.INPUT ROUTINE.***

30  FOR I = 0 TO 6
40  VTAB VTB + I: HTAB HTB
50 CN = 1
60  INVERSE
70  IF  NOT INV THEN  NORMAL
80  GET CH$: IF CH$ <  > CHR$ (13) THEN 110
90  IF CN > 23 THEN 200
100  FOR Z = CN TO 23:CH$ = " ": PRINT CH$;: GOSUB
     250:CN = CN + 1: NEXT : GOTO 200
110  IF CH$ =  CHR$ (27) THEN  GOSUB 270: GOTO 60
120  IF CH$ =  CHR$ (8) THEN  GOSUB 300: GOTO 60
130  IF CN > 23 THEN 200
140  IF CH$ =  CHR$ (21) THEN  GOSUB 340: GOTO 160

150  IF  ASC (CH$) < 32 THEN 60
160  PRINT CH$;
170  GOSUB 250
180 CN = CN + 1
190  GOTO 60
200  GOSUB 460
210  GOSUB 400
220  NEXT
230  RETURN
240  REM

***.SCRN ASC INTO BUFFER.***

250  IF  ASC (CH$) >  = 32 AND  ASC (CH$) < 64 THEN
      POKE BFR + I * 35 + 10 + CN, ASC (CH$) + ( NOT
     INV > 0) * 128: RETURN
260  POKE BFR + I * 35 + 10 + CN, ASC (CH$) - (INV
      > 0) * 64 + ( NOT INV > 0) * 128: RETURN
270  REM

***.CHANGE INPUT STATE.***

280  IF INV THEN INV = 0: RETURN
290 INV = 1: RETURN
300  REM

***.BACKSPACE ROUTINE.***
```

```
310 CN = CN - 1: IF CN = 0 THEN  POP : GOTO 50
320  PRINT CH$:
330  RETURN
340  REM

***.FORWARDSPACE ROUTINE.***

350 ASKII =  PEEK ( PEEK (40) + 256 *  PEEK (41) +
     PEEK (36))
360  IF ASKII < 32 THEN CH$ =  CHR$ (ASKII + 64): RETURN
370  IF ASKII < 64 THEN CH$ =  CHR$ (ASKII): RETURN

390 CH$ =  CHR$ (ASKII - 128): RETURN
400  REM

***.PLACE COMMON POINTERS.***

410  POKE BFR + I * 35 + 1,TRK
420  POKE BFR + I * 35 + 2,1
430  POKE BFR + I * 35 + 3,0
440  POKE BFR + I * 35 + 34,0
450  POKE BFR + I * 35 + 35,0: RETURN
460  REM

***.PUT BKSPACES IN DIRECTORY.***

470  FOR M = 4 TO 10: POKE BFR + I * 35 + M,136: NEXT
     : RETURN
475  REM

***.SET DISK BUFFER.***

480 BL =  PEEK (115):BH =  PEEK (116) - 1:BUFR = B
    L + BH * 256
490  REM

***.INITIALIZE SCREEN.***

500  TEXT : HOME : VTAB 2: INVERSE : FOR I = 1 TO
     40: PRINT "=";: NEXT
510  PRINT "=       APPLE II CATALOG CUSTOMIZER
      =";
520  PRINT "=";
530  FOR I = 1 TO 38: PRINT " ";: NEXT
540  PRINT "=";
550  PRINT "=              ";: NORMAL : PRINT "BY G.
     J. VULLINGS";: INVERSE : PRINT "           =";

560  FOR I = 1 TO 40: PRINT "=";: NEXT : NORMAL
```

```
570  POKE 34,6
580  REM

***.INITIALIZE VARIABLES.***

590  HIMEM: BUFR:IOB = 904:ITRK = IOB + 4:ISECT =
     IOB + 5:IBUFP = IOB + 8:ICMD = IOB + 12:ST =
     IOB + 13:RWTS = 896:D$ =  CHR$ (13) +  CHR$ (
     4):RD = 1:WRT = 2:BFR = BUFR + 10
600  GOSUB 1020: POKE IBUFP,BL: POKE IBUFP + 1,BH
610  HOME : VTAB 20: PRINT "INSERT DISK TO BE CUST
     OMIZED"
620  VTAB 22: PRINT "THEN PRESS ";: INVERSE : PRINT
     " RETURN ";: NORMAL : GET Z$: PRINT Z$
630  REM

***.READ CATALOG INTO BUFFER.***

640 TRK = 17:SECTR = 0
650  POKE ITRK,TRK: POKE ISECT,SECTR: POKE ICMD,RD
     : CALL RWTS
660 TRK =  PEEK (BUFR + 1):SECTR =  PEEK (BUFR + 2
     )
670  POKE ITRK,TRK: POKE ISECT,SECTR: CALL RWTS
675  REM

***.MAIN ROUTINE.***

680  HOME : VTAB 20: PRINT "(I)NVERSE OR (N)ORMAL
     ";: GET A$: PRINT A$:VTB = 12:HTB = 8
690 INV = 0
700  IF A$ = "I" THEN INV = 1
710 Z1$ = "00000000001111111111222"
720 Z2$ = "12345678901234567890123"
730  VTAB 10: HTAB HTB: PRINT Z1$: HTAB HTB: PRINT
     Z2$
740 TB = 12: FOR Z = 0 TO 6
750  VTAB TB + Z: HTAB 7: PRINT "+";: IF INV THEN
      INVERSE
760  FOR J = 1 TO 23: PRINT " ";: NEXT : NORMAL : PRINT
     "+"
770  NEXT
780  VTAB 20: CALL  - 958: HTAB 5: PRINT "INPUT LI
     NES OF CUSTOM CATALOG"
790  VTAB 22: PRINT " PRESS ";: INVERSE : PRINT "
     ESC ";: NORMAL : PRINT " TO CHANGE DISPLAY ST
     ATUS"
800  GOSUB 30: NORMAL
810  REM
```

```
***.WRITE SECTOR TO DISK.***

820  PRINT : VTAB 20: CALL  - 958: PRINT "IS THIS
     WHAT YOU WANT? (Y/N) ";: GET ZZ$: PRINT ZZ$
830  IF ZZ$ = "N" THEN 680
840  POKE ICMD,WR: CALL RWTS
850  REM

***.CHANGE DISK VOLUME.***

860  PRINT : PRINT "IS ";: INVERSE : PRINT " DISK
     VOLUME ";:: NORMAL : PRINT " TO BE REPLACED?
     (Y/N) ";: GET Z$: PRINT Z$
870  IF Z$ <  > "Y" THEN 930
880 TRK = 2:SECTR = 2: POKE ITRK,TRK: POKE ISECT,S
     ECTR: POKE ICMD,RD: CALL RWTS
890  INPUT "INPUT 11 CHARACTER HEADER: ";MS$:LN =
      LEN (MS$): IF LN >  = 11 THEN MS$ =  LEFT$ (
     MS$,11): GOTO 910
900  FOR I = LN + 1 TO 11:MS$ = MS$ + " ": NEXT
910 J = 0: FOR I = BUFR + 176 TO BUFR + 186: POKE
     I, ASC ( MID$ (MS$,11 - J,1)) + 128:J = J + 1
     : NEXT
920  POKE ICMD,WR: CALL RWTS
930  REM

***.DISPLAY CATALOG AND FINISH.***

940  HOME : PRINT D$"CATALOGD1"
950  PRINT : PRINT "MORE CUSTOMIZING? (Y/N) ";: GET
     ZZ$: PRINT ZZ$
960  IF ZZ$ = "Y" THEN 610
970  TEXT : HOME : VTAB 10: HTAB 11: FLASH : PRINT
     " SEE YA' LATER!! ": NORMAL
980  VTAB 23: END
990  RETURN
1000  HOME : PRINT "***.ERROR.***": END
1010  REM

***.SET-UP IOB.***

1020  FOR I = 1 TO 25: READ I%: POKE 896 + I - 1,I
     %: NEXT I: RETURN
1030  DATA  160,136,169,3,32,181,183,96,1,96,1,0,1
     7,15,251,183,0,128,0,0
1040  DATA  2,2,254,96,1,59,236,236,59,59,236,236,
     59,27,236,28,29,30,236,236
```

# Apple Input and Menu Screens

Dan Jordan

*The formatting routines described here will let you personalize your programs, and also make them easier to use.*

Menus and formatted screens are two excellent tools that you can use to make programs more user-friendly. The programs given here are simple illustrations of how these techniques can be applied.

The "Menu Screen Routine" (Program 1) generates a menu and uses a selection bar to help the user choose program functions. To create the illusion of bar movement, lines 370–390 blot out the existing bar and lines 310–340 place a new bar on the next line.

The "Input Screen Routine" (Program 2) prints a form on the screen and indicates, by the length of the inverse blank field, the amount of data to be entered. A subroutine can be added to check for field length, if desired. The correction routine (lines 500–570) lets you correct a data section without affecting any other part of the program.

PRINT CHR$(7) rings a bell, prompting the user to answer a question printed on the screen. Using GET rather than INPUT saves keystrokes when answering these screen prompts (the RETURN key need not be hit to enter data that is input with a GET).

## Program 1. Menu Screen Routine

```
190  HOME
200  PRINT "********** MENU ************"
210  PRINT "1-STEP NUMBER 1"
220  PRINT "2-STEP NUMBER 2"
230  PRINT "3-STEP NUMBER 3"
240  PRINT "4-STEP NUMBER 4"
250  PRINT "5-STEP NUMBER 5"
260  PRINT "6-STEP NUMBER 6"
270  PRINT : PRINT
280  PRINT "HIT (RETURN) TO SELECT --OR--"
290  PRINT "HIT ANY OTHER KEY TO CHANGE SELECTION"
300 I = 2
310  VTAB I
```

```
315  HTAB 17
320  INVERSE
330  PRINT "      ";
340  NORMAL
350  GET X$
360  IF X$ =  CHR$ (13) THEN Y = I - 1: GOTO 490
370  VTAB I
380  HTAB 17
390  PRINT "      "
400 I = I + 1
410  IF I >  = 8 THEN I = 2
420  GOTO 310
490  VTAB 14
500  ON Y GOTO 1000,2000,3000,4000,5000,6000
1000  REM STEP NO.1 PROCEDURES
1010  PRINT "STEP NO. 1"
1020  GOTO 7000
2000  REM STEP NO.2 PROCEDURES
2010  PRINT "STEP NO. 2"
2020  GOTO 7000
3000  REM STEP NO.3 PROCEDURES
3010  PRINT "STEP NO. 3"
3020  GOTO 7000
4000  REM STEP NO.4 PROCEDURES
4010  PRINT "STEP NO. 4"
4020  GOTO 7000
5000  REM STEP NO.4 PROCEDURES
5010  PRINT "STEP NO. 5"
5020  GOTO 7000
6000  REM STEP NO.6 PROCEDURES
6010  PRINT "STEP NO. 6"
6020  GOTO 7000
7000  END
```

## Program 2. Input Screen Routine

```
180  CLEAR
190  DIM A$(5,100)
200  HOME
210  PRINT "*****NAME & ADDRESS INPUT *****"
220  PRINT "1-NAME----------"
230  PRINT "2-ADDRESS LINE 1"
240  PRINT "3-ADDRESS LINE 2"
250  PRINT "4-CITY STATE ZIP"
260  PRINT "5-TELEPHONE NO.-"
270  FOR I = 2 TO 6
280  VTAB I
290  HTAB 17
300  INVERSE
```

```
310  PRINT "                          ": REM  20 SPACES
320  NORMAL
330  NEXT I
335 X = 1
340  FOR I = 2 TO 6
345  VTAB I: HTAB 17
350  INPUT A$(I - 1,X)
360  NEXT I
370  PRINT : PRINT  CHR$ (7)
380  PRINT "DO YOU WISH TO MAKE A CORRECTION (Y OR
     N)?";
390  GET X$
400  IF X$ = "Y" THEN  GOTO 500
410  IF X$ = "N" THEN  GOTO 450
420  VTAB 7: GOTO 370
450  PRINT  CHR$ (7);
460  PRINT "DO YOU HAVE ANY MORE TO ENTER (Y OR N)
     ?";
470  GET X$
480  IF X$ = "N" THEN  GOTO 1000
485  IF X$ = "Y" THEN X = X + 1: GOTO 200
490  VTAB 8: GOTO 450
500  PRINT  CHR$ (7);
510  PRINT "ENTER LINE NUMBER YOU WISH TO CORRECT"
     ;
520  GET Y
530 Y = Y + 1
540  VTAB Y
550  HTAB 17
560  INPUT A$(Y - 1,X)
570  VTAB 7
580  GOTO 370
1000  REM  PRINT OR SAVE TO DISK
1010  END
```

# Using Commas, Colons, and Quote Marks in Apple INPUT Statements

Craig Peterson

*Want to make Applesoft INPUT more versatile? Try adding the "Comma Input Routine" to your programs. It also works with disk input.*

Have you ever wanted to input commas, colons, or quotation marks as part of an INPUT statement but found that your Apple kept coming back with EXTRA IGNORED?

You might have tried using GET statements, as described in Apple's *Contact 4*, but all that B$=B$+A$ stuff meant that you frequently had to endure string garbage cleanup delays. *Contact 6* offered an alternative solution, totally avoiding garbage collection, but it presented a subtle problem you might not have been aware of. The input routine used to fill the input buffer made no allowance for the high bit of each character in the input line. The routine used to fill the input buffer left the high bit set, just as it came from the keyboard, but Applesoft wanted the high bit to be zero for its string characters. The line will print correctly and will appear on the screen just like what you typed in. But you'll never get a match if you use a line like IF IN$ = "Q". In addition, if you try to VAL (IN$), when IN$ was input as "1234", you'll get a value of 0.

The solution to this dilemma is to use the program listed below. The subroutine shown in lines 1000 to 1020 (for Applesoft ROM BASIC) will gather any input (including commas, colons, and quote marks) and place it into the variable IN$. The only exempt characters are those used in the standard keyboard escape sequences.

Location 54572 is the Applesoft equivalent of the monitor's keyboard input routine, except that it strips the high bit from all of the input characters. So line 1000 fills the input buffer with normal Applesoft string characters gathered from the keyboard. Line 1010 finds the length of the string, and line 1020 finds the IN$ variable and sets its pointers to the

keyboard buffer. Then IN$ is relocated into RAM, away from the keyboard buffer.

It's not necessary for IN$ to be the first variable. Lines 1000–1020 can be placed anywhere in your program. The pointers for IN$ are found through locations 131 and 132, which hold the address of the pointers for the last-used variable. It's fast, it totally avoids string garbage build-up, and it's done in BASIC.

One additional note: Not only does this routine work well for keyboard input, but it also performs the same feat for disk input—a feature that can be particularly handy. Commas or other previously forbidden characters in the middle of a name file cause no difficulty when read from the disk. Please note, however, that this routine limits the size of an input string to 239 characters, just like the Applesoft INPUT statement does.

## Comma Input Routine

```
10  HOME : VTAB 4: PRINT "INPUT ANYTHING THAT YOU
    WANT..": PRINT : GOSUB 1000: PRINT : PRINT "V
    OILA..": PRINT : PRINT IN$: END
20 :
30  REM  LINES 1000 TO 1020 ARE A SUBROUTINE THAT
    PUTS ANY INPUT INTO IN$
40 :
1000  CALL 54572
1010  FOR B = 512 TO 751: IF  PEEK (B) <  > 0 THEN
    NEXT
1020 IN$ = "": POKE  PEEK (131) + 256 *  PEEK (132
    ) + 1,0: POKE  PEEK (131) + 256 *  PEEK (132 +
    2,2: POKE  PEEK (131) + 256 *  PEEK (132),B -
    512:IN$ =  MID$ (IN$,1): RETURN
```

# Undeletable Lines

Michael P. Antonovich

*Have you ever wanted to create "permanent" program lines (for instance, to put your name into a program in such a way that another computer user could not delete it and claim the program as his own)? With this program, you'll be able to do just that.*

Ordinarily, Applesoft does not allow you to enter lines with numbers greater than 63999. But it can be done. This article shows you how—and the lines so entered are effectively undeletable.

The Apple stores program lines beginning at memory location $800 (the $ sign indicates that the number is in hexadecimal). Enter the following small program to illustrate the way a program is stored.

```
1 REM
2 A=8
3 PRINT A
4 END
```

To see how the Apple stores this program, enter the monitor with a CALL-151. However, before listing the program, there is one other piece of information that you need to determine. To add lines to an existing program, you need to know where the current program ends in memory, and you can page through the memory to find the program's last byte. But that's the hard way. The Apple also stores the location of the last memory byte in locations $69 and $6A.

Filing that away temporarily, enter the monitor to check your program:

```
CALL-151
*69.6A
0069-1E 08
*800.81F
0800-00    07 08 01 00 B2 00 0F
0808-08    02 00 41 D0 38 00 16
0810-08    03 00 BA 41 00 1C 08
0818-04    00 80 00 00 00 FF FF
```

Although you may not recognize it, that's a memory dump of your program.

Now examine how your BASIC lines were translated to the above hex dump. The first byte, $00 at location $800, has no special meaning to our program. In fact, location $800 will always contain $00. The program lines begin after that point. Each line is prefixed by four bytes, and the first pair of bytes stores the starting byte address of the next line. In this example, locations $801 and $802 indicate that the next line will begin at memory location $807. Remember that the location is split into two bytes; note too that they are stored in what seems (to us humans) to be reversed order.

The second pair of bytes contains the line number assigned to the program line. In this example we started with the line number 1. Thus memory locations $803 and $804 indicate that the first line number is 1. In addition to the four bytes which prefix each line, each line is ended with single byte 00 to separate it from the next line. Therefore, there is a five-byte overhead for each program line used. If multiple statements are combined with a colon (using one byte) on a single line, you can save four bytes for each extra line you eliminate. If you have any doubts, try it yourself with the above program.

The second program line begins at memory address $807. The first four bytes indicate that the next statement will begin at location $80F and will have statement number 2. The next three bytes (41 D0 38) represent the tokens for the equality (A=8).

The information you need to understand these tokens is found in Appendix F and Appendix K of the *Applesoft Reference Manual.* Appendix F lists the decimal tokens for all of the keywords used by the Apple. However, when in the monitor, you need the hexadecimal equivalent of the tokens. For example, the hex equivalent for END is $80, for REM is $B2, and for PRINT is $BA. You might want to take the time now to write the hexadecimal equivalents next to the decimal values for all of the tokens.

Variable names, numbers, and strings are not listed in Appendix F. These must be constructed by using the individual ASCII character representations. In the manual, Appendix K gives the ASCII character set with equivalent decimal and hexadecimal codes. Again, you are interested in the hexadecimal codes. In this example, we need the A or $41 and the 8 or $38.

That leaves the equal sign (=). Both Appendix F and Appendix K give hex codes for the equal sign, but each gives a different code. Which one is correct? To construct a variable name, number, or string of characters, use Appendix K. Any symbol used in an arithmetic expression (such as =,(,), etc.) should be taken from Appendix F.

Finally, even though the program ends with an END statement, the Apple does not know that it has reached the end of the program. Instead, it recognizes the end when it finds the byte pair 00 00 in the locations where it expects to find the next line number.

Now that you know how the Apple interprets the program and stores it in memory, you are ready to add those "undeletable" lines. Normally, Applesoft only recognizes line numbers in the range 0–63999. Converting 63999 to hexadecimal, you get $F9FF—but you *can* write larger hexadecimal numbers than that in two bytes. In fact, you should be able to use numbers from $FA00 through $FFFF (that is, from 64000 through 65536). Even though the Apple won't let you enter such line numbers via the keyboard, you now know enough about how the Apple stores program lines to sneak them in.

Let's keep this example simple. Assume that you want to store your name and the date as REM statements; you could just as easily make them PRINT statements. In any case, these are the statements you want:

```
64000 REM MICHAEL P. ANTONOVICH
64001 REM JUNE 28, 1984
```

Now enter the monitor (CALL-151) and type the following:

```
81C:37 08 00 FA
820:B2 4D 49 43 48 41 45 4C
828:20 50 2E 20 41 4E 54 4F
830:4E 4F 56 49 43 48 00 4A
838:08 01 FA B2 4A 55 4E 45
840:20 33 30 2C 20 31 39 38
848:31 00 00 00
```

Before you return to Applesoft, you must reset the end-of-program pointer. If you don't, then any variables you store will write over the new lines you just added the first time you run your program.

This example now ends at memory location $84C, and that information must be put into locations $69 and $6A:

```
69:4C 08
```

Now, reenter Applesoft (using CTRL-C RETURN) and list the program. There are lines 64000 and 64001 at the end. Try to delete them. You can't! You can save this program, reload it, run it, and copy it, and still those two lines will be there. In fact the only way to get rid of them is to enter the monitor, find where you want the program to end, change the last two bytes to 00 00, and change the program ending location in addresses $69 and $6A. It's easy, but only if you know how.

REM statements are not the only things that you can put into undeletable lines. You can store anything you want, from program lines using tokens and character strings to machine language programs.

## Invisible and Unlistable

Using your monitor to manipulate BASIC, you can also keep lines from listing at all. To make this program print the 8, but to prevent line 3 from appearing in the program listing, simply change one hexadecimal number to 16:

```
0800 00 07 08 01 00 B2 00 16
```

In each case, note that the tokens we have been using are *not* machine language. All microcomputers use tokens to store keywords. All BASIC program lines are stored in the above manner, not in machine language, and the program lines must be interpreted each and every time that they are run.

# ML Tracer

Thomas G. Gordon
Apple Version by Tim Victor

*Attempting to debug a machine language program can sometimes be a trying experience, especially when the program always seems to exit into the twilight zone. And trying to study a program in ROM can be just as frustrating, even with a disassembler (where do branch instructions go?). Here's an excellent programming utility that helps solve this problem.*

Anyone who has ever worked with machine language knows how helpful it can be to be able to single-step through a program. "ML Tracer" allows you to step through a machine language routine one event at a time and print out the contents of all of the microprocessor registers after each instruction. It also allows you to follow all branches, jumps, and returns. The program will display the address, opcode, mnemonic, and operand of each instruction.

When Tracer is run, there will be a ten-second delay while the DATA statements are read. You'll then be asked for the hex address of the ML program you wish to examine. You can change the contents of any register, before each instruction is executed. Press A for the accumulator, X for the X register, Y for the Y register, S for the stack pointer, P for the processor status, or I for the instruction pointer (program counter). When you're through loading registers, press RETURN once more to execute the next instruction.

Hexadecimal numbers are used for all input and output. If you enter an address as a one-, two-, or three-digit hexadecimal number, zeros will be added on the left to make a four-digit number. If too many digits are entered, the rightmost four digits will be used. The same applies to changing the value in a register. The number that you enter will be converted to a two-digit hexadecimal number using the same rules.

## The Execution Subroutine

The program is written mostly in BASIC, but contains two machine language subroutines. The first, the initialization subroutine, copies the lowest three pages (768 bytes) of RAM,

which are used by BASIC, to a location above the BASIC program. The other, the execution subroutine, exchanges the two three-page blocks of data and loads all the registers with their saved values, then executes one instruction (which has been POKEd in from BASIC). When the instruction has been executed, the registers are saved and BASIC's original lower three pages of memory are restored.

Lines 10000–10031 contain four-character extended mnemonics for the 6502's instruction set. The fourth character is a tag code identifying the addressing mode of the instruction. In lines 110–120, the mode is identified and the proper subroutine is called.

There are several instructions which cannot be allowed to actually execute in the machine language subroutine. If any control transfer instructions (JMP, JSR, RTS, RTI, or a conditional branch) were executed, control would not be returned properly to the BASIC program. These instructions are simulated in BASIC instead, so that they appear to execute successfully. The SEI and CLI instructions are ignored, since interrupts are always disabled during the execution subroutine.

## How Does It Work?

The simplest way to see how the program works is to trace through an example. Suppose the instruction LDA #$20 resides at addresses $03C0–$03C1. For this instruction, the extended mnemonic is LDAB, where LDA stands for LoaD Accumulator, and B is the tag code for immediate addressing. The hexadecimal representation for LDA immediate is $A9, which is equivalent to decimal 169.

Line 50, the top of the main loop, calls the keyboard pause routine at line 7000, which also handles changing registers. In line 55, the variable C is loaded with 169 by PEEKing the memory addressed by B, the instruction pointer. The value of B, 960 in this example, is then converted to hexadecimal characters in line 2000 and PRINTed.

In line 60, NOP instructions are POKEd into the execution routine to take up space after one- or two-byte instructions. The hexadecimal value of the opcode is printed next, and then the mnemonic is retrieved from the array R$( ). If the mnemonic is a blank, this instruction is undefined and an error message is displayed. Otherwise, the standard (three-

character) mnemonic is PRINTed, the opcode is POKEd into the execution routine at OP, and the program counter is incremented to 961.

The ASCII code for B is 66, so the ON GOSUB in line 120 transfers control to line 400. Here, the symbol for the addressing mode, #$, is printed. The one-byte operand routine, at line 3000, PEEKs location 961, pointed to by the program counter. This number is POKEd into OP+1, then converted to hexadecimal and PRINTed. After incrementing the program counter to point to the start of the next instruction, a RETURN is executed at line 3000.

At line 5000, the execution routine is CALLed. The contents of the registers are displayed, and control passes back to line 120. Here, a GOTO 50 takes us back to the top of the loop, where the instruction at $3C2 will be executed.

## Tracing Is Educational Too

You will find that this program is most useful for testing small ML programs, such as those called as subroutines from BASIC. It's also good for examining sections of larger programs when you're not sure how a particular routine works. If you're learning machine language, you'll find that the register display is an enormous help in understanding the effects and side effects of each instruction, especially the bits (flags) of the processor status register.

Do be careful, though. Any program is vulnerable when dealing with something as powerful as machine language, and this one is no exception. There are more ways to kill a BASIC program from ML than anyone can name in one sitting, so always be conscientious about saving your programs. After you type this one in, save it before you even think about running it. One typographical error could cause the program to erase itself, or at least lock up the computer.

There are also some ML programs that this tracer can't follow, such as those which disconnect the keyboard or video display (whether intentionally or accidentally). If everything is saved on disk or tape (for real security, take the diskette or cassette out of the drive), you can experiment as much as you want, and then if disaster struck all you'd have to do is just turn the computer off and reload the program.

# ML Tracer

```
10  GOSUB 6000
35  POKE A,0: POKE X,0: POKE Y,0: POKE P,52: POKE
    S,255
40  PRINT "START ADDRESS (HEX)";: INPUT H$
42  IF H$ = "" THEN H$ = "C000"
45 H$ =  RIGHT$ (H$,4): GOSUB 1500:B = D: PRINT "A
    NY KEY TO STEP"
50  GOSUB 7000:D =  FRE (0)
55  PRINT :C =  PEEK (B):D = B: GOSUB 2000: PRINT
    H$" ";
60  POKE OP + 1,234: POKE OP + 2,234
70 D = C: GOSUB 2000: PRINT  RIGHT$ (H$,2)" ";
80  IF R$(C) = "" THEN  PRINT "INVALID OPCODE": PRINT
    : GOTO 35
90 R$ =  LEFT$ (R$(C),3): PRINT R$" ";: POKE OP,C:
    B = B + 1
100  IF R$ = "BRK" THEN  PRINT : GOTO 35
110 U$ =  RIGHT$ (R$(C),1): IF U$ = " " THEN  GOSUB
    200: GOTO 50
120  ON  ASC (U$) - 64 GOSUB 300,400,500,600,700,8
    00,900,1000,1100,1200,1300: GOTO 50
199  REM      >IMPLIED MODE<
200  IF R$ = "RTS" THEN  GOSUB 4000:B = D: GOSUB 4
    000:B = D * 256 + B + 1: GOSUB 5005: RETURN
203  IF R$ <  > "RTI" THEN 208
205  GOSUB 4000: POKE P,D: GOSUB 4000:B = D: GOSUB
    4000:B = D * 256 + B: GOSUB 5005: RETURN
208  IF R$ = "SEI" OR R$ = "CLI" THEN  GOSUB 5005:
     RETURN
210  GOSUB 5000: RETURN
299  REM      >ABSOLUTE MODE<
300  PRINT "$";: GOSUB 2500
310  IF R$ = "JMP" THEN B =  PEEK (OP + 1) +  PEEK
    (OP + 2) * 256: GOSUB 5005: RETURN
320  IF R$ <  > "JSR" THEN 340
330 B = B - 1:D =  INT (B / 256): GOSUB 3500:D = B
     -  INT (B / 256) * 256: GOSUB 3500
335 B =  PEEK (OP + 1) +  PEEK (OP + 2) * 256: GOSUB
    5005: RETURN
340  GOSUB 5000: RETURN
399  REM      >IMMEDIATE MODE<
400  PRINT "#$";: GOSUB 3000: GOSUB 5000: RETURN
499  REM      >ZERO PAGE MODE<
500  PRINT "$";: GOSUB 3000: GOSUB 5000: RETURN
599  REM      >ABSOLUTE,X<
600  PRINT "$";: GOSUB 2500: PRINT ",X";: GOSUB 50
    00: RETURN
699  REM      >ABSOLUTE,Y<
```

```
700  PRINT "$";: GOSUB 2500: PRINT ",Y";: GOSUB 50
     00: RETURN
799  REM      >(INDIRECT,X)<
800  PRINT "($";: GOSUB 3000: PRINT ",X)";: GOSUB
     5000: RETURN
899  REM      >(INDIRECT),Y<
900  PRINT "($";: GOSUB 3000: PRINT "),Y";: GOSUB
     5000: RETURN
999  REM      >ZERO PAGE,X<
1000  PRINT "$";: GOSUB 3000: PRINT ",X";: GOSUB 5
     000: RETURN
1099  REM     >ZERO PAGE,Y<
1100  PRINT "$";: GOSUB 3000: PRINT ",Y";: GOSUB 5
     000: RETURN
1199  REM     >RELATIVE JUMP<
1200  PRINT "TO ";:D =  PEEK (B):B = B + 1:D = D -
     (D > 127) * 256:D = B + D:B1 = D
1210  GOSUB 2000: PRINT "$"H$;:BM = BM( INT (C / 6
     4)):BC =  INT ( PEEK (P) / BM):BC = BC - 2 *
      INT (BC / 2)
1220  IF BC = ( INT (C / 32) - 2 *  INT (C / 64)) THEN
     B = B1
1230  GOSUB 5005: RETURN
1299  REM     >INDIRECT JUMP<
1300  PRINT "(";: GOSUB 2500: PRINT ")";:B =  PEEK
     (OP + 1) +  PEEK (OP + 2) * 256
1310 B =  PEEK (B) +  PEEK (B + 1) * 256: GOSUB 50
     05: RETURN
1499  REM     > HEX TO DEC <
1500 D = 0: FOR I = 1 TO  LEN (H$):J =  ASC ( MID$
     (H$,I,1)) - 48:D = D * H + J - 7 * (J > 9): NEXT
     : RETURN
1999  REM     > DEC TO HEX <
2000 H$ = "": FOR I = 1 TO 4:E =  INT (D / H):J =
     D - E * H:H$ =  CHR$ (J + 48 + 7 * (J > 9)) +
     H$:D = E: NEXT
2005  RETURN
2499  REM     > 2BYTE OPERAND <
2500 D =  PEEK (B + 1): POKE OP + 2,D: GOSUB 2000:
      PRINT  RIGHT$ (H$,2);: GOSUB 3000:B = B + 1:
      RETURN
2999  REM     > 1BYTE OPERAND <
3000 D =  PEEK (B): POKE OP + 1,D: GOSUB 2000: PRINT
      RIGHT$ (H$,2);:B = B + 1: RETURN
3499  REM     > PUSH <
3500 J =  PEEK (S): POKE ML + 512 + J,D
3505  IF J = 0 THEN  PRINT : PRINT "WARNING: STACK
     OVERFLOW":J = 256
3510  POKE S,J - 1: RETURN
```

```
3999  REM     > POP <
4000 J =  PEEK (S):D =  PEEK (ML + 513 + J)
4005  IF J = 255 THEN  PRINT : PRINT "WARNING: STA
     CK UNDERFLOW":J =  - 1
4010  POKE S,J + 1: RETURN
4999  REM     > EXECUTE ONE INSTRUCTION <
5000  CALL (ML + 23)
5005  PRINT : FOR K = 0 TO 4:D =  PEEK (A + K): GOSUB
     2000
5010  PRINT  MID$ (" A= X= Y= S= P=",3 * K + 1,3);
     : PRINT  RIGHT$ (H$,2);: NEXT : PRINT : RETURN

5999  REM     > INITIAL STUFF <
6000 ML = 2 * 4096 + 8 * 256
6001 A = ML + 240:X = A + 1:Y = X + 1:S = Y + 1:P =
     S + 1:H = 16:OP = ML + 92
6002  DIM R$(255): DIM BM(3): FOR I = 0 TO 3: READ
     B:BM(I) = B: NEXT
6003  FOR T = 0 TO 255: READ R$(T): NEXT
6004  READ R$: IF R$ <  > "END" THEN  PRINT "ERROR
      IN OPCODES": PRINT "CHECK FOR TYPO'S": END
6005 I = 0: FOR T = ML TO ML + 164: READ B: POKE T
     ,B:I = I + B: NEXT
6008  IF I <  > 17737 THEN  PRINT "ERROR IN ML DAT
     A": PRINT "CHECK FOR TYPO'S": END
6010  CALL ML
6015  HOME : PRINT "6502 ML TRACER"
6020  RETURN
6999  REM    > PAUSE <
7000  GET A$: IF A$ = "" THEN 7000
7010  IF A$ = "I" THEN D = B:L = 4: GOSUB 7100:B =
     D: GOTO 7000
7020  IF A$ = "A" THEN D =  PEEK (A):L = 2: GOSUB
     7100: POKE A,D: GOTO 7000
7030  IF A$ = "X" THEN D =  PEEK (X):L = 2: GOSUB
     7100: POKE X,D: GOTO 7000
7040  IF A$ = "Y" THEN D =  PEEK (Y):L = 2: GOSUB
     7100: POKE Y,D: GOTO 7000
7050  IF A$ = "S" THEN D =  PEEK (S):L = 2: GOSUB
     7100: POKE S,D: GOTO 7000
7060  IF A$ = "P" THEN D =  PEEK (P):L = 2: GOSUB
     7100: POKE P,D: GOTO 7000
7065  IF A$ =  CHR$ (3) THEN  STOP
7070  RETURN
7100  PRINT A$"=";: GOSUB 2000:A$ = H$: INPUT H$: IF
     H$ = "" THEN H$ = A$
7110 H$ =  RIGHT$ (H$,L): GOSUB 1500: RETURN
9000  DATA 128,64,1,2
10000  DATA BRK ,ORAF,,,,ORAC,ASLC,
```

```
10001  DATA PHP ,ORAB,ASL ,,,ORAA,ASLA,
10002  DATA BPLJ,ORAG,,,,ORAH,ASLH,
10003  DATA CLC ,ORAE,,,,ORAD,ASLD,
10004  DATA JSRA,ANDF,,,BITC,ANDC,ROLC,
10005  DATA PLP ,ANDB,ROL ,,BITA,ANDA,ROLA,
10006  DATA BMIJ,ANDG,,,,ANDH,ROLH,
10007  DATA SEC ,ANDE,,,,AMDD,ROLD,
10008  DATA RTI ,EORF,,,,EORC,LSRC,
10009  DATA PHA ,EORB,LSR ,,JMPA,EORA,LSRA,
10010  DATA BVCJ,EORG,,,,EORH,LSRH,
10011  DATA CLI ,EORE,,,,EORD,LSRD,
10012  DATA RTS ,ADCF,,,,ADCC,RORC,
10013  DATA PLA ,ADCB,ROR ,,JMPK,ADCA,RORA,
10014  DATA BVSJ,ADCG,,,,ADCH,RORH,
10015  DATA SEI ,ADCE,,,,ADCD,RORD,
10016  DATA ,STAF,,,STYC,STAC,STXC,
10017  DATA DEY ,,TXA ,,STYA,STAA,STXA,
10018  DATA BCCJ,STAG,,,STYH,STAH,STXI,
10019  DATA TYA ,STAE,TXS ,,,STAD,,
10020  DATA LDYB,LDAF,LDXB,,LDYC,LDAC,LDXC,
10021  DATA TAY ,LDAB,TAX ,,LDYA,LDAA,LDXA,
10022  DATA BCSJ,LDAG,,,LDYH,LDAH,LDXI,
10023  DATA CLV ,LDAE,TSX ,,LDYD,LDAD,LDXE,
10024  DATA CPYB,CMPF,,,CPYC,CMPC,DECC,
10025  DATA INY ,CMPB,DEX ,,CPYA,CMPA,DECA,
10026  DATA BNEJ,CMPG,,,,CMPH,DECH,
10027  DATA CLD ,CMPE,,,,CMPD,DECD,
10028  DATA CPXB,SBCF,,,CPXC,SBCC,INCC,
10029  DATA INX ,SBCB,NOP ,,CPXA,SBCA,INCA,
10030  DATA BEQJ,SBCG,,,,SBCI,INCI,
10031  DATA SED ,SBCE,,,,SBCD,INCD,
10032  DATA END
20000  DATA 162,0,181,0,157,0,41,189
20001  DATA 0,1,157,0,42,189,0,2
20002  DATA 157,0,43,232,208,236,96,120
20003  DATA  162,0,181,0,168,189,0,41
20004  DATA 149,0,152,157,0,41,189,0
20005  DATA 1,168,189,0,42,157,0,1
20006  DATA 152,157,0,42,189,0,2,168
20007  DATA 189,0,43,157,0,2,152,157
20008  DATA 0,43,232,208,213,186,138,174
20009  DATA 243,40,154,141,243,40,172,242
20010  DATA 40,174,241,40,173,244,40,72
20011  DATA 173,240,40,40,234,234,234,8
20012  DATA 141,240,40,104,141,244,40,142
20013  DATA 241,40,140,242,40,186,138,174
20014  DATA  243,40,154,141,243,40,162,0
20015  DATA 181,0,168,189,0,41,149,0
20016  DATA 152,157,0,41,189,0,1,168
```

```
20017  DATA 189,0,42,157,0,1,152,157
20018  DATA 0,42,189,0,2,168,189,0
20019  DATA 43,157,0,2,152,157,0,43
20020  DATA 232,208,213,88,96
```

# All About the Status Register

Louis F. Sander

*The status registers have always been a mystery to the beginning machine language programmer. This article will help clear up the mystery.*

All but the simplest machine language programs make use of the 6502's seven processor status flags, and any ML programmer worth his salt masters their functions and uses. Like almost everything in ML programming, the flags operate in a straightforward and unambiguous way, but they are full of mystery for the beginner.

If you've started ML programming, but are confused by that NV-BDIZC business, this article will help you understand it. It includes a fully explained ML demo program.

These explanations will assume that you have some ML knowledge and at least a beginning grasp of hexadecimal arithmetic.

Let's start by defining a *register,* which is a circuit inside a processor. Registers have the characteristics of memory locations, in that data can be written to them or read from them. But they often don't have addresses as such, since they are used internally by the microprocessor itself. The accumulator is the most familiar register, but there are many others in your computer.

The 6502 has an internal 8-bit register, variously called the flags register, processor status register, or P register, the bits of which are set or cleared by the results of various operations. In this context, *set* means equal to 1, and *cleared* means equal to 0. At times the bits are set and cleared, or *conditioned,* automatically by the 6502 chip itself; other times they are conditioned by specific program instructions. Any book on 6502 programming will show you each instruction's effect on the status bits.

## Bit Branches

Programs can check these bits and use the results of the check for whatever purpose the programmer has in mind, often to decide on a branch. The bits are sometimes called flags, and

indeed, they work like the little red flags on rural mailboxes—the postal patron can raise the flag to let the mailman know there's outgoing mail, and the mailman can lower it to signal he's emptied the box. Here are the names and purposes of the eight bits in the status register, moving from left (high-order bit) to right (low-order bit):

N (bit 7)—Negative flag. (Some books call it S, for sign.) The N flag matches the high bit of the result of whatever operation the processor has just completed. If you load $FF (1111 1111) into the Y register, for example, since the high bit of the Y register is set, the N flag will be set, too. ML programmers make good use of the N flag. (By the way, even though this is the eighth bit, we call it bit 7, because computers start numbering things at 0.) In a computer technique called twos complement arithmetic, the high-order bit of a number is set to 1 if the number is negative, and cleared to 0 if it's positive, and that's where the N flag gets its name.

V (bit 6)—Overflow flag. This flag is important in twos complement arithmetic, but elsewhere it is rarely used. In the interest of simplicity, we'll say no more about it.

Bit 5 has no name, and is always set to 1. Since nothing can change it, it is of no use to the programmer.

B (bit 4)—Break flag, set whenever a BRK instruction is executed, clear at all other times. Rarely used by beginners.

D (bit 3)—Decimal flag. When D is set by the programmer, the 6502 does its arithmetic in BCD, binary coded decimal, which is yet another exotic type of computer math. Fortunately for nonexperts, it's seldom used, and the beginner's only concern with the D flag is to be sure it is not set unintentionally, because when it *is*, program behavior can be bizarre.

I (bit 2)—Interrupt mask. When this bit is set, the computer will not honor interrupts, such as those used for keyboard scanning in many computers. It is widely used, but so different from the other flags that we'll say no more about it.

Z (bit 1)—Zero flag. This one's used a great deal, and basically the computer sets it when the result of any operation is zero. Load the X register with $00, and you set the zero flag. Subtract $32 from $32, and you do the same. Many 6502 instructions affect the Z flag, and there's always a "zero or not-zero" aspect to it, but it's not always obvious to the novice when a zero condition exists. This is probably the most im-

portant of the flags, and if you master it, mastery of the others will be easy.

C (bit 0)—Carry flag. Carry is set whenever the accumulator rolls over from $FF to $00 (just like the odometer on a car, rolling over from all nines to all zeros). It's also set by various rotation and comparison instructions. The carry flag is about as important as the Z flag, and a little more mysterious, at least to me, but its operation is really rather simple.

## 6502 Monitor

The foregoing brief description of the 6502's 8-bit processor status register and the seven status flags it contains may have cleared some mystery away, but it surely isn't comprehensive. That sort of description is found in ML programming books, to which you are now referred, and which will be much easier to understand once you've mastered what is presented here. Let's get that mastery by running a simple test program, using a machine language *monitor* to observe its effects on the status register.

A monitor is nothing more than a machine language program that makes it easier to work with other ML programs. Apples have a simple monitor built into the ROM.

The monitor is a wonderful tool for the beginning ML programmer, and if you've dabbled with ML, you've at least used it to examine memory locations and to save ML programs on tape or disk. I used mine for those things for many months, but never paid much attention to the registers display. That's the line of labeled numbers the monitor prints on the screen when a BRK instruction is encountered. (You can also get a register display by typing CTRL-E and RETURN at a monitor prompt.) It looks like this:

```
330D-  A=00  X=5E  Y=04  P=30  S=F8
```

The first number shows the contents of the PC register, the address in the 6502's program counter, which is nothing more than the address of the next instruction to be executed. Because of various quirks, the value shown in the register display is the address two bytes after the BRK. (The register display you get with CTRL-E does not include the PC value.)

A, X, and Y show the contents of the accumulator, X, and Y registers, respectively, at the moment the monitor was activated. P gives the contents of the processor status register,

expressed in hexadecimal form. People with 6502s in their cerebral cortices may be able to determine individual flag statuses from a hex display, but it's a burdensome interpretation for the rest of us. Who can figure out whether $FB means the Z flag is set or clear? Not me, I can guarantee you. The table is a handy guide for interpreting that byte. With it, you can tell at a glance which flags are set or cleared in a given status byte, and just what each flag means. And that ability can be a golden key to better machine language programming.

## Decoding Status Displays

| First Digit | | Second Digit | |
|---|---|---|---|
| 0 | | 0 | D I Z C |
| 1 | | 1 | D I Z **C** |
| 2 | N V - B | 2 | D I **Z** C |
| 3 | N V - **B** | 3 | D I **Z C** |
| 4 | | 4 | D **I** Z C |
| 5 | | 5 | D **I** Z **C** |
| 6 | N **V** - B | 6 | D **I Z** C |
| 7 | N **V - B** | 7 | D **I Z C** |
| 8 | | 8 | **D** I Z C |
| 9 | | 9 | **D** I Z **C** |
| A | **N** V - B | A | **D** I **Z** C |
| B | **N** V - **B** | B | **D** I **Z C** |
| C | | C | **D I** Z C |
| D | | D | **D I** Z **C** |
| E | **N V** - B | E | **D I Z** C |
| F | **N V - B** | F | **D I Z C** |

This table decodes two-digit hex displays of the processor status register. Bold face indicates bit set; regular face, bit clear.

S gives the value of the stack pointer, which is yet another useful value that's beyond our present scope. The value will vary from time to time and from machine to machine.

## Stepping Through Flags

Now that you've seen a description of the register display, plus that handy table, let's use them to experiment with the important flags. Our experiment will have the dual benefit of making us more fluent in ML, and giving us practice using the register display.

The program at the end of this article is an instructive, but do-nothing, ML program that occupies an innocuous corner of

memory. From left to right, each line shows a memory address, the bytes held by it and maybe its upward neighbor, and the mnemonic for the machine language instruction that those bytes represent. The program's first seven lines set all the 6502's flags and registers to zero, then break to the monitor, where we can review their status.

## Single, Simple Operations

The rest of the program is a series of single, simple operations, each followed by a break to the monitor. We're about to go through them one by one, and see what happens to the negative, break, zero, and carry flags. We'll leave V, D, and I for another day, for the reasons previously mentioned.

The figure will be used to track our demonstration.

Steps 1–3. Our first step will be to put the ML demo program into memory. Do it now, by carefully following Steps 1, 2, and 3. If you've never worked with ML before, don't worry—the process is easy, and we'll take you through it step by step. When you finish Step 3, come back here for further instructions.

At the end of Step 3, the monitor should still be active, and your screen should be showing you its distinctive monitor prompt. You're now ready to run the ML demo program, which you do by executing your monitor's G command. Be sure to use the correct syntax; it is illustrated in the figure. Monitor commands are fussy about spaces, etc., so pay close attention to details at this point. Now go do Step 4, which will start execution of the machine language routine at address $3300. That routine will run until a BRK instruction is executed, at which point processing will stop and the monitor's register display will appear on the screen. When that happens, which should be immediately, come back here.

Step 4. Study the register display, disregarding S, and observe that A, X, and Y are all set to $00. Use the figure to confirm that $30 means that all P flags are clear, except for the B and the meaningless bit that's always set. Remember what the B flag is for, and it will be easy to see why it's set. Our program was designed to zero everything out, and it worked as it was designed. So far, so good. (If things are not so good, you've made a mistake. Repeat your work from the beginning.)

## Nothing Has Changed

Step 5. Now perform Step 5, and notice what has happened. The program has loaded $80 (1000 0000) into the accumulator, and the monitor AC display so indicates. Since the leftmost bit of $80 is a 1, the computer set its own N flag. The program counter has advanced, but nothing else has changed. (If your stack pointer changed, never mind—the monitor, not our program, changed it.) The BRK brought us back to the monitor. Simple, isn't it?

Step 6. The LDA has loaded $7F (0111 1111) into the accumulator, setting N to match its highest bit. The register display shows the $7F, and proves that N is now clear, while all other flags remain the same. Now do Step 7.

Step 7. Putting $00 (0000 0000) in the accumulator sets the Z bit, since zeros beget zeros. Notice how the PC is stepping right along with us, and do Step 8.

Step 8. $FF (1111 1111) is *not* a zero, so the zero flag is cleared. Its high bit is a 1, so the N flag is set. Move on to the next step.

Step 9. The ADC instruction adds 1 to the accumulator. Like driving another mile when the speedometer reads 99999, this rolls the accumulator over to $00 (0000 0000). We can tell when this happens, because the rollover automatically sets the carry flag. The carry bit is often used in just this way, to tell when a counter has reached its maximum. In our example, Z is also set, since the operation resulted in a zero. When you've absorbed those simple details, go on to Step A.

## Screen Dialogue

**Step 1** To activate the monitor, type CALL-151, then press RETURN.

```
]CALL -151
```

**Step 2** Put the program into memory by making these entries *exactly* as shown. Press RETURN at the end of each line.

```
*3300:D8 18 A9 00 AA A8 C9 FF
*3308:00 A9 80 00 A9 7F 00 A9
*3310:00 00 A9 FF 00 69 01 00
*3318:69 01 00 C9 02 00 00 00
```

**Step 3** Check your work by entering this command and comparing your screen display with the program.

```
*3300L
```

**Step 4** Type the G command, then press RETURN. When this line appears, return to the text.

```
*3300G
330A-    A=00 X=00 Y=00 P=30 S=C9
```

**Step 5** This and the following steps are identical to Step 4, except for the numbers entered and displayed.

```
*3309G
330D-   A=80 X=00 Y=00 P=B0 S=C7
```

**Step 6** As above.

```
*330CG
3310-   A=7F X=00 Y=00 P=30 S=C5
```

**Step 7** As above.

```
*330FG
3313-   A=00 X=00 Y=00 P=32 S=C3
```

**Step 8** As above.

```
*3312G
3316-   A=FF X=00 Y=00 P=B0 S=C1
```

**Step 9** As above.

```
*3315G
3319-   A=00 X=00 Y=00 P=33 S=BF
```

**Step A** As above.

```
*3318G
331C-   A=02 X=00 Y=00 P=30 S=BD
```

**Step B** As above. This is the last step in our demonstration.

```
*331BG
331F-   A=02 X=00 Y=00 P=33 S=BB
```

## Bump A Counter

Step A. The last operation did not roll over the accumulator, so the carry bit was cleared. What it did was to add 1 to the zero in the accumulator, giving a result of 2. How on earth do 1 + 0 = 2? The answer is in the carry bit. An ADC adds its operand plus the carry bit to the contents of the accumulator, then reconditions C based on the result. That's very useful, because often when a counter rolls over, we want to increment a higher-order counter, so nothing gets lost in the counting. Many programs look for the carry bit, and bump a counter if it's set. Our own little program didn't go that far, but it did show us how such things can be done. Now do the next G.

Step B. What's this? We compared a 2 to a 2, and the zero and carry flags got set. That's a special use of flags in comparing numbers. CMP and the other comparison instructions don't store their results anywhere, but they *do* condition the N, Z, and C flags in a special way that facilitates branching after the comparison. Read up on the CMP, CPX, and CPY instructions for full information on how they set the flags.

We're now at the end of our flag-waving tour. If you kept with us this far, you're in the know about some elementary but important attributes of the processor status register, and you may have improved your knowledge of your monitor. Dig into those ML texts that you didn't understand last time, and

you'll be surprised how easy they've become. If you're really feeling like an expert, come up with a branch instruction to take our program back to $3300.

## Machine Language Demonstration Program

```
3300 D8          CLD
3301 18          CLC
3302 A9 00       LDA #$00
3304 AA          TAX
3305 A8          TAY
3306 C9 FF       CMP #$FF
3308 00          BRK
3309 A9 80       LDA #$80
330B 00          BRK
330C A9 7F       LDA #$7F
330E 00          BRK
330F A9 00       LDA #$00
3311 00          BRK
3312 A9 FF       LDA #$FF
3314 00          BRK
3315 69 01       ADC #$01
3317 00          BRK
3318 69 01       ADC #$01
331A 00          BRK
331B C9 02       CMP #$02
331D 00          BRK
```

# Chapter 5

# Sound and Graphics

# Introduction

Sound and graphics have always been important elements of many home computer applications, and the programs in this chapter will help you get the most out of your Apple's sound and graphics capabilities.

Every programmer will enjoy Blaine Mathieu's "Apple Sounds," a comprehensive two-part guide to creating custom sounds on the Apple. It shows you how to produce a variety of sound effects—but it goes beyond that. Using the programs in the articles, you'll be able to create and play musical compositions too.

Other programs in this chapter will let you put Apple graphics to work, and the results may amaze you. J.F. Johnson's "Apple Shape Generator," for instance, takes the work out of creating shapes on your Apple. It automatically performs all necessary numerical conversions and creates the shape table too.

Equally versatile is "The Apple Hi-Res Painter," by James Totten. It's a full-featured, menu-driven drawing tool that will let you create virtually any drawing you desire.

Would you like to add depth to your graphics creations? Tim R. Colvin's "3-D Plotting" illustrates one approach—and you can easily modify his programs to produce 3-D drawings of your own.

Finally, Chayim Avinor's "Spiralizer" creates intricate spiral shapes under your direction. You can draw shapes one at a time or combine them for special effects.

# Apple Sounds—from Beeps to Music, Part 1

Blaine Mathieu

*In this first of two articles, the author takes you from creating the simplest possible sound on the Apple to producing musical notes. Several useful demonstration programs are included.*

Since I first acquired an Apple II+ about a year and a half ago, I have been fascinated by the strange noises I often hear. In this first of two articles I hope to save you all the trouble I went through in learning how to use Apple sounds. Readers who already understand how to use CTRL-G and −16336 may want to skip the next section and go on to "Paddle Sounds."

## Beeps and Clicks

Before you read this section, you should enter Program 1 on your computer and save it. Then run the program. If you entered it correctly, you should see SOUND at the top, a line from the program, and a small menu.

The first sound that you ever heard from your Apple's speaker was probably the so-called bell sound. You can reproduce this in immediate mode by holding down the control key (CTRL) and pressing the G key. In line 30 a CHR$(7) is being printed (7 is the numeric code for CTRL-G). Note: If you are in Integer BASIC, you will have to use the format shown in line 35. In this line you'll see a PRINT with two quotes. Inside these quotes is a CTRL-G. The REM statement in line 37 shows how to type line 35. (As you can see, control characters don't show up in a line listing or when you type them. An interesting side effect is that when you LIST your program, you will hear all the bell sounds in your program that are printed using the method in line 35.)

In Program 1, the computer waits for you to hit a key. If you hit R, it will repeat any sound that might be produced by the above program lines. If you hit C, you will proceed to the next sound in Program 1. Any other key (except RESET) will cause no change.

## Clicking

Now hit C to go on to the second sound (SOUND #2). In this program a simple FOR-NEXT loop is set up to beep the Apple's speaker ten times. Note the semicolon at the end of line 80; this prevents the screen from scrolling. If I hadn't used the semicolon, the imaginary cursor would move down the screen as each CTRL-G was printed until the screen started to scroll upward. In most cases, that's undesirable.

Looking at SOUND #3, you will notice the number −16336, which is the memory address of the Apple's speaker. Every time this address is accessed, the Apple gives a little push on its speaker, creating a small click. PEEKing, as I have done in line 130, is just one simple way of accessing this address. If you missed the sound the first time, press R to hear it again.

SOUND #4 includes another simple loop that will PEEK the speaker's memory address 100 times. Instead of typing −16336 every time I wanted to use it, I assigned −16336 to the variable NO (for NOise). You may use any variable you wish.

In SOUND #5, you'll notice line 250, which strings a lot of clicks together. This produces a longer noise than in SOUND #3 and a higher-pitched noise than in SOUND #4. As a rule, the closer your PEEKs, the higher-pitched your noise is going to be. In line 250 you will notice that you PEEKed −16336 a total of 15 times, a purely arbitrary number.

Finally, SOUND #6 demonstrates most of what you've learned about clicks. It uses a FOR-NEXT loop to cause line 320 to repeat 100 times. Line 320 has an assortment of minus and plus signs to show that it rarely makes a difference what you do to this location, as long as you access it.

Now on to something a little more exciting and complicated.

## Paddle Sounds

Program 2 requires paddles or a joystick. It's a simple BASIC program which reads a byte from the DATA statement and POKEs it into memory locations 768 ($300) to 786 ($312). The routine begins by CALLing 768. If you entered the program correctly, you should hear a fairly high-pitched whine;

as you move the paddles or joystick, this whine will change in pitch. You may leave the program by pressing RESET or CTRL-RESET, depending on your model.

Here is the source code for the machine language:

```
100             ORG  $300        ;768 DECIMAL
200  PDLZERO    EQU  $00
300  PDLONE     EQU  $01
400  PREAD      EQU  $FB1E
500  SPEAKER    EQU  C030
600  START      LDX  #PDLZERO    ;SET UP FOR PADDLE
                                  ZERO
700             JSR  PREAD       ;GET DELAY FROM
                                  PADDLE ZERO
800             STA  SPEAKER     ;TWEAK SPEAKER
900             LDX  #PDLONE     ;REPEAT PROCESS FOR
                                  PADDLE ONE
1000            JSR  PREAD
1100            STA  SPEAKER
1200            JMP  START       ;START OVER
```

Here is a quick explanation of how it works:

1. Put the paddle number in the X register.
2. Jump to the PREAD subroutine (see *Apple II Reference Manual*). PREAD acts as a delay, dependent on the paddle setting.
3. Tweak the speaker by accessing −16336 ($C030).
4. Repeat for next paddle.
5. Jump to beginning.

The pitch of the noise depends on how close together the tweaks are. The lower the paddle setting, the higher the pitch of the noise.

## Making Music

Now we'll look at a program that lets you produce notes (and thus music) on your Apple. Of course, there are some limitations; you won't be playing Beethoven's Fifth Symphony. You'll need peripheral boards to do things such as that. However, this program will let you do quite a lot with the hardware already in your Apple.

"Note Maker" (Program 3) is a simple BASIC program that POKEs in a machine language subroutine, sets up a few parameters, and CALLs the subroutine. The program continues running until a key is pressed. Try running it. If you've

never heard notes from your Apple, you may be quite surprised.

After the program has POKEd in the subroutine, it POKEs one random number (pitch) into location 768 ($300) and POKEs another random number (duration) into 769 ($301). The maximum value that can be POKEd into these locations is 255.

The source code is given below:

```
100             ORG  $300          ;768 DECIMAL
200  TWEAK      EQU  $C300
300  PITCH      EQU  $300
400  DURATION   EQU  $301
500             DS   2            ;MAKE SPACE FOR
                                   PITCH AND
                                   DURATION
600  START      LDA  TWEAK        ;TWEAK THE
                                   SPEAKER
700  BRANCH1    DEY
800             BNE  BRANCH2
900             DEC  DURATION     ;DURATION =
                                   DURATION-1
1000            BEQ  RETURN       ;IF DURATION = 0
                                   THEN RETURN
1100 BRANCH2    DEX
1200            BNE  BRANCH1
1300            LDX  PITCH
1400            JMP  START        ;CONTINUE TO
                                   SOUND
                                   NOTE
1500 RETURN     RTS               ;GO BACK TO OPER-
                                   ATING SYSTEM
```

In essence, the program works much like "Paddle Sounds." The main difference is that instead of the paddles controlling the pitch of the sound, locations 768 and 769 control the pitch and duration. The source code contains comments that should help you understand what is happening.

As you can see, whenever you want a sound routine, you're going to have to access location −16336 ($C030). Try experimenting with this program by POKEing in your own note values and hearing the results.

In Part 2, we'll look at a program called "Apple Music Writer," which will let you edit and play your own songs. Until then, experiment with the programs here. You're sure to come up with some surprising results.

## Program 1: Sounds and Variations

```
10 I = 10: HOME
20 PRINT "SOUND #1": PRINT : LIST 30,37
30  PRINT  CHR$ (7)
35  PRINT "": REM  CTRL-G
37  REM  PRINT"CTRL-G"
40  GOTO 10000
50 I = 50: HOME
60 PRINT "SOUND #2": PRINT : LIST 70,90
70  FOR LOOP = 1 TO 10
80  PRINT  CHR$ (7);
90  NEXT
100  GOTO 10000
110 I = 110: HOME
120 PRINT "SOUND #3": PRINT : LIST 130
130 X =  PEEK ( - 16336)
140  GOTO 10000
150 I = 150: HOME
160 PRINT "SOUND #4": PRINT : LIST 170,200
170 NO =  - 16336
180  FOR LOOP = 1 TO 100
190 X =  PEEK (NO)
200  NEXT
210  GOTO 10000
220 I = 220: HOME
230 PRINT "SOUND #5": PRINT : LIST 240,260
240 NO =  - 16336
250 X =  PEEK (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK (NO) +
      PEEK (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK (NO) +
      PEEK (NO)
260  REM  FIFTEEN TIMES
270  GOTO 10000
280 I = 280: HOME
290 PRINT "SOUND #6": PRINT : LIST 300,330
300 NO =  - 16336
310  FOR LOOP = 1 TO 100
320 X =  PEEK (NO) -  PEEK (NO) +  PEEK (NO) -  PEEK
     (NO) +  PEEK (NO) -  PEEK (NO) +  PEEK (NO)
330  NEXT
10000  POKE  - 16368,0: VTAB 20: HTAB 1: CALL  - 9
     58: PRINT "'R' FOR REPEAT, 'C' TO CONTINUE ";
     : GET A$
10010  IF A$ <  > "R" AND A$ <  > "C" THEN 10000
10020  IF A$ = "C" THEN 10100
10030  IF I = 10 THEN 30
10040  IF I = 50 THEN 70
```

```
100050  IF I = 110 THEN X =  PEEK ( - 16336): GOTO
     130
100060  IF I = 150 THEN 170
100070  IF I = 220 THEN 240
100080  IF I = 280 THEN 300
100100  IF I = 10 THEN 50
100110  IF I = 50 THEN 110
100120  IF I = 110 THEN 150
100130  IF I = 150 THEN 220
100140  IF I = 220 THEN 280
100150  TEXT : HTAB 1: PRINT "END OF LISTING#1"
```

## Program 2. Paddle Sounds

```
20  FOR LOC = 768 TO 786: READ BYTE: POKE LOC,BYTE
    : NEXT LOC
30  DATA 162,0,32,30,251,141,48,192,162,1,32,30,25
    1,141,48,192,76,0,3
40  CALL 768
```

## Program 3. Note Maker

```
10  FOR LOC = 770 TO 790: READ BYTE: POKE LOC,BYTE
    : NEXT
20  POKE 768, INT ( RND (1) * 255) + 1: POKE 769, INT
    ( RND (1) * 100) + 1: CALL 770:X =  PEEK ( -
    16384): IF X < 127 THEN  POKE  - 16368,0: GOTO
    20
30  DATA  173,48,192,136,208,5,206,1,3,240,9,202,2
    08,245,174,0,3,76,2,3,96
40  POKE  - 16368,0
```

# Apple Sounds—from Beeps to Music, Part 2

Blaine Mathieu

*In this article, the author combines ideas and programs from Part 1 to create the "Apple Music Writer." An easy-to-use tool for composing or reproducing songs, it offers a great variety of commands.*

"Apple Music Writer" is a program that allows any Apple owner to easily create music or reproduce favorite songs.

When you run the program, you'll see the title screen and hear a short tune. After the tune ends, you will be prompted by the word COMMAND? and a flashing cursor. At the top of the screen you should see a list of the possible commands; on the right will be a list of note names and their corresponding values.

## Commands

It's important that you understand and know how to use the commands, so it's helpful to go over them in the order that they appear on the screen. Press RETURN after each command, and feel free to experiment as we go along.

**A (ADDNOTE).** This command will let you begin your music file (song) and add to it. Every time you press A (and RETURN) you will be prompted to enter the note, a comma, and the duration. For example:

```
NOTE#1
NOTE,DURATION 128,200
```

The maximum usable note value is 255. The same is true for the duration value. After you've entered your values, you will hear what the new note will sound like in the song.

**E (EDIT).** If you've made a mistake, you can fix it by typing E (and, as always, RETURN). You will then be asked the number of the note you want to edit. If the note you want to edit is not part of the music file, you will be reprompted for the note number. If you entered a valid note number, you will be given the old values for that note and prompted for new values.

The same data entry rules apply as for ADDNOTE. Say you want to edit note number one and replace the old values with new ones of 64 and 200:

```
COMMAND? E
EDIT NOTE#1
NOTE#1 OLD: NOTE=128 DUR=200
NOTE,DURATION: 64,200
```

**P (PLAY).** Typing P will put you into Play mode. This will play your song and print it to the screen at the same time. Because it is both listing and playing your music file, the playing will not be at the same speed as in your program. It will be slower and more pronounced.

After entering P you will be prompted for the starting and ending notes. If you simply press RETURN instead of entering values, defaults will be set (D is the default) and the whole song will be played.

**S (SAVE).** This command will save your music file to disk, if you have DOS loaded into your computer. First you will be prompted for a filename, which will be the name used when the file is saved. Then you'll be prompted for the number of the first and last note of your file that you want saved to disk.

The next question is FOR FUTURE ADDITION?. If you answer Y, a file will be created that can be reloaded into Apple Music Writer at any time. You should use this option if you feel you may want to add more notes or edit your song at a later date. If you enter N, the file will be one that you can easily turn into a BASIC program to play your song when run.

If you answer the FOR FUTURE ADDITION? question with an N, you will be asked for the starting line number of your soon-to-be-created BASIC music program. Then you will be asked if you want a FULL LOADER PROGRAM. If you answer Y, the BASIC program created will include the necessary information. If you answer N, the routine will not be included. You would answer N if the program you wanted to add the music to already included some sort of "Note" routine.

(Note: To turn the text file created by this program into a BASIC program, you will need to type *EXEC filename.*)

Finally, you will be prompted to check for errors. If everything is all right, enter Y and the file will be saved. If you

enter N, you have to repeat the entire SAVE process. Here is an example of what the average SAVE command might include:

COMMAND? S
(Screen is cleared)
FILENAME? SONG.1
STARTING NOTE NUMBER:2
ENDING NOTE NUMBER:10
FOR FUTURE ADDITION? N
STARTING LINENUMBER: 100
FULL LOADER PROGRAM ? Y
IS EVERYTHING OK? Y

Your music file would now be saved under the filename SONG.1. The file would consist of notes two through ten, and the generated program would start at line 100. The generated program would include the machine language "Note" routine.

**L (LOAD).** If you answer Y to the FOR FUTURE ADDITION? question back in the SAVE command, you can load an old music file back into the computer. The catch is that you will lose any data that you entered into the computer beforehand. If you don't want to lose your data, then answer N to the question about losing your data. Just enter the appropriate filename, and you can manipulate or add to your data once again.

**N (NORPLAY).** As mentioned earlier, when you P (play and list) your song, it will play at a slower speed because it has to list the note values at the same time. To alleviate that problem, you can use the NORmal PLAY command. This will play your song in the same tempo as it will normally be played by your generated program. Just enter the proper values (or use the defaults) and listen.

**D (DELETE).** After entering D in response to the COMMAND? prompt, you will be asked which note or notes you want to delete. If you hit RETURN after the first question without typing anything else, the default will be used and the last note in the music file will be deleted. If you enter a value for the first question, you will be asked the number of the last note up to which you want to delete. The appropriate notes will then be deleted. You'll then go back to the COMMAND? mode.

**I (INSERT).** This command is the exact opposite of the Delete command. Simply answer the few setup questions and enter the data. Note: You cannot leave the Insert mode until

you have entered all the data you said you were going to enter.

**R (RESTART).** This command lets you start over at note one with a clean slate.

**C (CATALOG).** The Catalog command will return a fairly standard DOS catalog.

**Q (QUIT).** Use this command to exit the program, but you will lose all data that hasn't been saved to disk. If you quit by accident, GOTO 200 will usually let you reenter the program with no data lost.

**. (DOS).** Typing a period followed by any normal DOS command will execute that command. A common use for this might be:

```
COMMAND? .DELETE FILENAME
```

Caution: Some DOS commands will cause the Apple Music Writer to cease functioning, thus causing a loss of data.

**H (HARD).** If you have a printer connected to your Apple, you can get a hard copy of your music file by entering H from the COMMAND? mode. You may have to edit lines 1210 and 1220 to accommodate different printers.

## Hints for Easier Use

**Saving.** One good idea is to save two copies of your music file to the disk. One copy should be done in the FUTURE ADDITION? mode so you can edit or add to it at a later date. If you wish, the other copy can be done in the *create program,* or FUTURE ADDITION? N mode. Always remember to use a different filename.

**Tempo.** When you enter your durations, remember that if your quarter note has a value of 50, your half note will have a value of 100 and so on. You should decide what duration you want a certain note to have and work from there. Rests are accomplished by using a note value of one.

**Limits.** The number of notes you can have in one song is limited. As the program is written, the limit is 500 notes. However, it can be changed by changing the value of L in line 120.

**Notes.** The note listings on the side of the screen are especially helpful if you are transposing sheet music to disk. The numbers listed are for the Music Writer's middle octave. For the higher octave, divide the number by two; for the lower octave, multiply the number by two. For example, the note F

could be represented by the numbers 36, 72, and 144. You can also make a separate list of all the notes and their numbers. Remember, F-sharp is the same as G-flat and so on. Once again, the number zero is equivalent to the number 256.

**EXEC.** In order to use a program that you made in the FUTURE ADDITION? N mode, you must EXEC it. EXEC is a DOS command that prints a sequential text file to the screen as if it were typed from the keyboard. In this way, you can EXEC your file and run it as a BASIC program. Later on, you can save it.

Another feature is that you can load an old BASIC program and EXEC your sound routine into it. For this to work properly, however, you must have picked a starting line number for your music file that will not conflict with lines in the program to which the routine is being added.

**Insert.** If you have to type in a large amount of repetitive data, one useful trick is to enter the last note of that data and then Insert the rest. That eliminates the bother of repeatedly typing A from COMMAND? mode. Note, however, that this is useful only if you know beforehand exactly what data you want to enter.

**Keys.** There are a number of key codes that you can use with the Apple Music Writer. If at any time the screen is getting too cluttered, an ESC-SHIFT-P should do the trick. You can stop a Catalog or a play/list at any time with CTRL-S (and restart it by pressing any key).

Finally, in this program, CTRL-C RETURN can be a useful but sometimes dangerous command. I would recommend using CTRL-C only as a last resort. If for any reason you find yourself bumped out of Apple Music Writer, you can usually reenter the program, without losing any data, by typing GOTO 200.

I encourage you to experiment. Nothing can take the place of hands-on experience with a program. Just be sure you know what's going to happen at all times before you take on a big musical project.

## Apple Music Writer

```
20  REM  INITIALIZATION
30  TEXT : HOME : VTAB 1: PRINT "A=ADDNOTE E=EDIT
       P=PLAY    S=SAVE    L=LOAD    N=NORPLAY D=
    DELETE  I=INSERT  R=RESTART C=CATALOG Q=QUIT
    .=DOS H=HARD": PRINT "----------------------
    ----------------": POKE 34,5
40  VTAB 6: HTAB 34: PRINT "G =64": PRINT  TAB( 34
    )"F#=68": PRINT  TAB( 34)"F =72": PRINT  TAB(
    34)"E =76": PRINT  TAB( 34)"D#=81": PRINT  TAB(
    34)"D =86": PRINT  TAB( 34)"C#=91"
50  PRINT  TAB( 34)"C =96": PRINT  TAB( 34)"B =102
    ": PRINT  TAB( 34)"A#=108": PRINT  TAB( 34)"A
     =115": PRINT  TAB( 34)"G#=121": PRINT  TAB(
    34)"G =128": PRINT  TAB( 34)"/2 FOR": PRINT  TAB(
    34)"HIGHER": PRINT  TAB( 34)"*2 FOR": PRINT  TAB(
    34)"LOWER": POKE 33,32
60  FOR LOC = 770 TO 790: READ BYTE: POKE LOC,BYTE
    : NEXT
70  DATA  173,48,192,136,208,5,206,1,3,240,9,202,2
    08,245,174,0,3,76,2,3,96
80  HOME : INVERSE : VTAB 10: HTAB 9: PRINT "APPLE
      MUSIC WRITER": VTAB 12: HTAB 17: PRINT "BY":
      VTAB 14: HTAB 11: PRINT "BLAINE MATHIEU": NORMAL

90  FOR R = 1 TO 26: READ P,D: POKE 768,P: POKE 76
    9,D: CALL 770: NEXT R
100  DATA 172,75,162,75,152,75,144,75,108,100,1,30
    ,144,75,108,100,1,30,144,75,108,255,1,10,108,
    75,96,75,91,75,86,75,108,75,96,75,86,100
110  DATA 1,10,115,75,96,100,1,10,108,150,144,150,
    216,200,
120  HOME :L = 500: DIM N(L),D(L),N$(L),D$(L),NN(L
    ),ND(L)
130  REM  MAIN ROUTINES START
140  VTAB 5: GOTO 190
150 I = I + 1
160  PRINT : INVERSE : PRINT "NOTE#"I: NORMAL : INPUT
    "NOTE,DURATION ";N$(I),D$(I): IF N$(I) = "" OR
    D$(I) = "" THEN N$(I) = N$(I - 1):D$(I) = D$(
    I - 1)
170 N(I) =  VAL (N$(I)):D(I) =  VAL (D$(I)): IF N(
    I) > 255 OR N(I) < 0 OR D(I) > 255 OR D(I) <
    0 THEN 160
180  POKE 768,N(I): POKE 769,D(I): CALL 770
190  ONERR  GOTO 370
200  PRINT : INPUT "COMMAND? ";A$
```

```
210  IF A$ = "A" AND I = L THEN  PRINT "YOU ARE AT
      YOUR LIMIT!!!": GOTO 200
220  IF A$ = "A" THEN 150
230  IF I <  = 0 AND (A$ = "E" OR A$ = "P" OR A$ =
     "H" OR A$ = "N" OR A$ = "I" OR A$ = "S") THEN
      PRINT "SORRY, NO NOTES":I = 0: GOTO 190
240  IF A$ = "Q" THEN 450
250  IF A$ = "E" THEN 470
260  IF A$ = "P" THEN 390
270  IF A$ = "S" THEN 530
280  IF A$ = "D" THEN 1410
290  IF A$ = "L" THEN 990
300  IF A$ = "R" THEN I = 0
310  IF A$ = "C" THEN  PRINT  CHR$ (4)"CATALOG"
320  IF  LEFT$ (A$,1) = "." THEN 1120
330  IF A$ = "H" THEN 1160
340  IF A$ = "N" THEN 1250
350  IF A$ = "I" THEN 1310
360  GOTO 190
370  PRINT "ERROR#" PEEK (222): GOTO 190
380  REM  PLAY ROUTINE
390  PRINT : INPUT "STARTING NOTE (D=1): ";SN$:SN =
      VAL (SN$): IF SN$ = "" THEN SN = 1
400  PRINT : INPUT "ENDING NOTE (D=LAST): ";EN$:EN
      =  VAL (EN$): IF EN$ = "" THEN EN = I
410  IF SN < 1 OR SN > I OR EN < 1 OR EN > I THEN
     390
420  PRINT : INVERSE : PRINT "START OF SONG": PRINT
     : NORMAL : FOR X = SN TO EN: POKE 768,N(X): POKE
     769,D(X): PRINT "NOTE#";X;: HTAB 10: PRINT "N
     OTE=";N(X);: HTAB 19: PRINT "DURATION=";D(X):
      CALL 770: NEXT X
430  INVERSE : PRINT : PRINT "END OF SONG": NORMAL

440  GOTO 190
450  TEXT : HOME : PRINT "GOODBYE": END
460  REM  EDIT ROUTINE
470  INPUT "EDIT NOTE# ";NN: IF NN > I OR NN < 1 THEN
     470
480  PRINT : INVERSE : PRINT "NOTE#"NN;: NORMAL : PRINT
     " OLD: NOTE="N(NN);" DUR="D(NN)"
490  INPUT "NOTE,DURATION: ";N$(NN),D$(NN):N(NN) =
      VAL (N$(NN)):D(NN) =  VAL (D$(NN)): IF N(NN)
      > 255 OR N(NN) < 0 OR D(NN) > 255 OR D(NN) <
     0 THEN 480
500  POKE 768,N(NN): POKE 769,D(NN): CALL 770
510  GOTO 190
520  REM  SAVE ROUTINE
530  ONERR  GOTO 860
```

```
540  HOME : INPUT "FILENAME? ";FI$: IF FI$ = "" THEN
     540
550  PRINT : INPUT "STARTING NOTE NUMBER: ";SN: IF
     SN < 1 OR SN > I THEN 550
560  PRINT : INPUT "ENDING NOTE NUMBER: ";EN: IF E
     N > I OR EN < 1 THEN 560
570  PRINT : INPUT "FOR FUTURE ADDITION? ";A$: IF
     A$ <  > "N" AND A$ <  > "Y" THEN 570
580  IF A$ = "Y" THEN  POKE 216,0:F2 = 1: GOTO 640

590 F2 = 0
600  PRINT : INPUT "STARTING LINENUMBER: ";SL: IF
     SL > 63900 OR SL < 0 THEN 600
610  PRINT : INPUT "FULL LOADER PROGRAM? ";A$:A$ =
      LEFT$ (A$,1): IF A$ <  > "Y" AND A$ <  > "N"
      THEN 610
620  IF A$ = "Y" THEN FL = 1
630  IF A$ = "N" THEN FL = 0
640  PRINT : INPUT "IS EVERYTHING OK? ";A$: IF  LEFT$
     (A$,1) = "Y" AND F2 = 1 THEN 880
650  IF  LEFT$ (A$,1) = "Y" AND F2 <  > 1 THEN 670

660  GOTO 190
670 D$ =  CHR$ (4): PRINT D$"OPEN"FI$
680  PRINT D$"DELETE"FI$
690  PRINT D$"OPEN"FI$
700  PRINT D$"WRITE"FI$
710  IF FL <  > 1 THEN  GOTO 740
720  PRINT SL;"FORLOC=770TO790:READBYTE:POKELOC,BY
     TE:NEXT":SL = SL + 2
730  PRINT SL;"DATA173,48,192,136,208,5,206,1,3,24
     0,9,202,208,245,174,0,3,76,2,3,96":SL = SL +
     2
740  PRINT SL;"FORR=1TO";EN - SN + 1;":READP,D:POK
     E768,P:POKE769,D:CALL770:NEXTR":SL = SL + 2
750  FOR Z = SN TO EN
760 N = N + 1: IF N = 20 THEN N = 1
770  IF N <  > 1 THEN 810
780  PRINT
790  PRINT SL;"DATA";
800 SL = SL + 2
810  PRINT N(Z);",";D(Z);: IF N <  > 19 THEN  PRINT
     ",";
820  NEXT Z
830  PRINT
840  PRINT D$"CLOSE"
850  GOTO 190
860  PRINT : PRINT  CHR$ (7);"ERROR#"; PEEK (222):
      PRINT D$"CLOSE": GOTO 190
```

```
870  REM  2ND SAVE ROUTINE
880  ONERR  GOTO 980
890 D$ =  CHR$ (4): PRINT D$"OPEN"FI$
900  PRINT D$"DELETE"FI$
910  PRINT D$"OPEN"FI$
920  PRINT D$"WRITE"FI$
930  FOR S = SN TO EN
940  PRINT N(S): PRINT D(S)
950  NEXT S
960  PRINT D$"CLOSE"
970  GOTO 190
980  REM  LOAD ROUTINE
990  ONERR  GOTO 1090
1000  INPUT "YOU WILL LOSE YOUR DATA, OK? ";OK$:OK
     $ =  LEFT$ (OK$,1): IF OK$ <  > "Y" AND OK$ <
      > "N" THEN 1000
1010  IF OK$ = "N" THEN  POKE 216,0: GOTO 190
1020  PRINT : INPUT "FILENAME: ";FI$: IF FI$ = "" THEN
     1020
1030 D$ =  CHR$ (4): PRINT D$"VERIFY"FI$: PRINT D$
     "OPEN"FI$
1040  PRINT D$"READ"FI$
1050  FOR Z = 1 TO L
1060  INPUT N(Z): INPUT D(Z)
1070  IF N(Z) <  = 255 AND D(Z) <  = 255 THEN  NEXT
     Z: POKE 216,0: PRINT D$"CLOSE":I = Z - 1: GOTO
     190
1080  PRINT : PRINT "INCOMPATIBLE FILE!!!": PRINT
     D$"CLOSE": POKE 216,0: GOTO 190
1090  PRINT D$"CLOSE": IF  PEEK (222) = 5 THEN  POKE
     216,0:I = Z - 1: GOTO 190
1100  PRINT : PRINT "ERROR#"; PEEK (222): PRINT D$
     "CLOSE": GOTO 190
1110  REM  HANDLE DOS COMMANDS
1120  ONERR  GOTO 1140
1130 DC$ =  RIGHT$ (A$, LEN (A$) - 1): PRINT  CHR$
     (4);DC$: POKE 216,0: GOTO 190
1140  PRINT "ERROR#" PEEK (222): PRINT  CHR$ (4)"C
     LOSE": POKE 216,0: GOTO 190
1150  REM  PRINTER ROUTINE
1160  PRINT : INPUT "PRINTER READY? ";A$: IF A$ <
      > "Y" AND A$ <  > "N" THEN 1160
1170  IF A$ = "N" THEN 200
1180  PRINT : INPUT "STARTING NOTE TO BE PRINTED -
     -  DEFAULT=1: ";ST$: IF ST$ = "" THEN ST$ = "
     1"
1190  PRINT : INPUT "ENDING NOTE TO BE PRINTED --
       DEFAULT=ALL: ";EN$: IF EN$ = "" THEN EN$ =
      STR$ (I)
```

```
1200 ST =  VAL (ST$):EN =  VAL (EN$): IF ST < 1 OR
     ST > I OR EN < 1 OR EN > I OR EN < ST THEN 11
     80
1210  PRINT : INPUT "NAME OF SONG: ";FI$: IF FI$ =
     "" THEN 1210
1220  PR# 1: PRINT : PRINT FI$: PRINT : FOR X = ST
      TO EN: PRINT "NOTE#";X;: HTAB 10: PRINT "NOT
     E=";N(X);: HTAB 19: PRINT "DURATION=";D(X): NEXT
     X
1230  PRINT : PRINT "END OF SONG": PR# 0: GOTO 190

1240  REM  NORMAL PLAY ROUTINE
1250  PRINT : INPUT "STARTING NOTE (D=1): ";SN$:SN
      =  VAL (SN$): IF SN$ = "" THEN SN = 1
1260  PRINT : INPUT "ENDING NOTE (D=LAST): ";EN$:E
     N =  VAL (EN$): IF EN$ = "" THEN EN = I
1270  IF SN < 1 OR SN > I OR EN < 1 OR EN > I THEN
     1250
1280  FOR Z = SN TO EN: POKE 768,N(Z): POKE 769,D(
     Z): CALL 770: NEXT Z
1290  GOTO 190
1300  REM  INSERT ROUTINE
1310  POKE 216,0: PRINT : INPUT "INSERT BEFORE WHA
     T NOTE? ";IB: IF IB < 1 OR IB > I THEN 1310
1320  PRINT : INPUT "HOW MANY NOTES TO INSERT? ";H
     M: IF HM > L - I OR HM < 1 THEN 1320
1330  FOR Z = IB TO IB + HM - 1
1340  PRINT : INVERSE : PRINT "NOTE#"Z: NORMAL : INPUT
     "NOTE,DURATION: ";NN(Z),ND(Z): IF NN(Z) < 0 OR
     NN(Z) > 255 OR ND(Z) < 0 OR ND(Z) > 255 THEN
     1340
1350  POKE 768,NN(Z): POKE 769,ND(Z): CALL 770
1360  NEXT Z
1370  FOR Z = I TO IB STEP  - 1:N(Z + HM) = N(Z):D
     (Z + HM) = D(Z): NEXT Z
1380  FOR Z = IB TO IB + HM - 1:N(Z) = NN(Z):D(Z) =
     ND(Z): NEXT Z
1390 I = I + HM
1400  GOTO 190
1410  REM  DELETE ROUTINE
1420  PRINT : INPUT "DELETE FROM NOTE (D=LAST): ";
     DF$: IF DF$ = "" THEN I = I - 1: IF I =  - 1 THEN
     I = 0: GOTO 190
1430  IF DF$ = "" THEN 190
1440  PRINT : INPUT "TO NOTE: ";DT$:DF =  VAL (DF$
     ):DT =  VAL (DT$): IF DT < 1 OR DT > I OR DF <
     1 OR DF > I OR DF > DT THEN 1420
1450  FOR Z = DT + 1 TO I:N(Z - (DT - DF + 1)) = N
     (Z):D(Z - (DT - DF + 1)) = D(Z): NEXT Z
1460 I = I - (DT - DF + 1): GOTO 190
```

# Apple Shape Generator

J. F. Johnson

*Applesoft allows shapes to be manipulated from within a BASIC program, but the process of creating shapes and entering them into a shape table can be tedious and error-prone. This program simplifies the process, automatically performing all required binary-to-hex conversions and creating a shape table.*

Many of the shape-drawing routines currently available for the Apple allow a shape to be created within a given rectangular drawing area. Such techniques are fine for creating relatively small shapes. However, as the size of the shape increases, the amount of wasted space (that is, the number of bytes which are "off" and represent only the background) becomes considerable.

This program creates shapes using an approach explained in the Applesoft manual (Chapter 9). The head-to-tail vector method is used to initially define the shape. These vectors are then "unwrapped" and sequentially combined in pairs for conversion from individual binary codes into equivalent hexadecimal codes. Each hexadecimal byte represents one byte in the shape definition.

The shape is then added to a table in memory, and the table's index is updated. Shapes which would otherwise require as much as 8K thus need less than 1K.

You can do a number of things with the shape generator:

- Construct a shape table comprised of 1–255 shapes.
- Create a table with a maximum length of 6K.
- Alter any shape after it has been entered into the table, or add "buffer bytes" at the end of a shape definition so that it can be slightly enlarged relative to its original definition.
- Correct mistakes which occur while entering vectors during a shape definition.
- View all the shapes in the current table.
- Display any particular shape, with the effect of ROT and SCALE variations (using the game paddles) immediately displayed on the hi-res screen.
- BSAVE and then BLOAD and modify any shape table. Existing shapes can be changed, or new ones can be added

(assuming the table does not contain the maximum number of shapes originally designated).
- Erase existing shape tables to create new tables or load existing ones.

## Use an EXEC File to Initialize

Shape Generator is written in Applesoft. Program 2 creates a text file (named "Key Shape Loader") which reassigns the beginning-of-program pointer (104, 103) and then runs the program.

By EXECing the text file "Key Shape Loader," the required POKEs are completed and "Key Shape Maker," Program 1, runs automatically. Be sure to save Program 1 with the filename "Key Shape Maker."

The program is loaded at $6001 (24577), just above the second hi-res page of graphics. The second hi-res page is used for the temporary storage of vectors that define the current shape. These vectors are then paired and converted into their equivalent hexadecimal code, with the resulting hex code defining the shape stored on the second hi-res page. If the shape is to be saved, the hex code is then transferred to the shape table. The creation and display of all shapes utilize the first hi-res page. The shape table is stored at $800 (2048), and its length may not exceed $2000 (8196) since the first hi-res page is used for display purposes.

## Execution

The user is initially prompted for the number of shapes (1–255) that will be entered into the table. Since extra shapes are invariably required at some future date, it is always better to enter a number larger than what is currently estimated. Since the table need not be completed at one setting, the partially constructed table can be BSAVEd and then BLOADed at a future date. Additional shapes can then be added (up to the original number specified) or current shapes can be redefined.

The maximum number of shapes is then POKEd into $801. Room for the shape table index (starting at $802) is then allocated. The index stores the locations of all shapes relative to the start of the table ($800). The index must contain two bytes for each stored shape. If the estimated number of shapes to be stored is later found to be too low, you'll be out of luck. Location $800 initially contains a value of zero; it's incremented by one each time a shape is added to the table.

The shapes are created using two sets of four keys. Plotting vectors are entered using the I, K, M, and J keys, while nonplotting vectors are entered using the E, D, X, and S keys. Both sets of keys are arranged on the keyboard in a north-east-south-west fashion. You can use your right-hand set for plotting and your left-hand set for nonplotting. The back arrow key (←) may be used to sequentially erase vectors starting with the last one entered, and is very useful for correcting any mistakes. The keystroke ! (Shift-1) terminates shape definition.

When drawing begins, a single-dot cursor is positioned on the first hi-res screen (it marks the point at which drawing will begin) and the shape is then displayed as it is constructed.

Any nonplotting vectors which cross any existing outline of the shape will result in the boundary being erased where the crossover occurs. However, when the final shape is displayed for verification, it will show the contiguous boundary that was originally constructed.

Also displayed during construction are the current X and Y coordinates of the cursor, the three-digit binary code of each vector as it is entered, and the maximum number of bytes which may be used to define the present shape.

When the definition of the shape is ended (by pressing !, or Shift-1), the keystroke vectors are converted to hexadecimal code and the resulting shape is displayed once more before being stored. If you decide to save it, it will be appended to the current table. The corresponding index locations will be updated, and location $800 will be incremented by one. If the shape is not saved, the defining of additional shapes simply continues.

## Shape Table Commands

Several subroutines allow you to experiment with various shape table commands. That makes it much easier to explore the capabilities (as well as the limitations) of shapes within Applesoft and may facilitate inclusion of shape tables within programs. To use these subroutines, call item 1 (DISPLAY SHAPES IN CURRENT TABLE) from the main menu; then call item 2 (VIEW ONLY ONE SHAPE) from the subsequent menu and follow the prompts.

The SCALE command allows the expansion of a defined shape. Since the original shape was constructed using the

smallest SCALE value, figures can only be expanded using this command. You'll soon discover, however, that the contiguous boundary of a shape may become segmented when its size is enlarged through SCALEing; in fact, the shape can quickly become unrecognizable. That problem can usually be overcome by redefining the same shape boundary using a different sequence of plotting/nonplotting vectors. The ability to redefine any given shape will allow the user to experiment.

Rotations in the plane of the screen are controlled by the ROT command. An inverse relationship exists between the number of unique rotational values defined by the ROT command and the SCALE command. Increasing ROT from 0 to 64 will rotate it 360 degrees about the origin. As the value for SCALE increases from 0, more unique rotational values are recognized between the ROT values of 0 and 64; thus, the incremental rotational angle decreases. By making the original shape very small, and then expanding it using the SCALE command, a smaller angle of rotation can be realized between the ROT value of 0 and 64. The values for both of the commands may be varied for a chosen shape, with the effects on the shape displayed on the screen.

## Using the Shapes

To display the shapes from Applesoft, use either DRAW or XDRAW. XDRAW complements the current color of the shape at its present location and is very convenient for displaying and erasing shapes. DRAW requires that HCOLOR be changed from a value of 3 to 0 if the shape is to be drawn and then erased. These commands may also display the same shape differently. If any nonplotting vectors cross the boundary of plotting vectors in the original shape definition, then DRAW (HCOLOR=3) will display a contiguous shape.

XDRAW, however, effectively erases any regions of plotting/nonplotting vector overlap. This should be taken into consideration when defining shape boundaries. The shape display for verification purposes (prior to appending the shape to the current table) is displayed using DRAW(HCOLOR=3). During viewing of a shape with ROT and SCALE variations, the shape is drawn and erased using XDRAW.

Key Shape Maker creates a shape table starting at $800 (2048) in RAM. It may be BLOADed into another region if there exists a conflict with the storage of the controlling

Applesoft program, or one with a machine language program which must occupy this region.

There are two DOS entry points which store both the starting address and length of a BLOADed file. Since the user specifies the starting address of a binary file, only the length must be determined. This is accomplished in the following manner.

After BSAVEing your shape table to disk, BLOAD it back into memory. This may be done in direct mode or under Key Shape Maker control. If the shape table has been loaded by an Applesoft program, press the reset button. Then enter the following as a direct execution instruction:

**PRINT PEEK(43616)+PEEK(43617)*256<ret>**

The base ten number that appears on the screen immediately will be the length of the shape table (see Appendix E of the DOS manual, "DOS Entry Points And Schematics"). With that information, you have some flexibility in BLOADing the shape table into various regions of RAM. For example, a shape table of byte length 350 may be BLOADed at location 24577 (immediately above the second hi-resolution page) with the following Applesoft instruction:

**100 PRINT CHR$(4)"BLOAD SHAPE TABLE-1, A24577,L350"**

Finally, you must supply the location of the shape table. The pointer designating the beginning of the current shape table is located on the zero page of memory and consists of the locations $E8 (232) and $E9 (233). The integer value obtained by dividing the starting address by 256 is POKEd into 233, with the remainder POKEd into 232. For example, 24577/256=96, with a remainder of 1, and would be POKEd as shown:

**110 POKE 233,96: POKE 232,1**

Your Applesoft program will then be able to use the shape table currently residing in RAM.

## Program 1. Key Shape Maker

```
60  REM    TS=START OF SHAPE TABLE///VC=MARKER USE
    D IN DISPLAY OF 6 DIGITS REPRESENTING 2 VECTO
    RS///VS=MARKER FOR START OF TEMPORARY STORAGE
     FOR VECTOR TABLE AND ENSUING TEMPORARY STORA
    GE DERIVED SHAPE///16395=START OF TEMPORARY S
    HAPE TABLE
```

```
70 A$ = "PRESS ! TO STOP DRAWING SHAPE."
80 TS = 2048: POKE TS,0:VC = 16389:VS = 16396:LI =
     2050:MI = 2051:D$ =  CHR$ (4): GOTO 4000
100  HCOLOR= 3: HPLOT X,Y: FOR J = 1 TO 20: NEXT J
     : HCOLOR= 0: HPLOT X,Y:X =  PDL (0) / .913:Y =
      PDL (1) / 1.6: IF  PEEK ( - 16287) > 127 OR
      PEEK ( - 16286) > 127 THEN  RETURN
105  GOTO 100
110 S1 =  INT (1 +  PDL (0) * ( PEEK (TS) - 1) / 2
     40): ROT= 0: HCOLOR= 3: SCALE= 1: RETURN
115 S2 =  INT (1 +  PDL (0) * ( PEEK (TS) - 1) / 2
     40): RETURN
120  XDRAW S1 AT X,Y: VTAB 24: HTAB 1: CALL  - 868
     : PRINT "SHAPE #"S1".";
125  GOSUB 115: IF  PEEK ( - 16287) > 127 THEN  RETURN

130  IF S2 <  > S1 THEN  XDRAW S1 AT X,Y:S1 = S2: GOTO
     120
135  GOTO 125
140  GOSUB 110
145  VTAB 5: HTAB 1: CALL  - 868: PRINT "SHAPE #"S
     1"."
150  GOSUB 115: IF S2 <  > S1 THEN S1 = S2: GOTO 1
     45
152  IF  PEEK ( - 16287) > 127 THEN  RETURN
154  GOTO 150
158 S1 =  INT ( PDL (1) * 7 / 240): RETURN
159 S2 =  INT ( PDL (1) * 7 / 240): RETURN
160  GOSUB 158
162  VTAB 10: HTAB 1: CALL  - 868: PRINT "HCOLOR="
     S1"."
164  GOSUB 159: IF S2 <  > S1 THEN S1 = S2: GOTO 1
     62
166  IF  PEEK ( - 16286) > 127 THEN  RETURN
168  GOTO 164
170  GOTO 166
172 R1 =  PDL (0) / 3:S1 =  PDL (1) / 3: RETURN
173 R2 =  PDL (0) / 3:S2 =  PDL (1) / 3: RETURN
174  GOSUB 172
175  HCOLOR= HC: ROT= R1: SCALE= S1: DRAW SH AT XI
     ,YI: VTAB 24: HTAB 1: CALL  - 868: PRINT "ROT
     =" INT (R1) SPC( 8)"SCALE=" INT (S1);
176  GOSUB 173: IF R2 <  > R1 OR S2 <  > S1 THEN R
     1 = R2:S1 = S2: CALL 62450: GOTO 175
177  IF  PEEK ( - 16287) > 127 OR  PEEK ( - 16286)
      > 127 THEN  RETURN
178  GOTO 176
200  POKE TS + 1, VAL (NS$): RETURN : REM  MAXIMUM
      NUMBER OF SHAPES THAT CAN BE ENTERED INTO TH
     IS TABLE
```

```
203 PA = 256 *  PEEK (MI) +  PEEK (LI) + TS: RETURN

205 PA = TS + 4 + 2 *  VAL (NS$): RETURN : REM
    IS LOCATION IN TABLE WHERE FIRST SHAPE WILL B
    E SAVED
210 LS = TS + 2 * SH:MS = TS + 1 + 2 * SH:DD = 256
     * ( PEEK (MS + 2) -  PEEK (MS)) + ( PEEK (LS
    + 2) -  PEEK (LS)): RETURN
212 LI = TS + 2 * ( PEEK (TS) + 1):MI = LI + 1: RETURN
    : REM   INIT INDEX FOR TABLE THAT HAS BEEN LO
    ADED
215 LI = LI + 2:MI = MI + 2: RETURN : REM   INCREM
    ENT INDEX LOCATION FOR NEXT SHAPE
220 LI = LI - 2:MI = MI - 2: RETURN : REM   DECREM
    ENT INDEX LOCATION FOR FIRST SHAPE TO BE DRAW
    N IN LOADED OR ALTERED TABLE
225 IP = VS: RETURN : REM  INITIALIZE LOCATION WHE
    RE PLOTTED VECTORS ARE STORED TEMPORARILY UNT
    IL THEY ARE CONVERTED INTO A SHAPE
230 N = VS + 1:SL = VS + 1: RETURN : REM  INITIALI
    ZE TWO COUNTERS WHICH ARE USED DURING THE CON
    VERSION OF STORED VECTORS INTO A SHAPE
235 PA = TS + 256 *  PEEK (MS) +  PEEK (LS): RETURN
    : REM  LOCATION IN TABLE OF START OF NEXT SHA
    PE
240  POKE LI, INT ((((PA - TS) / 256) -  INT ((PA -
    TS) / 256)) * 256 + .5): POKE MI, INT ((PA -
    TS) / 256): RETURN : REM    POKE STARTING LOC
    ATION FOR GIVEN SHAPE IN APPROPRIATE INDEX LO
    CATION
250 A = 0:B = 0:C = 0: RETURN : REM   INITIALIZE A
    ,B,C TO ZERO
255 L = IP - VS:K =  INT (L / 2) +  INT ((L / 2 -
     INT (L / 2)) * 2 + .05): RETURN : REM  L=#BY
    TES CONTAINING VECTORS///K=#BYTES REQUIRED TO
     STORE SHAPE;1 SHAPE BYTE PER 2 VECTOR BYTES
260  POKE 233,64: POKE 232,9: POKE 16393,1: POKE 1
    6395,4: POKE 16396,0: RETURN : REM   DEFINE U
    NIT SHAPE TABLE WHERE TEMPORARILY DEFINED SHA
    PE EXISTS
265  POKE 233,8: POKE 232,0: RETURN : REM  LOCATIO
    N OF SHAPE TABLE
270 RS =  PEEK (TS + 1) -  PEEK (TS): RETURN : REM
      RS=# OF SHAPES THAT MAY STILL BE ENTERED IN
    TO SHAPE TABLE
299  REM  PLOT/ERASE POINT AT CURRENT X,Y UNTIL KE
    Y PRESS OCCURS.
```

```
300 XO = X:YO = Y: HCOLOR= 3: HPLOT XO,YO: FOR J =
    1 TO 20: NEXT J: HCOLOR= 0: HPLOT XO,YO: FOR
    J = 1 TO 20: NEXT J: IF  PEEK ( - 16384) < 12
    8 THEN 300
310  HCOLOR= 3: POKE  - 16368,0:Z =  PEEK ( - 1638
    4): RETURN
324  REM   PLOT PRESENT POINT IF ENTERED VECTOR IS
     A PLOT-THEN-MOVE VECTOR
325  HCOLOR= 3: HPLOT XO,YO: RETURN
329  REM  ERASE PREVIOUS POINT PLOTTED
330  HCOLOR= 0: HPLOT XO,YO: RETURN
349  REM  EVALUATE KEY PRESS IN TERMS OF NEW X,Y C
    OORDINATES.
350 F1 = 0
352  IF Z = 73 OR Z = 69 THEN Y = Y - 1: GOSUB 362
    : RETURN : REM  MOVE UP
354  IF Z = 75 OR Z = 68 THEN X = X + 1: GOSUB 364
    : RETURN : REM   MOVE RIGHT
356  IF Z = 77 OR Z = 88 THEN Y = Y + 1: GOSUB 366
    : RETURN : REM  MOVE DOWN
358  IF Z = 74 OR Z = 83 THEN X = X - 1: GOSUB 368
    : RETURN : REM   MOVE LEFT
360 F1 = 1: RETURN : REM   FLAG F1 SET TRUE IF NO
    U,R,D,L MOVE
362  IF Y < 0 THEN Y = 0:F1 = 1
363  RETURN
364  IF X > 279 THEN X = 279:F1 = 1
365  RETURN
366  IF Y > 159 THEN Y = 159:F1 = 1
367  RETURN
368  IF X < 0 THEN X = 0:F1 = 1
369  RETURN
399  REM  EVALUATE 3 DIGIT BINARY EQUIVALENT OF IN
    DIVIDUAL VECTOR
400 F1 = 0: IF Z = 73 THEN A = 1:B = 0:C = 0: RETURN

402  IF Z = 75 THEN A = 1:B = 0:C = 1: RETURN
404  IF Z = 77 THEN A = 1:B = 1:C = 0: RETURN
406  IF Z = 74 THEN A = 1:B = 1:C = 1: RETURN
408  IF Z = 69 THEN A = 0:B = 0:C = 0: RETURN
410  IF Z = 68 THEN A = 0:B = 0:C = 1: RETURN
412  IF Z = 88 THEN A = 0:B = 1:C = 0: RETURN
414  IF Z = 83 THEN A = 0:B = 1:C = 1: RETURN
418 F1 = 1: RETURN
424  REM  PRINT PRESENT COORDINATES OF X,Y
425  VTAB 21: HTAB 1: CALL  - 868: PRINT "X="X,"Y=
    "Y: RETURN
```

```
449  REM  ERASE CURRENT POINT AND MOVE BACK ONE PO
     INT
450 PP =  PEEK (IP): IF IP = VS THEN  RETURN : REM
      CAN'T ERASE PAST ORIGIN OF SHAPE
455  IF PP = 0 OR PP = 4 THEN Y = Y + 1: GOSUB 475
     : RETURN
460  IF PP = 1 OR PP = 5 THEN X = X - 1: GOSUB 475
     : RETURN
465  IF PP = 2 OR PP = 6 THEN Y = Y - 1: GOSUB 475
     : RETURN
470  IF PP = 3 OR PP = 7 THEN X = X + 1: GOSUB 475
     : RETURN
475 XO = X:YO = Y: GOSUB 330: POKE IP,0:IP = IP -
     1: RETURN
499  REM  POKE VECTOR INTO RAM LOCATION IP
500 IP = IP + 1: POKE IP,4 * A + 2 * B + C: RETURN

509  REM  POKE BINARY EQUIVALENT OF VECTOR MOVE
510 P(1 + I * 3) = A:P(2 + I * 3) = B:P(3 + I * 3)
      = C
515  IF I = 1 THEN  FOR J = 0 TO 5: POKE 1872 + J,
     48: NEXT J: FOR J = 0 TO 2: POKE 1875 + J,P(4
      + J) + 48: NEXT J: RETURN
520  FOR J = 0 TO 2: POKE 1872 + J,P(1 + J) + 48: NEXT
     J: RETURN
525  FOR J = 1 TO 6:P(J) = 0: NEXT J: RETURN
600  HGR2 : HGR : SCALE= 1: ROT= 0: HCOLOR= 3:XX =
     139:YY = 80:X = XX:Y = YY: RETURN : REM    HI
     -RES INITIALIZATION
700 BL = 8190 - PA:DI = 24576 - 16396:VL = DI: RETURN
     : REM  NEW TABLE BYTE LIMITS
710 NS =  PEEK (2048):LI = TS + 2 * (NS + 1):MI =
     TS + 1 + 2 * (NS + 1):PA = TS + 256 *  PEEK (
     MI) +  PEEK (LI)
720 BL = 8190 - PA: IF DI < 2 * (8190 - PA) THEN V
     L = DI: RETURN
730 VL = 2 * (8190 - PA): RETURN
765 F1 - 0: IF VL < 100 THEN F1 = 1
767  RETURN
770 F2 = 0: VTAB 21: PRINT "THERE ARE "8190 - PA"
     BYTES REMAINING FOR MORE": PRINT "SHAPES IN C
     URRENT TABLE IF YOU HAVE NOT CONSTRUCTED THE
     LAST SHAPE."
775  IF 8190 - PA < 100 THEN  PRINT "NO MORE SHAPE
     S MAY BE ADDED TO CURRENT  TABLE.":F2 = 1
780  RETURN
800 F3 = 0:VL = VL - 1: VTAB 21: HTAB 33: CALL  -
     868: PRINT VL
```

```
805  IF VL < 200 THEN  VTAB 22: HTAB 1: PRINT "ONL
     Y "VL - 190" MOVES LEFT.";: FOR J = 1 TO 1000
     : NEXT J: HTAB 1: CALL  - 868: IF VL <  = 191
      THEN F3 = 1
810  RETURN
975  VTAB 24: HTAB 5: CALL  - 958: PRINT "PRESS AN
     Y LETTER TO CONTINUE.";: GET Z$:J =  FRE (0):
      RETURN
999  REM  INITIALIZE SHAPE TABLE PARAMETERS
1000  TEXT : HOME : PRINT  TAB( 5);"THE NUMBER OF
     SHAPES THAT MAY BE    ENTERED IN A SHAPE TABL
     E IS IN THE RANGEOF 1-255.  IT IS ALWAYS BEST
      TO ALLOW   EXTRA ROOM FOR ADDITIONAL SHAPES
     YOU MAYWISH TO INCLUDE IN THE FUTURE."
1010  INPUT "      ENTER A NUMBER BETWEEN 1 AND 255
     ,  THEN PRESS RETURN.";NS$: IF  VAL (NS$) < 1
      OR  VAL (NS$) > 255 THEN 1000
1020  GOSUB 200: REM   POKE MAX # OF SHAPES THAT C
     AN BE ENTERED INTO THIS TABLE
1030  GOSUB 205: REM  INITIAL RAM LOCATION FOR FIR
     ST SHAPE
1040  GOSUB 240: REM START OF FIRST SHAPE
1050  GOSUB 700: REM  BL,DI,VL
1055  RETURN
1060  GOSUB 250: REM  INITIALIZE COMPONENTS OF VEC
     TOR MOVE
1070  GOSUB 600: REM  HI-RES INIT
1080  RETURN
1200  TEXT : HOME : PRINT  TAB( 5);"BEFORE ACTUALL
     Y DRAWING A SHAPE,    THE BLINKING DOT MAY BE
      MOVED TO ANY    POSITION ON THE SCREEN.  USE
      THE E,S,D  AND X KEYS FOR DOT POSITIONING ON
     LY."
1205  PRINT "PRESS ! WHEN READY TO DRAW A SHAPE."
1210  PRINT  TAB( 5);"THE SHAPE YOU ARE TO DRAW MA
     Y THEN  BE COMPRISED OF PLOTTING AS WELL AS N
     ONPLOTTING VECTORS.  USE THE E,S,D AND X   KE
     YS FOR NONPLOTTING VECTORS, AND THE   I,J,K A
     ND M KEYS FOR PLOTTING VECTORS."
1220  PRINT "THE LEFT ARROW KEY (<-) MAY BE USED T
     O  ERASE MISTAKES, AND THE ! KEY TO TERM-  IN
     ATE THE SHAPE.": PRINT
1240  PRINT  TAB( 1);"-NONPLOTTING-"; SPC( 9);"-PL
     OTTING-"
1250  PRINT  TAB( 2);"E-MOVE UP" SPC( 7)"I-PLOT TH
     EN MOVE UP": PRINT  TAB( 2);"S-MOVE LEFT" SPC(
     5)"J-PLOT THEN MOVE LEFT"
```

```
1260  PRINT  TAB( 2);"D-MOVE RIGHT" SPC( 4)"K-PLOT
      THEN MOVE RIGHT": PRINT  TAB( 2);"X-MOVE DOW
     N" SPC( 5)"M-PLOT THEN MOVE DOWN"
1270  PRINT  TAB( 12);"<-  ERASER": PRINT  TAB( 12
     );"!   STOP"
1280  GOSUB 975
1285  HOME : VTAB 23: PRINT "PRESS ! WHEN YOU ARE
     READY TO DRAW YOUR SHAPE.": GOSUB 600: GOSUB
     425
1290  GOSUB 300: IF Z = 33 THEN XI = X:YI = Y: HOME
     : RETURN
1300  GOSUB 350: GOSUB 425: GOTO 1290
1350  GOSUB 225: GOSUB 425: GOSUB 525:I = 0: GOSUB
     515: VTAB 22: HTAB 1: PRINT A$: IF  NOT F5 THEN
      GOSUB 720: GOSUB 800: IF F3 THEN  RETURN
1360 I = 1 - I: REM  TOGGLE
1370  IF F5 THEN  HTAB 1: VTAB 24: CALL  - 868: PRINT
     "YOU HAVE UP TO "2 * DD - 3 - (IP - 16394)" M
     OVES LEFT.";: IF 2 * DD - 3 - (IP - 16394) =
     0 THEN  RETURN
1380  GOSUB 300: IF Z = 33 THEN  RETURN
1390  IF Z = 8 THEN  GOSUB 450: GOSUB 425: GOTO 13
     60: REM  ERASE LAST MOVE
1400  IF Z = 69 AND I = 0 THEN  VTAB 22: HTAB 1: PRINT
     "THIS MOVE HAS NO EFFECT ON THE SHAPE.";: FOR
     J = 1 TO 2000: NEXT J: HTAB 1: CALL  - 868: PRINT
     A$: GOTO 1370
1405  GOSUB 350: IF F1 THEN 1370: REM   EVAL KEY P
     RESS FOR NEW X,Y : SET FLAG F1 IF ILLEGAL
1410  GOSUB 400: REM  EVALUATE 3 DIGIT BINARY EQUI
      VALENT OF KEY PRESS
1420  GOSUB 500: REM  SAVE VECTOR MOVE WITH POKE
1430  GOSUB 510: REM  DISPLAY 'ACCUMULATOR' WITH T
     EXT POKES
1440  GOSUB 425: REM   PRINT NEW X,Y COORDS TO SCR
     EEN
1450  IF Z > 72 AND Z < 78 THEN  GOSUB 325: REM
     PLOT POINT ON HI-RES FOR APPROPRIATE PLOTTING
      VECTOR
1455  IF  NOT F5 THEN  GOSUB 800: IF F3 THEN  RETURN

1460  GOTO 1360
1500  GOSUB 230
1510  GOSUB 255
1520  FOR J = 1 TO K: POKE N,( PEEK (SL) + 8 *  PEEK
     (SL + 1)):SL = SL + 2:N = N + 1: NEXT J: POKE
     N,0: REM  POKE SHAPE 'ON TOP OF' VECTORS
1530  GOSUB 260: HGR : HCOLOR= 3: DRAW 1 AT XI,YI:
      GOSUB 265
```

```
1540  HOME : VTAB 21: PRINT "DO YOU WISH TO SAVE T
      HIS SHAPE (Y/N)?";: GET Z$: IF Z$ <  > "Y" AND
      Z$ <  > "N" THEN 1540
1545  IF F5 THEN  RETURN
1550  IF Z$ = "N" THEN  RETURN
1560  FOR J = N + 1 TO N + 2 + .25 * (N - VS + 1):
       POKE J,0: NEXT J: REM   EXPAND SHAPE 25% BY
      ADDING ZEROS AT END
1570 N = J - 1: FOR J = VS + 1 TO N: POKE PA, PEEK
      (J):PA = PA + 1: NEXT J: REM   TRANSFER SHAPE
       FROM TEMPORARY LOCATION TO SHAPE TABLE
1580 NS =  PEEK (TS):NS = NS + 1: POKE TS,NS: REM
       INCREASE # SHAPES IN INDEX BY 1
1590  GOSUB 215: GOSUB 240: REM   POKE DATA INTO T
      HIS INDEX LOCATION///INCREMENT INDEX LOCATION
       OF NEXT SHAPE
1610  RETURN
1700  GOSUB 600: HOME : VTAB 21: PRINT "USE THE GA
      ME PADDLES TO POSITION THE DOTAT WHICH POINT
      THE SHAPE WILL BE DRAWN. PRESS EITHER BUTTON
      WHEN READY TO VIEW  SHAPES.";
1720 X = 140:Y = 80: GOSUB 100:NS =  PEEK (2048): HOME
      : VTAB 21: PRINT "USE THE X GAME PADDLE TO VI
      EW ALL SHAPESIN CURRENT TABLE.  PRESS BUTTON
      WHEN    FINISHED VIEWING.": GOSUB 110: GOSUB
      120: RETURN
1750  TEXT : HOME : PRINT  TAB( 5);"THE FOLLOWING
      SEQUENCE WILL BE       FOLLOWED IN VIEWING A S
      HAPE."
1755  PRINT : PRINT "1) INPUT SHAPE NO. USING X GA
      ME PADDLE.": PRINT : PRINT "2) INPUT HCOLOR U
      SING Y GAME PADDLE.": PRINT : PRINT "3) MOVE
      SHAPE TO DESIRED POSITION.": PRINT : PRINT "4
      ) USE X PADDLE TO VARY ROT, AND            Y P
      ADDLE TO VARY SCALE.": GOSUB 975
1760  HOME : PRINT  TAB( 5);"USE THE X GAME PADDLE
       TO CHOOSE YOURSHAPE NO.  PRESS THE PADDLE'S
      BUTTON    WHEN FINISHED.": GOSUB 140:SH = S1
1765  PRINT : PRINT  TAB( 5);"INPUT THE HCOLOR USI
      NG THE Y PADDLE.PRESS ITS BUTTON WHEN FINISHE
      D.": GOSUB 160:HC = S1
1770  HGR : HOME : VTAB 21: PRINT  TAB( 5);"USE TH
      E GAME PADDLES TO LOCATE THE  POINT WHERE THE
       SHAPE WILL BE DRAWN.    PRESS EITHER BUTTON
      WHEN FINISHED.": GOSUB 100:XI =  INT (X):YI =
       INT (Y)
```

```
1775  HOME : VTAB 21: PRINT  TAB( 5);"USE PADDLES
      TO VARY ROTATION (X) ANDSCALE (Y).  PRESS EIT
      HER BUTTON TO STOP.": VTAB 23: PRINT "SHAPE #
      "SH SPC( 3)"HCOLOR="HC SPC( 3)"X="XI SPC( 3)"
      Y="YI: GOSUB 174: RETURN
1800  FOR J = N - VS + 1 TO DD:N = N + 1: POKE N,0
      : NEXT J
1810 N = VS + 1:J = 256 *  PEEK (MS) +  PEEK (LS) +
      TS: FOR K = 1 TO DD - 1: POKE J, PEEK (N):J =
      J + 1:N = N + 1: NEXT K: RETURN
3500  GOSUB 1000
3501  GOSUB 1060: REM  ENTRY FOR ADDING TO EXISTIN
      G TABLE
3502  HOME : GOSUB 770: GOSUB 975: IF F2 THEN  RETURN

3505  GOSUB 270: HOME : TEXT : PRINT RS" SHAPES MA
      Y BE ADDED TO THE CURRENT": PRINT "TABLE WHIC
      H CONTAINS "; PEEK (2048);" SHAPES.": GOSUB 9
      75
3510  IF  NOT RS THEN 3575
3515  HOME : PRINT  TAB( 5);"DO YOU WISH TO DRAW A
       SHAPE": PRINT "Y/N?";: GET Z$: IF Z$ <  > "Y
      " AND Z$ <  > "N" THEN 3515
3520  IF Z$ = "N" THEN 3575
3525  GOSUB 1200
3530  GOSUB 1350
3535  GOSUB 1500
3540  GOTO 3502
3575  RETURN
3650  GOSUB 8000: ONERR  GOTO 20000
3660  PRINT : PRINT D$"BLOAD"NA$",A"TS: GOSUB 270:
       GOSUB 212: GOSUB 203: GOSUB 700: POKE 216,0:
       RETURN
3670  HOME : PRINT  TAB( 5);"YOUR FILE NAME LENGTH
       IS ZERO.  DO  YOU STILL WISH TO BLOAD A SHAP
      E TABLE   FROM DISKETTE (Y/N)?": GET Z$: IF Z
      $ <  > "Y" AND Z$ <  > "N" THEN 3670
3680  IF Z$ = "N" THEN  RETURN
3690  IF Z$ = "Y" THEN 3650
4000  HOME : IF  PEEK (TS) > 0 THEN 4100
4010  PRINT  TAB( 5);"PRESS THE NUMBER OF YOUR CHO
      ICE.": PRINT : PRINT "1) DRAW SHAPES/CONSTRUC
      T A SHAPE TABLE.": PRINT "2) BLOAD A SHAPE TA
      BLE THAT HAS BEEN        CONSTRUCTED WITH THIS
       ROUTINE.": PRINT "3) QUIT."
4014  GET Z$: IF  VAL (Z$) < 1 OR  VAL (Z$) > 3 THEN
       HOME : GOTO 4010
4016  IF Z$ = "3" THEN 30000
4020  ON  VAL (Z$) GOSUB 3500,3650
```

```
4030  GOTO 4000
4100  HOME : PRINT  TAB( 5);"PRESS THE NUMBER OF Y
      OUR CHOICE.": PRINT
4105  PRINT : PRINT "1) DISPLAY SHAPES IN CURRENT
      TABLE.": PRINT : PRINT "2) ADD SHAPES TO CURR
      ENT TABLE.": PRINT : PRINT "3) CHANGE A SHAPE
       IN CURRENT TABLE."
4110  PRINT : PRINT "4) BSAVE CURRENT TABLE TO DIS
      KETTE.": PRINT : PRINT "5) DELETE TABLE CURRE
      NTLY IN MEMORY.": PRINT : PRINT "6) QUIT.": PRINT

4120  GET Z$: IF  VAL (Z$) < 1 OR  VAL (Z$) > 6 THEN
      4100
4130  IF Z$ = "6" THEN 30000
4150  HOME : ON  VAL (Z$) GOSUB 5200,5400,5600,580
      0,6000
4160  GOTO 4000
5200  TEXT : HOME : IF  PEEK (TS) = 0 THEN  PRINT
      "THERE ARE NO SHAPES IN TABLE.": GOSUB 975: RETURN

5205  GOSUB 265: PRINT  TAB( 5);"PRESS THE NUMBER
      OF YOUR CHOICE.": PRINT : PRINT "1) VIEW ALL
      SHAPES.": PRINT : PRINT "2) VIEW ONLY ONE SHA
      PE.": PRINT : PRINT "3) RETURN TO MAIN MENU."

5210  GET Z$: IF  VAL (Z$) < 1 OR  VAL (Z$) > 3 THEN
      5200
5215  IF Z$ = "3" THEN  RETURN
5220  ON  VAL (Z$) GOSUB 1700,1750: GOTO 5200
5400  GOSUB 3501: RETURN
5600  TEXT : HOME : IF  PEEK (TS) = 0 THEN  PRINT
       TAB( 5);"THERE IS NO TABLE CURRENTLY IN
       MEMORY.": GOSUB 975: RETURN
5610  PRINT  TAB( 5);"THERE ARE " PEEK (2048)" SHA
      PES IN TABLE.": INPUT "    ENTER THE NUMBER O
      F THE SHAPE YOU   WISH TO CHANGE, OR A ! TO R
      ETURN TO THE MAIN MENU.";SH$: IF SH$ = "!" THEN
      F5 = 0: RETURN
5620 SH =  VAL (SH$): IF SH < 1 OR SH >  PEEK (TS)
       THEN 5600
5630 F5 = 1: GOSUB 210: GOSUB 1200: GOSUB 1350: GOSUB
      1500
5640  IF Z$ = "Y" THEN  GOSUB 1800
5660  GOTO 5600
5800  IF  PEEK (TS) = 0 THEN  PRINT "THERE ARE NO
      SHAPES IN TABLE.": GOSUB 975: RETURN
```

```
5805  PRINT "IF YOU WISH TO SAVE THIS TABLE ON A
      DIFFERENT DISKETTE, PUT IT IN THE DRIVE AT
      THIS TIME.": PRINT : PRINT "PUT THE UTILITY
      DISKETTE BACK INTO THE  DRIVE AFTER THE DISK
      DRIVE'S RED LIGHT  GOES OFF.": GOSUB 975: GOSUB
      8000: PRINT
5810  PRINT : PRINT D$"BSAVE"NA$",A"TS",L"PA - TS:
      RETURN
5820  HOME : PRINT  TAB( 5);"YOUR FILE NAME LENGTH
      IS ZERO.  DO  YOU STILL WISH TO SAVE THE SHA
      PE TABLE  THAT IS CURRENTLY IN MEMORY (Y/N)?"
      ;: GET Z$: IF Z$ <  > "Y" AND Z$ <  > "N" THEN
      5820
5830  IF Z$ = "N" THEN  RETURN
5840  GOTO 5800
6000  HOME : PRINT  TAB( 5);"TYPE THE WORD ";: FLASH
      : PRINT "DELETE";: NORMAL : PRINT " TO DESTRO
      Y": PRINT "THE SHAPE TABLE THAT IS CURRENTLY
      IN    MEMORY.  TYPE ";: FLASH : PRINT "SAVE";

6002  NORMAL : PRINT " IF YOU DO NOT WISH TODESTRO
      Y THE SHAPE TABLE THAT CURRENTLY  IS IN MEMOR
      Y."
6005  PRINT : INPUT "     PRESS THE RETURN KEY AFT
      ER YOUR    CHOSEN ENTRY -> ";Z$
6010  IF Z$ <  > "DELETE" AND Z$ <  > "SAVE" THEN
      6000
6020  IF Z$ = "DELETE" THEN  RUN
6030  RETURN
8000  HOME : PRINT  TAB( 5);"ENTER THE NAME OF THE
      TABLE, THEN  PRESS RETURN.  THE TOTAL LENGT
      H CAN NOT EXCEED 30 CHARACTERS, AND THE FIRST
          CHARACTER MUST BE A LETTER."
8005 NA$ = "":X = 2:Y = 6: HTAB X: VTAB Y
8010  GET Z$
8020  IF  LEN (NA$) = 0 AND  ASC (Z$) < 65 OR  LEN
      (NA$) = 0 AND  ASC (Z$) > 90 THEN  VTAB 10: HTAB
      1: PRINT "THE FIRST CHARACTER MUST BE A LETTE
      R.";: FOR I = 1 TO 1500: NEXT I: HTAB 1: CALL
       - 868: HTAB X: VTAB Y: GOTO 8010
8030  IF Z$ = "," THEN  VTAB 10: HTAB 1: PRINT "DO
      NOT USE ANY COMMAS";: FOR I = 1 TO 1500: NEXT
      I: HTAB 1: CALL  - 868: HTAB X: VTAB Y: GOTO
      8010
8040  IF  ASC (Z$) = 8 AND  LEN (NA$) > 1 THEN X =
      X - 1: HTAB X: CALL  - 868:NA$ =  LEFT$ (NA$,
       LEN (NA$) - 1): GOTO 8010
8050  IF  ASC (Z$) = 8 AND  LEN (NA$) = 1 THEN X =
      X - 1: HTAB X: CALL  - 868:NA$ = "": GOTO 801
      0
```

```
8055  IF  ASC (Z$) = 13 OR  LEN (NA$) > 29 THEN  RETURN

8060  PRINT Z$;:NA$ = NA$ + Z$:X = X + 1: HTAB X: GOTO
      8010
8070  IF  ASC (Z$) = 13 THEN  RETURN
20000 ER =  PEEK (222):LN =  PEEK (218) +  PEEK (2
      19) * 256
20010  IF LN = 3660 THEN 21000: REM   FILE NOT FOU
      ND ERROR WHEN ATTEMPTING TO LOAD A SHAPE TABL
      E
20020  IF ER = 11 AND LN = 5810 THEN  PRINT "FIRST
       CHARACTER IN FILE NAME MUST BE    A LETTER,
      AND NO COMMAS MAY APPEAR IN   THE NAME.  PRES
      S ANY KEY TO CONTINUE.": GET Z$:Z$ = "4": GOTO
      4150
20050  STOP
21000  POKE 34,7: HOME : PRINT  TAB( 5);"YOUR INPU
      T FILE NAME DOES NOT EXIST ON DISKETTE.  DO Y
      OU WISH TO SEE A      CATALOG LISTING OF THE
      DISKETTE THAT IS CURRENTLY IN THE DRIVE (Y/N)
      ?": POKE 34,0
21010  GET Z$: IF Z$ <  > "Y" AND Z$ <  > "N" THEN
      21000
21020  IF Z$ = "N" THEN  GOTO 21050
21030  PRINT : PRINT D$"CATALOG"
21040  PRINT : PRINT  TAB( 5);"PRESS ANY LETTER TO
       CONTINUE.": GET Z$
21050  POKE 216,0: GOTO 4000
30000  END
```

## Program 2. Key Shape Loader Maker

```
10 D$ =  CHR$ (4)
15  PRINT D$"MON C,I,O"
20  PRINT D$"OPEN KEY SHAPE LOADER"
30  PRINT D$"WRITE KEY SHAPE LOADER"
40  PRINT "POKE 104,96"
50  PRINT "POKE 103,1"
60  PRINT "POKE 24576,0"
70  PRINT "RUN KEY SHAPE MAKER"
80  PRINT D$"CLOSE KEY SHAPE LOADER"
90  END
```

# Apple Hi-Res Painter

James Totten

*"Hi-Res Painter" is a graphics editor for use with a 32K Apple. It lets you use any one of six colors (or combine several colors into new ones); select from three different drawing pens; label pictures with upper- and lowercase lettering; color in squares and rectangles; and more. A color monitor is desirable but not required.*

When manually using the Apple's hi-res graphics, a lot of work is required to get even modest results. Since I use the graphics quite often (they are one reason I bought the computer), I didn't like spending hours to draw a fairly impressive title page, chart, or other picture. I needed a useful graphics utility, and "Hi-Res Painter" was the result.

## Menu Options

Hi-Res Painter runs from four menus: the main menu, the accessory menu, the disk menu, and the picture menu.

When you start, you automatically get the main menu. From there you can go to any of the other menus simply by pressing the first letter of its name. That letter is highlighted on the screen.

Pressing A will take you to the accessory menu. There, you can choose from Print, Fill, Keyboard, and Main. The Print option will work only for those who own either a Trendcom or Silentype printer.

However, the Fill option works for everyone. Select two points on the screen, one at the upper-left corner of the square you want to fill, and the other at the lower-right corner. Presto! The keyboard option lets you change from paddle or joystick control to keyboard control. With keyboard control, move the pen with the I, J, K, and M keys, as well as the U, N, O and comma keys. Use the Main option to return to the main menu.

The next menu is the disk menu, called by pressing D. It lets you Name, Delete, Load, or Rename any picture; Save will save the picture currently on the screen. Again, Main will return you to the main menu.

The final menu, the picture menu, is called by pressing P. The available options are View, Label, Drop (called by pressing

B), Color, Draw, Erase, Pens, and Main. The first option gives you a complete view (without text) of the graphics screen that you are working on; if you use it, remember to press M to get back to the main menu. Label will let you do just that, asking you for a date, name, or whatever, which is then transferred to the graphics screen.

The Drop option fills the screen (rather quickly) with a color of your choice. Color will allow you to choose a new color. Press the first letter of each as in the menu selections.

Use of Draw is obvious. To draw continuous lines simply move the cursor. You can also draw lines radiating from a single point. Experiment with your joysticks or paddles to get a feel for this option's many capabilities.

Use of Erase is straightforward, too. However, a note of warning is in order. If a picture is erased, it cannot be recalled unless it is on disk.

The Pens option, used with Draw, is actually two options in one. With it you can change the size of your pen (press 1, 2, or 3 and watch the screen), as well as turn your pen on or off. And again, pressing M returns you to the main menu. You can draw using paddles or a joystick, or you can switch the controls to use the keyboard.

To produce the most interesting designs, do not be afraid to try some experimentation. Fantastic pictures can be drawn much more easily than you might expect, and you'll soon find that this program is bringing out talents you never knew you had.

## Hi-Res Painter

```
20  LOMEM: 24576: ONERR  GOTO 1045
21  DIM PX(2),PY(2),C$(6),P$(1)
25  FOR L = 1 TO 4:MX(L) = 0:MY(L) = 0: NEXT L:D$ =
      CHR$ (4):C = 3:P = 0:BC = 0
30 KI =  - 16384:RK =  - 16368:B0 =  - 16287:B1 =
     - 16286:TG =  - 16301:FG =  - 16302
35 P$(0) = "OFF":P$(1) = "ON":C$(1) = "GREEN":C$(2
    ) = "PINK":C$(3) = "WHITE"
40 C$(4) = "BLACK":C$(5) = "ORANGE":C$(6) = "LT.BL
    UE":I = 1:P$ = "NOT NAMED"
41  IF  PEEK (233) <  > 64 THEN  PRINT D$"BLOAD CH
    ARACTERS/SH2": POKE 232,0: POKE 233,64
42  SCALE= 1: ROT= 0:X = 139:Y = 80
```

```
43  TEXT : HOME : NORMAL : VTAB 10: PRINT  TAB( 11
    )"THE HI-RES PAINTER": PRINT  TAB( 7)"-=(
                    )=-": PRINT  TAB( 11)"BY JAME
    S R. TOTTEN"
44  POKE RK,0: VTAB 24: PRINT "<< TO BEGIN PUSH AN
    Y KEY EXCEPT RESET >>"
45  IF  PEEK (KI) < 128 THEN 45
46  POKE RK,0
50  HGR : HCOLOR= C: POKE TG,0: POKE 34,20: HOME
55  PRINT "PAINTER MENU NUMBER 1 (MAIN)": PRINT
60  PRINT "A)CCESSORY   D)ISKETTE   P)ICTURE   >";
    : GET K$
65  IF K$ =  CHR$ (27) THEN  POKE RK,0: POKE 34,0:
      TEXT : HOME : END
70  IF K$ = "P" THEN 100
75  IF K$ = "A" THEN 450
80  IF K$ = "D" THEN 300
85  POKE RK,0: HOME : GOTO 55
100  POKE RK,0: HOME
105  PRINT "PAINTER MENU NUMBER 4 (PICTURE)": PRINT

110  PRINT "V)IEW  L)ABEL  B)DROP  C)OLOR
       D)RAW  E)RASE  P)ENS   M)AIN   >";: GET K$
115  IF K$ = "M" THEN 85
120  IF K$ =  CHR$ (27) THEN  POKE RK,0: POKE 34,0
    : TEXT : HOME : END
125  IF K$ = "E" THEN  HGR :BC = 0: GOTO 100
130  IF K$ = "V" THEN 145
132  IF K$ = "C" THEN 150
134  IF K$ = "B" THEN 240
136  IF K$ = "D" THEN 185
138  IF K$ = "P" THEN 164
140  IF K$ = "L" THEN 218
142  POKE RK,0: HOME : GOTO 105
145  POKE FG,0
146  IF  PEEK (KI) > 127 THEN  POKE TG,0: GOTO 100

147  GOTO 146
150  POKE RK,0: HOME : PRINT "CURRENT COLOR
    : ";: INVERSE : PRINT C$(C): NORMAL : PRINT
152  PRINT "G)REEN   O)RANGE   W)HITE
       B)LACK   L)T.BLUE  P)INK    >";: GET K$
154  IF K$ = "G" THEN C = 1: GOTO 100
155  IF K$ = "P" THEN C = 2: GOTO 100
156  IF K$ = "W" THEN C = 3: GOTO 100
158  IF K$ = "B" THEN C = 4: GOTO 100
159  IF K$ = "O" THEN C = 5: GOTO 100
160  IF K$ = "L" THEN C = 6: GOTO 100
162  GOTO 150
```

```
164 XC =  INT ( PDL (Ø)):YC =  INT ( PDL (1))
165  POKE RK,Ø: HOME : PRINT "PEN OPERATIONS": PRINT

166  PRINT "S)ET CURSOR SIZE     T)URN ON/OFF  >";:
      GET K$
167  IF K$ = "S" THEN 172
168  IF K$ <  > "T" THEN 165
169 P = P + 1: IF P > 1 THEN P = Ø
170  HOME : PRINT : PRINT "PEN IS NOW "P$(P): FOR
     L = 1 TO 3ØØ: NEXT L
171  GOTO 1ØØ
172  POKE RK,Ø: HOME : PRINT "TYPE A NUMBER FROM 1
      TO 3 FOR CURSOR    SIZE (1=SMALLEST). CURSOR
      IS SHOWN ON   SCREEN. WHEN DONE, PUSH RETURN
     .  >";: GET K$
174  IF K$ =  CHR$ (13) THEN 1ØØ
176  IF K$ = "1" THEN CS = Ø
177  IF K$ = "2" THEN CS = 4
178  IF K$ = "3" THEN CS = 8
179  HCOLOR= BC: FOR L = XC - 1 TO XC + 8: HPLOT L
     ,YC - 1 TO L,YC + 8: NEXT L: HCOLOR= C
18Ø  FOR L = XC TO XC + CS: HPLOT L,YC TO L,YC + C
     S: NEXT L
182  GOTO 172
185  IF K THEN 1Ø1Ø
186  POKE RK,Ø: HOME : PRINT : PRINT "TO BEGIN OR
     STOP DRAWING PUSH ANY KEY ";: GET K$
187  POKE FG,Ø: POKE RK,Ø
19Ø  IF CS = Ø THEN LL = 1:RL = 279:TL = Ø:BL = 19
     1
191  IF CS = 4 THEN LL = 1:RL = 274:TL = Ø:BL = 18
     6
192  IF CS = 8 THEN LL = 1:RL = 27Ø:TL = Ø:BL = 18
     2
194  HCOLOR= C
196 X =  INT ( PDL (Ø)):Y =  INT ( PDL (1))
198  IF X < LL THEN X = LL
2ØØ  IF X > RL THEN X = RL
2Ø2  IF Y > BL THEN Y = BL
2Ø4  FOR L = X TO X + CS: HPLOT L,Y TO L,Y + CS: NEXT
     L
2Ø5  IF  PEEK (KI) > 127 THEN  POKE TG,Ø: GOTO 1ØØ

2Ø6  IF P THEN 21Ø
2Ø8  HCOLOR= BC: FOR L = X TO X + CS: HPLOT L,Y TO
     L,Y + CS: NEXT L: HCOLOR= C
2Ø9  IF  PEEK (KI) > 127 THEN  POKE TG,Ø: GOTO 1ØØ

21Ø  IF CS = Ø THEN  IF  PEEK (B1) > 127 THEN  CALL
      - 198:XO = X:YO = Y
```

```
212  IF CS = Ø THEN  IF  PEEK (BØ) > 127 THEN  HPLOT
     X,Y TO XO,YO
215  GOTO 196
218  POKE RK,Ø: HOME : PRINT : INPUT "ENTER LABEL
     >";L$
219  IF L$ = "" THEN 218
220  HOME : PRINT : PRINT "DO YOU WANT IT ON TOP O
     R BOTTOM (T/B)? ";: GET K$
222  IF K$ = "B" THEN Y = 180: GOTO 226
224  IF K$ = "T" THEN Y = 6: GOTO 226
225  GOTO 220
226 L =  LEN (L$): IF L > 26 THEN 218
228 X = 137 -  INT ((L / 2) * 8)
230  FOR P = 1 TO L: IF  ASC ( MID$ (L$,P,1)) < 62
      THEN K =  ASC ( MID$ (L$,P,1)) - 31: GOTO 23
     2
231 K =  ASC ( MID$ (L$,P,1)) - 3
232  HCOLOR= Ø: FOR L = X - 2 TO X + 7: HPLOT L,Y -
     1 TO L,Y + 8: NEXT L: HCOLOR= 3
233  DRAW K AT X,Y:X = X + 8: NEXT P
234  HCOLOR= C: GOTO 100
240  POKE RK,Ø: HOME : PRINT "COLORS FOR BACKDROP.
     ..": PRINT : PRINT "G)REEN  B)LUE  P)INK  W)H
     ITE  O)RANGE": PRINT ">";: GET K$
242  IF K$ = "G" THEN  HCOLOR= 1:BC = 1: GOTO 248
243  IF K$ = "B" THEN  HCOLOR= 6:BC = 6: GOTO 248
244  IF K$ = "P" THEN  HCOLOR= 2:BC = 2: GOTO 248
245  IF K$ = "W" THEN  HCOLOR= 3:BC = 3: GOTO 248
246  IF K$ = "O" THEN  HCOLOR= 5:BC = 5: GOTO 248
247  GOTO 240
248  HPLOT Ø,Ø: CALL 62454
250 BD = 1: GOTO 100
300  POKE RK,Ø: HOME
302  PRINT "PAINTER MENU NUMBER 3 (DISKETTE)": PRINT
304  PRINT "N)AME  D)ELETE  S)AVE
       L)OAD  R)ENAME  M)AIN   >";: GET K$
306  IF K$ = "M" THEN 85
308  IF K$ =  CHR$ (27) THEN  POKE RK,Ø: POKE 34,Ø
     : TEXT : HOME : END
310  IF K$ = "N" THEN 320
311  IF K$ = "S" THEN 335
312  IF K$ = "L" THEN 355
313  IF K$ = "R" THEN 385
314  IF K$ = "D" THEN 370
315  GOTO 300
320  POKE RK,Ø: HOME : PRINT "USE NO COMMAS OR COL
     ONS IN NAME.": PRINT : INPUT "> ";P$
325  IF P$ = "" THEN 320
330  HOME : PRINT "NAME: "P$: NORMAL
```

```
332  PRINT : PRINT "IS THIS CORRECT? ";: GET K$: IF
     K$ = "N" THEN 320
333  IF K$ = "Y" THEN 300
334  POKE RK,0: GOTO 330
335  IF P$ = "NOT NAMED" THEN  HOME : CALL  - 198:
      POKE RK,0: PRINT : PRINT "PICTURE HAS NOT BE
     EN NAMED": FOR L = 1 TO 550: NEXT L: GOTO 300

340  POKE RK,0: HOME : PRINT "PICTURE NAME: "P$: PRINT
345  PRINT "SAVE WITH THIS NAME? ";: GET K$: PRINT
     K$: IF K$ = "Y" THEN 350
346  IF K$ = "N" THEN 300
347  GOTO 340
350  PRINT D$"BSAVE "P$",A$2000,L$1FFF": GOTO 300
355  POKE RK,0: HOME : PRINT : INPUT "NAME? ";P$
356  IF P$ = "" THEN 355
358  HOME : PRINT "PICTURE NAME: "P$: PRINT
360  PRINT "IS THIS NAME CORRECT? ";: GET K$: PRINT
     K$
362  IF K$ = "N" THEN 300
363  IF K$ = "Y" THEN 365
364  GOTO 358
365  PRINT D$"BLOAD "P$
366  GOTO 300
370  POKE RK,0: HOME : PRINT : INPUT "NAME? ";P$
371  IF P$ = "" THEN 370
372  HOME : PRINT "PICTURE NAME: "P$: PRINT
375  PRINT "DELETE THIS PICTURE? ";: GET K$: PRINT
     K$
376  IF K$ = "Y" THEN 380
377  IF K$ = "N" THEN 300
378  GOTO 372
380  PRINT D$"DELETE "P$: GOTO 300
385  POKE RK,0: HOME : PRINT "USE NO COMMAS OR COL
     ONS IN NEW NAME": PRINT
388  INPUT "CURRENT NAME? ";P1$: IF P1$ = "" THEN
     385
390  INPUT "NEW NAME? ";P2$: IF P2$ = "" THEN 385
393  HOME : PRINT "OLD NAME: "P1$: PRINT "NEW NAME
     : "P2$: PRINT
395  PRINT "ARE THESE BOTH CORRECT? ";: GET K$: PRINT
     K$: IF K$ = "N" THEN 385
396  IF K$ = "Y" THEN 400
398  GOTO 393
400  PRINT D$"RENAME "P1$","P2$: GOTO 300
450  POKE RK,0: HOME
452  PRINT "PAINTER MENU NUMBER 2 (ACCESSORY)": PRINT
454  PRINT "P)RINT  F)ILL  K)EYBOARD  M)AIN    >";
     : GET K$
456  IF K$ = "M" THEN  POKE RK,0: HOME : GOTO 55
```

```
458  IF K$ =  CHR$ (27) THEN  TEXT : POKE RK,Ø: HOME
     : END
459  IF K$ = "P" THEN 475
460  IF K$ = "F" THEN 500
461  IF K$ = "K" THEN 465
462  GOTO 450
465  POKE RK,Ø: HOME : IF K THEN K = Ø: GOTO 468
466  IF  NOT K THEN K = 1
468  IF K = Ø THEN  PRINT : PRINT "KEYBOARD IS OFF
     "
469  IF K = 1 THEN  PRINT : PRINT "KEYBOARD IS ON"

470  FOR L = 1 TO 300: NEXT L: GOTO 450
475  POKE RK,Ø: HOME : PRINT "PICTURE PRINTING OPT
     IONS -": PRINT
476  PRINT "I)NVERSED  N)ORMAL
       R)OTATED   C)ONTINUE     >";: GET K$
478  IF K$ = "N" THEN ST = Ø: GOTO 475
480  IF K$ = "I" THEN ST = 1: GOTO 475
482  IF K$ = "R" THEN RR = 1: GOTO 475
484  IF K$ = "C" THEN 488
486  GOTO 475
488  POKE RK,Ø: HOME : PRINT : PRINT "TURN PRINTER
      ON AND PRESS ANY KEY ";: GET K$
490  IF RR AND ST THEN  POKE 1145,88: CALL  - 1603
     8: GOTO 450
492  IF RR THEN  POKE 1145,120: CALL  - 16038: GOTO
     450
494  IF ST THEN  POKE 1400,Ø: CALL  - 16036: GOTO
     450
496  CALL  - 16044: GOTO 450
500  POKE RK,Ø: HOME : INPUT "UPPER LEFT POINT (X,
     Y)  >";UX$,UY$: IF UX$ = "" OR UY$ = "" THEN
     500
505  IF ( VAL (UX$) < Ø) OR ( VAL (UX$) > 279) THEN
     500
506  IF ( VAL (LY$) < Ø) OR ( VAL (LY$) > 191) THEN
      VTAB  PEEK (37): GOTO 507
507  INPUT "LOWER RIGHT POINT (X,Y)  >";LX$,LY$: IF
     LX$ = "" OR LY$ = "" THEN  VTAB  PEEK (37): GOTO
     507
508  IF ( VAL (LX$) < Ø) OR ( VAL (LX$) > 279) THEN
      VTAB  PEEK (37): GOTO 507
510  HOME : PRINT : PRINT "PRESS A KEY TO BEGIN FI
     LL ";: GET K$: PRINT K$
511  HCOLOR= C
515  FOR L =  VAL (UX$) TO  VAL (LX$): HPLOT L, VAL
     (UY$) TO L, VAL (LY$): NEXT L
520  GOTO 450
```

```
1010  POKE RK,0: HOME : PRINT : PRINT "TO BEGIN OR
      STOP DRAWING PUSH RETURN ";: GET K$
1012  POKE FG,0: POKE RK,0
1015  IF CS = 0 THEN LL = 1:RL = 279:TL = 0:BL = 1
      91
1016  IF CS = 4 THEN LL = 1:RL = 274:TL = 0:BL = 1
      86
1017  IF CS = 8 THEN LL = 1:RL = 270:TL = 0:BL = 1
      82
1018  HCOLOR= C
1019  FOR L = X TO X + CS: HPLOT L,Y TO L,Y + CS: NEXT
      L
1020  IF  NOT P THEN  HCOLOR= BC: FOR L = X TO X +
      CS: HPLOT L,Y TO L,Y + CS: NEXT L: HCOLOR= C
1021  IF  PEEK (KI) < 128 THEN 1019
1023 L =  PEEK (KI)
1024  IF L = 201 THEN Y = Y - 1: GOTO 1036
1025  IF L = 205 THEN Y = Y + 1: GOTO 1036
1026  IF L = 202 THEN X = X - 1: GOTO 1036
1027  IF L = 203 THEN X = X + 1: GOTO 1036
1028  IF L = 213 THEN X = X - 1:Y = Y - 1: GOTO 10
      36
1029  IF L = 206 THEN X = X - 1:Y = Y + 1: GOTO 10
      36
1030  IF L = 207 THEN X = X + 1:Y = Y - 1: GOTO 10
      36
1031  IF L = 172 THEN X = X + 1:Y = Y + 1: GOTO 10
      36
1032  IF (CS = 0) AND (L = 211) THEN XO = X:YO = Y
      : CALL  - 198: GOTO 1036
1033  IF (CS = 0) AND (L = 196) THEN  HPLOT X,Y TO
      XO,YO: GOTO 1036
1034  IF L = 141 THEN  POKE TG,0: GOTO 100
1035  POKE RK,0: GOTO 1021
1036  IF X < LL THEN X = LL
1037  IF X > RL THEN X = RL
1038  IF Y > BL THEN Y = BL
1039  IF Y < TL THEN Y = TL
1040  POKE RK,0: GOTO 1019
1045  HOME : PRINT : PRINT "DISK ERROR CODE" PEEK
      (222): PRINT "CHECK SYNTAX AND TRY AGAIN": PRINT
       CHR$ (7);: GET K$
1050  POKE RK,0: HOME : GOTO 55
```

# 3-D Plotting

Tim R. Colvin

*How many times have you admired those three-dimensional graphics plots in ads for video monitors and printers? Now, with these easy-to-use programs, you can create three-dimensional images of your own.*

These two programs, "Rectan" and "Spheri," will plot three-dimensional figures using information that you provide.

You don't really need to delve into the mathematics which produce the images. You can just fiddle with the examples given to produce many effective displays. Let's look at some graphic examples. First type in each program and save it to tape or disk.

Then load Rectan (Program 1). To have Rectan draw a hyperbolic paraboloid, or "saddle function" (it resembles a riding saddle), let line 790 be the following:

```
790 Z=X*X/4-Y*Y/9
```

and give the following inputs:

```
-2,2,-3,3,25,25,45
```

For another interesting design, use this line:

```
790 Z=-1/(X*X+Y*Y+.5)
```

and give the following inputs:

```
-1,1,-1,1,20,20,45
```

The program will print SCREEN SCALING IN PROGRESS. The program is scaling the image to fit on the screen, which can require a lot of time. The rule is: The more complicated the description of the surface, the longer that step takes.

## The Plotting Begins

When the previous step is completed, the screen will clear and high-resolution plotting will begin. When the plot is finished, the color of the top left corner of the screen will change. The program will be locked in a loop, so you can admire your creation for as long as you wish.

Despite the fine graphics they produce, these programs have a couple of limitations. Screen pixels are taller than they are wide, which makes spheres look slightly less round than they should. Also, you see the surface as if it were transparent

and contour lines were drawn on it. A more advanced program would remove lines that you couldn't see if the surface were not transparent.

## A Spheri Demonstration

To see a torus (doughnut shape), type NEW to clear memory. Then load Spheri (Program 2). Use the following lines for lines 820–840:

```
820 XT=(4+C1)*C2
830 YT=(4+C1)*S2
840 ZT=S1
```

and give the following inputs:

```
0,360,0,360,25,25,45
```

If you prefer to draw a sphere, use these lines instead:

```
XT=C1*C2
YT=C1*S2
ZT=S1
```

and give the following inputs:

```
0,360,0,180,15,15,45
```

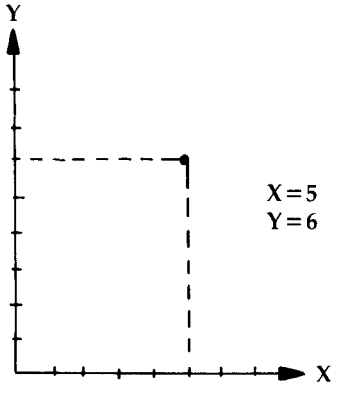
## An Illusion of Depth

These programs use *rectangular* and *spherical coordinate systems* to create an illusion of depth in the screen image. You're probably familiar with the X-Y coordinate system used to specify the location of a point on a flat surface. For example, in Figure 1 the point is located five units over on the X axis and six units up on the Y axis. The point is said to be at location 5,6.

Such a simple system works well for specifying the location of a point in a two-dimensional design on a flat surface, but for 3-D plotting you need a third coordinate.

Several coordinate systems are commonly used to plot three-dimensional surfaces. The particular system you should use depends on the shape you want to draw. Any system can be used, but if you choose the right system, you can simplify your calculations considerably.

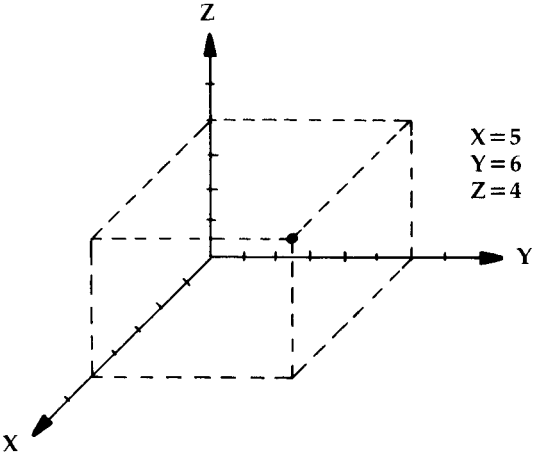
## Figure 1: Two-Dimensional Rectangular Coordinates




## A Simple Solution

The easiest system to understand is simply an extension of the rectangular (X-Y) coordinate system you are already familiar with. All you need to add is a third coordinate (Z) for the third dimension. For example, the point shown in Figure 2 is located five units out on the X axis, six units over on the Y axis, and four units up on the Z axis. The point is said to be at location 5,6,4.

## Figure 2: Three-Dimensional Rectangular Coordinates

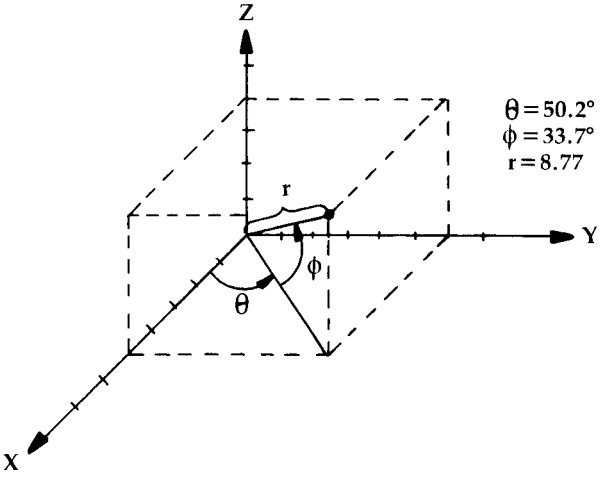


## A System for the Stars

If the design you wish to draw is roughly the shape of a sphere, you should use *spherical coordinates.* In that system, a point is described by two angles and by a distance from the origin. For example, astronomers use spherical coordinates to describe the position of a star relative to the earth. For example, the *azimuthal angle* of a star, designated by the Greek letter theta ( θ ), is the direction you must face to view the star. If north is taken to be zero degrees, then a star that lies due east would have an azimuthal angle of 90 degrees. The *elevation angle,* designated by the Greek letter phi ( ϕ ), specifies how much you must tilt your head back to look directly at the star. If the horizon is taken to be zero degrees, a star that is directly overhead would have an elevation angle of 90 degrees. Finally, the *radial distance,* designated by the letter r, is the distance from Earth to that star.

Using spherical coordinates, the point shown in Figure 2 has an azimuthal angle of 50.2 degrees, an elevation angle of 33.7 degrees, and a radial distance of 8.77 units, as shown in Figure 3.

## Figure 3: Spherical Coordinates



## The Mathematics of 3-D Plotting

Rectan plots surfaces using rectangular coordinates (X,Y,Z). The values for X and Y are specified; the value of Z is then given by Z=f(X,Y) for some function f.

To use Rectan, specify the function f(X,Y) in line 790. For example, Z=X*X/4−7*Y/9 defines a hyperbolic paraboloid.

Spheri plots surfaces using spherical coordinates. This method describes a point on the surface using three parameters: radial distance from the origin, r; azimuthal angle, theta ( θ ); and elevation angle, phi ( ϕ ).

To use Spheri, specify X,Y, and Z (called XT,YT, and ZT in lines 820–840) as functions of r, theta, and phi in lines 820–840.

## Parameters and Slices

The two programs are structured in much the same way, allowing you to specify parameter ranges. In Rectan you choose ranges for X and Y; in Spheri, you select ranges for theta and phi.

You will also be asked to enter the number of *slices* for the parameters. Each slice corresponds to a contour line on the surface. A contour line is a line along which one of the parameters is held constant.

Finally, you specify an observation angle. That is the angle which allows you to see a three-dimensional surface on a two-dimensional video screen. The most commonly used angle is 45 degrees.

## Program 1. Rectan

```
100  HCOLOR= 3
130  HOME
140  INPUT "LOWER X LIMIT:";A1
150  INPUT "UPPER X LIMIT:";B1
160  INPUT "LOWER Y LIMIT:";A2
170  INPUT "UPPER Y LIMIT:";B2
180  INPUT "SLICES IN X:";N
190  INPUT "SLICES IN Y:";M
200  INPUT "OBSERVATION ANGLE:";Q
210  PRINT "SCREEN SCALING IN PROGRESS"
215 U = .0174532925
220 Q = Q * U
230 CS =  COS (Q)
240 SI =  SIN (Q)
```

```
250 H1 = (B1 - A1) / 279:H2 = (B2 - A2) / (N - 1)
260 H3 = (B1 - A1) / (M - 1):H4 = (B2 - A2) / 279
270 M1 = 9999999:M2 = M1:N1 =  - M1:N2 = N1
280  FOR Y = A2 TO B2 STEP H2
290  FOR X = A1 TO B1 STEP H1
300  GOSUB 610
310  NEXT
320  NEXT
330  FOR X = A1 TO B1 STEP H3
340  FOR Y = A2 TO B2 STEP H4
350  GOSUB 610
360  NEXT
370  NEXT
380  HGR2
420 T1 = (N1 - M1) / 2
430 T2 = (N2 - M2) / 2
440 W = T1 / T2
450  IF W < 1.46333333 THEN 480
460 XS = 139:ZS = 139 / W
470  GOTO 490
480 XS = 95 * W:ZS = 95
490  FOR Y = A2 TO B2 STEP H2
500  FOR X = A1 TO B1 STEP H1
510  GOSUB 690
520  NEXT
530  NEXT
540  FOR X = A1 TO B1 STEP H3
550  FOR Y = A2 TO B2 STEP H4
560  GOSUB 690
570  NEXT
580  NEXT
600  END
610  GOSUB 790
620 XT = X - Y * CS
630 ZT = Z - Y * SI
640  IF XT > N1 THEN N1 = XT
650  IF XT < M1 THEN M1 = XT
660  IF ZT > N2 THEN N2 = ZT
670  IF ZT < M2 THEN M2 = ZT
680  RETURN
690  GOSUB 790
700 XT = 140 +  INT (XS * (X - Y * CS - N1 + T1) /
     T1)
710 ZT = 96 -  INT (ZS * (Z - Y * SI - N2 + T2) /
     T2)
720  HPLOT XT,ZT
780  RETURN
790 Z =  - 1 / (X * X + Y * Y + .5)
800  RETURN
```

## Program 2. Spheri

```
100  HCOLOR= 3
130  HOME
140  INPUT "LOWER THETA LIMIT:";A1
150  INPUT "UPPER THETA LIMIT:";B1
160  INPUT "LOWER PHI LIMIT:";A2
170  INPUT "UPPER PHI LIMIT:";B2
180  INPUT "SLICES IN THETA:";N
190  INPUT "SLICES IN PHI:";M
200  INPUT "OBSERVATION ANGLE:";Q
210  PRINT "SCREEN SCALING IN PROGRESS"
215 U = .0174532925
220 Q = Q * U
230 CS =  COS (Q)
240 SI =  SIN (Q)
250 H1 = (B1 - A1) / 279:H2 = (B2 - A2) / (N - 1)
260 H3 = (B1 - A1) / (M - 1):H4 = (B2 - A2) / 279
270 M1 = 99999999:M2 = M1:N1 =  - M1:N2 = N1
280  FOR Y = A2 TO B2 STEP H2
290  FOR X = A1 TO B1 STEP H1
300  GOSUB 610
310  NEXT
320  NEXT
330  FOR X = A1 TO B1 STEP H3
340  FOR Y = A2 TO B2 STEP H4
350  GOSUB 610
360  NEXT
370  NEXT
380  HGR2
420 T1 = (N1 - M1) / 2
430 T2 = (N2 - M2) / 2
440 W = T1 / T2
450  IF W < 1.46333333 THEN 480
460 XS = 139:ZS = 139 / W
470  GOTO 490
480 XS = 95 * W:ZS = 95
490  FOR Y = A2 TO B2 STEP H2
500  FOR X = A1 TO B1 STEP H1
510  GOSUB 690
520  NEXT
530  NEXT
540  FOR X = A1 TO B1 STEP H3
550  FOR Y = A2 TO B2 STEP H4
560  GOSUB 690
570  NEXT
580  NEXT
600  END
610  GOSUB 790
```

```
620 XT = XT - YT * CS
630 ZT = ZT - YT * SI
640  IF XT > N1 THEN N1 = XT
650  IF XT < M1 THEN M1 = XT
660  IF ZT > N2 THEN N2 = ZT
670  IF ZT < M2 THEN M2 = ZT
680  RETURN
690  GOSUB 790
700 XT = 140 +  INT (XS * (XT - YT * CS - N1 + T1)
     / T1)
710 ZT = 96 -  INT (ZS * (ZT - YT * SI - N2 + T2) /
    T2)
715  IF XT < 0 THEN XT = 0
716  IF XT > 279 THEN XT = 279
720  HPLOT XT,ZT
780  RETURN
790 XA = X * U:C1 =  COS (XA):S1 =  SIN (XA)
800 YA = Y * U:C2 =  COS (YA):S2 =  SIN (YA)
820 XT = (4 + C1) * C2
830 YT = (4 + C1) * S2
840 ZT = S1
850  RETURN
```

# Spiralizer

Chayim Avinor

*This program uses high-resolution graphics to create dazzling patterns based on complex geometrical principles—and its onscreen menu makes it very easy to use.*

"Spiralizer" is a program for creating patterns on the Apple's high-resolution screen. The results may remind you of pictures drawn with a spirograph, which uses toothed wheels of different sizes to control the motion of a pen. However, Spiralizer can create a far greater array of patterns than its mechanical predecessors.

The patterns are actually made by two radii. One of them is turning around a stationary or linearly moving center (depending on your input), and the center of the other radius is the free end of the first one. You are given control of the relative speed and length of the radii and some additional handy features.

## Using the Program

After typing RUN and RETURN, you are asked to enter the relative speed between the two radii. This is actually the number of loops the pattern is going to have. You can choose an answer between −50 and 50. If you type 4 and RETURN, your pattern will have four complete loops. If you type −6 there will be six loops, but they'll be on the inner side of the pattern.

Use the back arrow to delete a character. If you simply press RETURN without typing in a number, the program will default to a speed value of 5 (and will display it in the menu). Large numbers cause the program to draw straight segments, because of the large steps involved.

A pattern with three loops is easy to imagine, but what would a pattern with two loops look like? How about one loop? Could a pattern possibly have zero loops? Try the numbers and see.

Next, you're asked for the radius, and your input determines the ratio between the radii. You can choose any number between 1 and 60. A small number would make the inner radius small and the outer radius large, and vice versa. If de-

sired, simply press RETURN and take the default value of 35.

After choosing the radius, specify spin. An answer larger than 1 will make the pattern rotate while it's drawn (and, of course, will change the number of loops). You can choose numbers from 1 (no spin) to 18. When spinning, the lines remain smooth and curvy, but it takes more time to draw a complete pattern. If you decide to quit while a pattern is being drawn, press any key and the program will return to the menu. To escape from the program, use RESET.

## Added Features

Now things become more complicated. You are asked to specify movement or decrement. If you choose M, for movement, the whole pattern will move while it is being drawn. For example, if you choose M with a spin of 1, the pattern will be drawn five times while it moves. If the spin is greater than 1, the pattern will move until it finishes rotating. If the spin is greater than 1 but less than 9, you will not be asked for this input.

Pressing D will cause the pattern to decrease in size while being drawn. The rules are the same as for M. If you press RETURN, the default value is NONE, and none of the above actions will take place.

Finally, you are asked if you want to clear the screen. If you decide not to, then the new pattern will be drawn on the previous one. This feature allows you to make interesting overlays of patterns.

For a nice sample, I suggest you try the following INPUTs: for speed, enter 7; for radius, 50; for spin, 18; then choose M for movement and type Y to clear the screen.

Experiment with different values, and you'll see some stunning designs.

## Spiralizer

```
10  ONERR  GOTO 90
60 R$ =  CHR$ (8): HCOLOR= 3:H$ = "     " + R$ + R$
      + R$ + R$
70  HOME : HGR
80  VTAB 9: HTAB 15: FLASH : PRINT "SPIRALIZER": NORMAL

90  POKE  - 16368,0:Z = 5: HTAB 1: VTAB 21: PRINT
     "SPEED (-50,50)? ";: GOSUB 380:K = Z
100  IF Z <  - 50 OR Z > 50 THEN 90
110 K = K - 1
```

```
120 Z = 35: PRINT "RADIUS (1,60)? ";: GOSUB 380:R =
    Z
130  IF Z < 1 OR Z > 60 THEN  VTAB 22: GOTO 120
140 R = R + 13:S = 1
150 Z = 1: PRINT "SPIN (1,18)? ";: GOSUB 380
160  IF Z < 1 OR Z > 18 THEN  VTAB 23: GOTO 150
170 A = 1 / Z: IF Z > 1 AND Z < 9 THEN 240
180 SM = 1:M = 2: PRINT "MOVEMENT OR DECREMENT (M/
    D)? "H$;
190  GET X$: IF X$ =  CHR$ (13) THEN M = 0:SM = 0:
     PRINT "NONE";: GOTO 240
200  IF X$ = "M" THEN SM = 0: GOTO 230
210  IF X$ = "D" THEN M = 0: GOTO 230
220  GOTO 190
230  PRINT X$;
240  VTAB 21: HTAB 24: PRINT "CLEAR (Y/N)? "H$;: GET
    T$: IF T$ <  > "N" THEN  PRINT "YES": TEXT : CALL
    62450: HGR
250  IF T$ = "N" THEN  PRINT "NO"
260 W = 1:Z = 139: IF M = 2 THEN Z = 80: IF A = 1 THEN
    W = 5:M = 1:Z = 122
270  IF SM - A = 0 THEN W = 5
280  IF A < 1 THEN K = K + A
290 C = 0.001: IF A < 1 / 9 THEN M = M / 2:C = C /
    2
300 J = R:I = 79 - R
310  HPLOT Z,0
320  REM -MAIN LOOP-
330  FOR T = 0 TO 6.2831 / A * W STEP 0.06283:F =
     PEEK ( - 16384): POKE  - 16368,0: IF F > 127
     THEN 90
340  IF SM THEN J = R * S:I = 79 * S - J:S = S - C

350  HPLOT  TO Z + T * M -  SIN (T) * J +  SIN (T *
    K) * I,79 -  COS (T) * J -  COS (T * K) * I: NEXT

360  GOTO 90
370  REM -INPUT SUBROUTINE-
380 L0 = 0:L1 = 1:B$ = ""
390  PRINT H$;: GET A$
400  IF A$ = "-" AND L0 = 0 THEN  PRINT A$::B$ = A
    $:L0 = 1:L1 = 2: GOTO 390
```

```
410  IF A$ =  CHR$ (13) AND LØ > Ø THEN Z =  VAL (
     B$): PRINT : RETURN
420  IF A$ =  CHR$ (13) THEN  PRINT Z: RETURN
430  IF A$ = R$ AND LØ > 1 THEN  PRINT A$;:B$ =  LEFT$
     (B$, LEN (B$) - 1):LØ = LØ - 1: GOTO 390
440  IF A$ = R$ AND LØ = 1 THEN  PRINT A$;:B$ = ""
     :LØ = Ø: GOTO 390
450  IF LØ > L1 THEN 390
460  IF A$ < "Ø" OR A$ > "9" THEN 390
470  PRINT A$;:B$ = B$ + A$:LØ = LØ + 1: GOTO 390
```

# Appendix

# A Beginner's Guide to Typing In Programs

# A Beginner's Guide to Typing In Programs

## What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential,* but without a program, it isn't going anywhere. Most of the programs published in this book are written in a computer language called Applesoft BASIC. It is easy to learn and works on the Apple II, II+, IIe, and IIc.

## BASIC Programs

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as O for the numeral 0, a lowercase l for the numeral 1, or an uppercase B for the numeral 8. Also, you must enter all punctuation such as colons and commas just as they appear in the book. Spacing can be important. To be safe, type in the listings *exactly* as they appear.

## About DATA Statements

Some programs contain a section or sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard may seem dead, and the screen may go blank. But don't panic; no damage is done. To regain control, you have to reset your computer (in effect, you turn it off and then turn it back on). That will erase whatever program was in memory, *so always save a copy of your program before you run it.* If your computer crashes, you can load the program and look for your mistake.

Sometimes a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

## Get to Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program, so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you made a mistake? You can always retype the line, but you at least need to know how to backspace. Do you know how to enter reverse video, lowercase, and control characters? It's all explained in your computer's manuals.

## A Quick Review

1. Type in the program a line at a time, in order. Press RETURN at the end of each line. Use backspace or the back arrow to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

# Index

# COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403.

| Quantity | Title | Price* | Total |
|---|---|---|---|
| _____ | COMPUTE!'s Commodore Collection, Volume 1 | **$12.95** | _____ |
| _____ | Commodore Peripherals: A User's Guide | **$ 9.95** | _____ |
| _____ | Creating Arcade Games on the Commodore 64 | **$14.95** | _____ |
| _____ | Machine Language Routines for the Commodore 64 | **$14.95** | _____ |
| _____ | Mapping the Commodore 64 | **$14.95** | _____ |
| _____ | COMPUTE!'s First Book of VIC | **$12.95** | _____ |
| _____ | COMPUTE!'s Second Book of VIC | **$12.95** | _____ |
| _____ | COMPUTE!'s Third Book of VIC | **$12.95** | _____ |
| _____ | COMPUTE!'s First Book of VIC Games | **$12.95** | _____ |
| _____ | COMPUTE!'s Second Book of VIC Games | **$12.95** | _____ |
| _____ | Creating Arcade Games on the VIC | **$12.95** | _____ |
| _____ | Programming the VIC | **$24.95** | _____ |
| _____ | VIC Games for Kids | **$12.95** | _____ |
| _____ | Mapping the VIC | **$14.95** | _____ |
| _____ | The VIC and 64 Tool Kit: BASIC | **$16.95** | _____ |
| _____ | Machine Language for Beginners | **$14.95** | _____ |
| _____ | The Second Book of Machine Language | **$14.95** | _____ |
| _____ | Computing Together: A Parents & Teachers Guide to Computing with Young Children | **$12.95** | _____ |
| _____ | BASIC Programs for Small Computers | **$12.95** | _____ |

*Add $2.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.

**Shipping & handling: $2.00/book** _____
**Total payment** _____

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.
NC residents add 4.5% sales tax.
☐ Payment enclosed.
Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No._____ Exp. Date_____

Name_____

Address_____

City_____ State _____ Zip_____

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

What's the connection between cavern exploration, weather forecasting, and writing a term paper? All can be done on your Apple computer—and *COMPUTE!'s First Book of Apple* shows you how.

From exciting games and educational programs to home applications, programming techniques, graphics routines, and even music and sound, *COMPUTE!'s First Book of Apple* offers a collection of outstanding Apple programs from past issues of *COMPUTE!* magazine. Here's just a sample of what you'll find inside:

- "Astrostorm," a fast-paced arcade game where you pilot a vital supply ship through a perilous storm of asteroids.
- "Caves of Ice," a game that challenges you to escape from a multidimensional cavern.
- "Letter and Number Play" and "Snertle," two educational games for young children.
- "Home Energy Calculator" and "Utility Bill Audit," two programs designed to help you save money.
- "Weather Forecaster," an accurate forecasting package that uses many of the same techniques employed by professional meteorologists.
- "Chemistry Lab," a program that teaches basic chemistry with a colorful, animated display.
- "Apple Sounds," innovative programs that let you create sound effects or compose and play music on your computer.
- "The Apple Hi-Res Painter," a complete drawing package with an impressive catalog of commands.
- "Spiralizer," a routine that creates complex and captivating screen displays.
- And much, much more.

Each program has been thoroughly tested, and all are ready to type in and run. Many can be easily customized, allowing you to tailor their operation to your own needs.

*COMPUTE!'s First Book of Apple* is sure to have something for every Apple owner. It will make an already versatile computer more exciting than ever before.

ISBN 0-942386-69-8